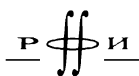


УДК 621.3+681.3
ББК 32.813
Е60



*Издание осуществлено при поддержке
Российского фонда фундаментальных
исследований по проекту 02-01-14148*

Емельянов В. В., Курейчик В. В., Курейчик В. М. **Теория и практика эволюционного моделирования.** — М.: ФИЗМАТЛИТ, 2003. — 432 с. — ISBN 5-9221-0337-7.

Рассматриваются вопросы, связанные с эволюционным развитием сложных систем различной природы. Аналогия эволюционного развития естественных и искусственных систем позволяет развить подходы и методы эволюционного моделирования, генетических оптимизационных алгоритмов, распределенного искусственного интеллекта и искусственной жизни. Описаны генетические и синергетические подходы, а также средства эволюционного моделирования. Представлены современные взгляды на гибридные системы, основанные на имитационных моделях и эволюционном моделировании.

Книга рассчитана на специалистов в области перспективных информационных технологий. Она будет полезна студентам, магистрантам и аспирантам соответствующих специальностей.

Табл. 17. Ил. 236. Библиогр. 207 назв.

Список сокращений

БД	— база данных
БЗ	— база знаний
ГА	— генетический алгоритм
ЕС	— естественная система
ИЖ	— искусственная жизнь
ИИ	— искусственный интеллект
ИС	— искусственная система
ЛПР	— лицо, принимающее решение
МАС	— многоагентная система
ОВ	— оператор вставки
ОИ	— оператор инверсии
ОК	— оператор кроссинговера
ОМ	— оператор мутации
ОР	— оператор рекомбинации
ОС	— оператор сегрегации
ОТ	— оператор транслокации
ОУ	— оператор удаления
ПГА	— простой ГА
РДО	— язык имитационного моделирования
ФП	— функция пригодности
ЦФ	— целевая функция
ЭМ	— эволюционное моделирование
ЭС	— экспертная система

ОГЛАВЛЕНИЕ

Предисловие	6
Введение	9

Г Л А В А 1

ЭВОЛЮЦИЯ ЕСТЕСТВЕННЫХ И ИСКУССТВЕННЫХ СИСТЕМ

1.1. Эволюция Дарвина	19
1.2. Эволюции Ламарка, де Фриза, Поппера и синтетическая теория эволюции	29
1.3. Эволюционная кибернетика	36
1.4. Эволюция сложных систем	40
1.5. Тейлоровские и посттейлоровские организации	45

Г Л А В А 2

ЭВОЛЮЦИЯ И СИНЕРГЕТИКА

2.1. Анализ и построение искусственных систем	50
2.2. Иерархия в интеллектуальных искусственных системах	59
2.3. Порядок и хаос в моделях искусственных систем	64
2.4. Гомеостаз	74
2.5. Фракталы	87

Г Л А В А 3

ГЕНЕТИЧЕСКИЕ АЛГОРИТМЫ

3.1. Определение и структура генетических алгоритмов	91
3.2. Простой генетический алгоритм	103
3.3. Теоремы эволюционного моделирования	110

Г Л А В А 4

ИНСТРУМЕНТАЛЬНЫЕ СРЕДСТВА ЭВОЛЮЦИОННОГО МОДЕЛИРОВАНИЯ

4.1. Технологии локального и генетического поиска	121
4.2. Архитектуры и стратегии эволюционного моделирования	136
4.3. Модифицированные генетические операторы	158
4.4. Генетическое программирование	167

Г Л А В А 5

ГИБРИДНЫЕ СИСТЕМЫ

5.1. Генетические алгоритмы и имитационное моделирование	176
5.2. РДО-модель простого генетического алгоритма	181
5.2.1. Задача оптимизации	181
5.2.2. Типы ресурсов и ресурсы	182

5.2.3. Основные параметры	184
5.2.4. Схема работы	184
5.2.5. Реализация блоков	187
5.3. Система с подстройкой параметров генетических алгоритмов	195
5.4. Эволюция популяции автоматов	201
5.5. Многоагентные системы	208
5.5.1. Взаимодействие агентов	208
5.5.2. Транспортная задача и структура системы	210
5.5.3. Функционирование мобильного агента	211
5.5.4. Организация взаимодействия агентов с помощью генетических алгоритмов	216
5.6. Искусственная жизнь и многоагентная система	218
5.7. Многомодельные системы	226
5.8. Генетический поиск с миграцией особей	229

Г Л А В А 6

ОПТИМИЗАЦИОННЫЕ ЗАДАЧИ НА ГРАФАХ

6.1. Постановка оптимизационных задач	233
6.2. Генетические алгоритмы разбиения графов	243
6.3. Размещение вершин графов в линейке и на плоскости	264
6.4. Генетические алгоритмы построения деревьев Штейнера	276
6.5. Трассировка соединений	280
6.6. Решение задач о коммивояжере методами моделирования эволюций	293
6.7. Задачи раскраски, построения клика и независимых множеств графов	302
6.8. Определение планарности графов на основе генетического поиска	311
6.9. Определение изоморфизма графов	333

Г Л А В А 7

АЛГОРИТМЫ РЕШЕНИЯ ЗАДАЧ НА ГРАФАХ

7.1. Программная среда	341
7.2. Разбиение графа на части по критерию числа внешних ребер	350
7.3. Исследование генетического алгоритма размещения	357
7.4. Исследование задачи о коммивояжере	359
7.5. Задачи построения независимых множеств, раскраски графа	362
7.6. Задачи распознавания изоморфизма графа	365

Г Л А В А 8

ПРИКЛАДНЫЕ ОПТИМИЗАЦИОННЫЕ ЗАДАЧИ

8.1. Задачи планирования кристалла и сжатия топологии	369
8.2. Упаковка блоков	374
8.3. Решение задачи плоской раскладки	377
8.4. Планирование поставок на многопродуктовый склад	386
8.5. Планирование работы производственного участка	393
8.6. Динамический оптимальный раскрой материала	400
8.7. Задача оптимальной раскладки грузов на поддоне	409
Заключение	414
Список литературы	416
Предметный указатель	427

Предисловие

При построении общей стратегии создания интеллектуальных искусственных систем перед учеными все время возникают вопросы, связанные с развитием природы. Какие пути эволюции при этом выбирает природа? Почему природа экономна? Почему она идет оптимальным путем? Каким образом природе удастся найти наиболее устойчивые формы? Каким образом происходит самоорганизация? Какие механизмы лежат в основе алгоритмов построения порядка из хаоса? Как природа научилась ускорять эволюцию? И многие другие. В. Вернадский писал: «Какое наслаждение «вопрошатель природы»! Какой рой вопросов, мыслей, соображений! Сколько причин для удивления, сколько ощущений приятного при попытке объяснить своим умом, воспроизвести в себе ту работу, какая длилась веками в бесконечных ее областях!» Д. Дидро говорил: «Природе доставляет удовольствие варьировать один и тот же механизм бесконечным числом различных способов».

Считается, что история эволюционного моделирования или эволюционных вычислений началась с работ Дж. Холланда, Л. Фогеля, А. Овена и М. Уолша, И. Букатовой, Л. Растригина и других исследователей. Все они взяли за основу ряд преобразований живого материала существующих в природе, упростили их, построив ряд принципов и моделей эволюционных процессов. Со временем эволюционное моделирование превратилось в теорию, на основе которой осуществляется поиск квазиоптимальных и оптимальных решений задач, некоторые из них считались до этого неразрешимыми.

Основное преимущество эволюционного моделирования заключается в возможности решения задач, имеющих много локальных оптимумов за счет комбинирования элементов случайности и направленности аналогично тому, как это происходит в природе. Другим важным фактором эволюционного моделирования является моделирование процессов селекции, размножения и наследования. При этом получаемые по определенным правилам альтернативные решения могут порождать новые решения, которые будут «наследовать» лучшие характеристики предыдущих. Суть стратегии эволюционного моделирования состоит в реализации целенаправленного процесса размножения-гибели, при котором размножению соответствует появление новых объектов, а гибели — удаление объектов в соответствии с определенными критериями естественного и искусственного отбора.

Моделирование развития и совершенствования природы позволяет найти новые пути построения интеллектуальных искусственных систем. Основными направлениями здесь могут выступить эволюционные стратегии, генетические алгоритмы, а также генетическое программирование.

Все, даже наиболее сложные структуры интеллектуальных искусственных систем строятся на основе дискретного подхода как некой общей основы. Существует особый строительный блок — «универсальный кирпич», который затем достраивается, входит в сложные иерархические конгломераты по различным нечетким алгоритмам. Искусственные системы имеют фрактальное строение и повторяются в различных масштабах. Применение генетических алгоритмов позволяет получать набор альтернативных решений, среди которых с большой вероятностью находится оптимальное.

Природа питает предпочтение к определенным формам, чаще всего встречающимся в естественных системах. Использование этих форм приводит к быстрой эволюции систем (устойчивым формам), т. е. установлению гомеостаза. Относительно устойчивые структуры, на которые выходят процессы эволюции в открытых и нелинейных системах, самоорганизуются. Процессы самоорганизации являются ключевыми понятиями новой развивающейся науки — эволюционного моделирования.

Даже на упрощенных математических моделях искусственных и естественных систем можно видеть все разнообразие возможных путей эволюции. Эволюция имеет сквозной характер. Она пронизывает все уровни естественных и искусственных систем. Основой эволюции является отбор на основе стратегии — «выживают сильнейшие». Такое развитие через хаос — это медленный путь эволюции природы, путь случайных вариаций и естественного отбора, постепенного перехода от простых самоорганизующихся систем к более сложным. При этом природа часто делает скачки, осуществляя сжатие времени эволюции. Один из основных вопросов современной науки — это изучение взаимодействия части и целого. Он связан с проблемой совместной эволюции и ее ускорения при объединении эволюционирующих частей, при этом способы самоорганизации не произвольны, а обусловлены нелинейными свойствами внешней среды.

В монографии описаны некоторые принципы и подходы к построению интеллектуальных искусственных систем. Обобщены материалы зарубежных, российских ученых и результаты исследований авторов в области эволюционного моделирования. Описана теория эволюционного моделирования и приведены ее основные теоремы. Рассмотрены алгоритмы анализа и синтеза сложных систем и решения оптимизационных задач принятия решений на графах и других моделях в различных проблемных областях. Сложная эволюционирующая система не может быть представлена единственной моделью, как правило, это целый комплекс моделей, каждая из которых отражает различные стороны поведения моделируемой системы. Одновременное использование различных методов для построения моделей эволюционирующих сложных систем приводит к рассмотрению и использованию гибридных систем, объединяющих как единое целое множество подсистем различного типа и назначения.

Таким образом, использование стратегий, принципов, методов и алгоритмов эволюционного моделирования при решении оптимизационных задач является качественно новым перспективным подходом к созданию и применению сложных искусственных систем. Данный подход позволяет

получать эффективные решения в различных областях науки и техники за приемлемое время и с требуемым качеством.

Ядром книги является новая перспективная технология генетического поиска и эволюционного моделирования. Авторы стремились показать, как они используются для решения практических задач. В этой связи книга структурирована следующим образом. Сначала рассмотрены модели эволюции Ч. Дарвина, Ж. Ламарка, де Фриза, К. Попера и синтетическая теория эволюции. Далее показана взаимосвязь между эволюцией и новой, развивающейся наукой — синергетикой. Затем приведена теория эволюционного моделирования и сформулирована концепция построения генетических алгоритмов. На основе указанного материала представлены подходы к построению гибридных систем искусственного интеллекта и разработке эффективных методов решения графовых и прикладных оптимизационных задач.

Монография предназначена для специалистов, ведущих разработки перспективных САПР, интеллектуальных информационных технологий в науке, технике, биологии, экономике, социологии и других областях. Она может быть полезна для аспирантов и студентов всех специальностей, изучающих теорию систем и методы системного анализа, информатику, методы оптимизации, исследования операций и принятия решений.

Данные исследования основаны на работах Н. Винера, И. Пригожина, Г. Хакена, Д. Холланда, Д. Гольдберга, Д. Поспелова, И. Моисеева, А. Мелихова и других российских и зарубежных ученых. Исследования, на базе которых написана книга, проводились в научных школах Московского государственного технического университета им. Н.Э. Баумана (МГТУ) и Таганрогского государственного радиотехнического университета (ТРТУ) по генетическим алгоритмам и эволюционному моделированию. Авторы так же выражают благодарность С. Ясенновскому, С. Родзину, П. Захарову, М. Крючкову, П. Афонину, А. Назаровой, Ю. Разумцовой, Л. Гладкову, А. Бородулину, А. Бондалетову, чьи материалы использованы в монографии. Авторы благодарны Российскому фонду фундаментальных исследований за финансовую помощь при публикации данных исследований (грант № 02-01-14148), ректору ТРТУ В. Захаревичу, коллегам и сотрудникам МГТУ и ТРТУ за творческую поддержку и обсуждение материалов монографии. Большое влияние на проводимые авторами исследования оказали тесные творческие связи авторов с коллегами из Российской ассоциации искусственного интеллекта Э. Поповым и В. Тарасовым.

Все критические замечания, предложения, пожелания и рекомендации, которые могут быть в перспективе использованы для развития эволюционного моделирования и смежных дисциплин, будут с благодарностью приняты авторами (E-mail: evv@rk9.bmstu.ru, vkur@tsure.ru, kur@tsure.ru).

Авторы

Введение

Одна из основных проблем в науке и технике сегодня — это разработка теории, математических методов и моделей для эффективного принятия решения в сложных системах. Сложность систем, их уникальность, отсутствие адекватного математического аппарата для создания моделей приводит к необходимости поиска новых направлений моделирования и синтеза. Эти направления в настоящее время активно разрабатываются и составляют основу понятия «искусственный интеллект». Именно на пути применения перспективных интеллектуальных технологий большинство исследователей надеется найти решение стоящих перед учеными и инженерами проблем и задач. При этом важнейшими задачами являются оптимизация, построение интеллектуальных искусственных систем поддержки принятия решений; моделирование процессов эволюционного развития природы; адаптация и взаимодействие искусственных систем с внешней средой; исследование открытых систем; построение порядка из хаоса; поддержание гомеостаза в искусственных системах; использование иерархической самоорганизации.

Основным языком интеллектуальных искусственных систем (ИС) является математический аппарат системного анализа, теория сложных систем, теория графов и множеств, математическое моделирование, имитация. Способность формировать математические структуры и оперировать ими — краеугольный камень человеческого интеллекта. Математические модели и алгоритмы — это те формы, в которых теория эволюционного моделирования и искусственного интеллекта описывают знания и способы работы с ними.

Эволюционное моделирование (ЭМ) представляет собой одно из быстро развивающихся направлений математического моделирования, объединяющее компьютерные методы моделирования эволюционных процессов в естественных (ЕС) и искусственных системах, такие как генетические алгоритмы, эволюционные стратегии, эволюционное программирование и другие эвристические методы. Главная трудность построения вычислительных систем, основанных на принципах эволюции живой природы и применении этих систем в прикладных задачах, состоит в том, что природные системы достаточно хаотичны, а действия исследователей носят направленный характер. Компьютер используется как инструмент для решения определенных задач, которые пользователь формулирует, акцентируя внимание на максимально быстром решении при минимальных затратах. Природные системы не имеют никаких таких целей или ограничений, во всяком случае, они не очевидны. Однако биологические системы обладают свойствами воспроизводства, адаптации, самонисправления, устойчивости, гибкости и многими другими, которые лишь фрагментарно присутствуют в искусственных системах.

Одним из видов интеллектуальной деятельности человека является реализация нечетких алгоритмов лицом, принимающим решение. Главным помощником при этом стали интеллектуальные искусственные системы поддержки и принятия решений.

Создание интеллектуальных искусственных систем, переход от алгоритмов обработки данных к альтернативной технологии автоформализации профессиональных знаний, моделирование эволюции, построение гомеостатических, синергетических и эволюционных моделей принятия решений позволяют увеличить эффективность деятельности инженеров, конструкторов, технологов, менеджеров. Все это связано с новой концепцией развития естественных и искусственных систем, основанных на самоорганизации, построении порядка из хаоса, интеллектуальном моделировании.

В новом развивающемся научном направлении «эволюционное моделирование» исследуется совместное действие многих интеллектуальных искусственных систем, в результате которого возникают новые знания и новые структуры. Философское определение интеллектуальной искусственной системы, как известно, включает в себя понятие целостности, которое совмещает свойства открытости и закрытости. Для снятия данного противоречия и выхода к синтезу систем в последнее время предлагается триада, обладающая системными свойствами и оказывающаяся простейшей структурной ячейкой интеллектуальной искусственной системы. Комплексы из трех равноправных объектов, находящихся в заданных отношениях, существовали давно: цель—план—стратегия; точность—локальность—простота; индукция—дедукция—абдукция; изменчивость—наследственность—отбор и др. В триаде пара элементов находится в отношении дополнительности, а третий элемент задает отношение (меру) совместимости. Общей для разных интеллектуальных объектов характеристикой является структура или архитектура интеллектуальной ИС. Это совокупность отношений, т. е. набор знаков или чисел, в который вкладывается определенный смысл. Она характеризуется многоуровневостью, единством и непротиворечивостью.

Новая интеллектуальная ИС создается, а точнее, самоорганизуется, путем эволюции. Тогда решение ИС некоторой задачи соответствует построению определенной математической модели, обладающей нужными свойствами, которых не было сначала. Смысл «существования» таких интеллектуальных ИС сводится к порождению себе подобных систем, которые при благоприятных обстоятельствах дают новое «потомство» до тех пор, пока возникающая популяция альтернативных интеллектуальных ИС существует.

Основное отличие эволюции ИС от эволюции ЕС состоит в наличии у ИС цели, связанной с оптимальным решением определенного класса задач. Здесь важна разработка критериев искусственного отбора, т. е. способов селекции и репродукции (воспроизведения), при которых кроме бесперспективных элементов из ИС удаляются и малоперспективные.

Реализация эволюционных преобразований аналогична решению оптимизационных задач. Для таких задач необходимо каждый раз доказывать сходимость процесса к искомому решению. Один эволюционный процесс отличается от другого видом объектов, способом размножения, числом

объектов в популяции, числом поколений и критериями выживания. При математической интерпретации основная идея решения оптимизационной задачи методами ЭМ состоит в следующем. Производится перебор объектов по всей популяции, в результате чего возникают объекты двух поколений — родители и потомки. При этом метод эволюции превращается в алгоритм смены поколений, в котором потомок становится родителем только в следующей генерации. При этом проведение аналогий с ЕС представляется эффективным для решения оптимизационных задач.

Процесс создания интеллектуальных ИС имеет множество путей. Аналогично множество альтернативных путей развития имеет и эволюция. Отсюда разнообразие возможных форм ИС, но, как правило, реализуется лишь одна из них. Эволюция подтверждает, что в основе ЕС и ИС заложены модели и аналогии принципа взаимодействия противоположностей ИНЬ—ЯН. Их взаимодействие является причиной изменения и развития природы ЕС. ИНЬ — символизирует неопределенность и неоднозначность, случайные движения, а ЯН — завершенность, реализацию цели и построение целого.

В процессе эволюции интеллектуальной ИС прошлое не исчезает. Надо научиться находить и определять в эволюционных структурах информацию о прежних состояниях и процессах. Вся природа устроена так, что в ней действуют принципы экономии и ускорения эволюции. Для этого природа выработала специальные механизмы воздействий. Необходимо научиться правильно моделировать и распределять эти воздействия. Часто важна не величина управляющего воздействия, а его структура, пространственная конфигурация, топология, симметрия или отсутствие таковой. При этом возможна самоорганизация объектов на различных иерархических уровнях. Эволюция в интеллектуальной ИС невозможна без смены одной устойчивой структуры другой, без конкуренции, без конфликтов, которые разрешаются в ее процессе путем борьбы и компромиссов, установления гармонии и гомеостаза. В этих случаях трудоемкость оптимизационных задач резко возрастает и становится невозможным использование алгоритмов полного перебора с экспоненциальной временной сложностью из-за необходимости обработки огромных массивов информации. Поэтому требуется проведение модернизации структуры как самих традиционных интеллектуальных ИС, так и основных блоков, стратегий, концепций, алгоритмов и методов решения оптимизационных задач. Одним из таких подходов является использование методов эволюционного моделирования, генетических алгоритмов (ГА), синергетических принципов управления, построения порядка из хаоса и адаптации.

ЭМ — хорошо известная и эффективная оптимизационная методология, которую применяют для решения различных оптимизационных задач в науке и технике. Она основана на аналогии с естественными процессами селекции и генетическими преобразованиями, протекающими в природе. В быстро прогрессирующей теории ЭМ содержится анализ класса адаптивных, управляющих, интеллектуальных ИС, в которых структурные модификации осуществляются последовательными и параллельными преобразованиями.

При попытках эффективного решения задач проектирования сложных интеллектуальных ИС постоянно возникает конфликт между сложностью систем и требованиями увеличения качества проектирования, конструирования и технологичности производственных процессов. Данные проблемы не могут быть полностью решены распараллеливанием процесса проектирования, увеличением персонала конструкторов и качеством рабочих станций. Одним из возможных подходов к решению этой проблемы является использование ЭМ. Исследования в этой области интенсивно ведутся многими исследователями.

Эволюция сложных систем происходит таким образом, что в ее процессе увеличивается степень неопределенности и неоднородности системы в целом. В системах неопределенность проявляется в двух видах — внутренней и внешней. Внутренняя связана с характером поведения отдельных подсистем и элементов, их стохастическим характером, сложным динамическим поведением, незнанием исследователем свойств элементов системы и закономерностей их развития и функционирования. Внешняя неопределенность — это неопределенность взаимодействия с внешней по отношению к системе средой. Воздействие внешней среды имеет, как правило, случайный, часто непредсказуемый характер. В этих условиях сложная система представляет собой конечное множество относительно самостоятельных и постоянно взаимодействующих подсистем. Их поведение часто не удается описать математически и остается практически единственный аппарат для построения моделей как отдельных подсистем, так и их совокупности — имитационное моделирование. Итак, имитационное моделирование — тот единственный универсальный метод, который приходится применять при решении различных задач, возникающих при анализе и синтезе сложных систем.

В связи с указанными выше причинами рассмотрение вопросов использования имитации в эволюционном моделировании при решении оптимизационных задач представляет большой интерес, когда необходимо решать задачи, применительно к реальным сложным системам. Имитация не может использоваться сама по себе, имитационный эксперимент позволяет оценить лишь одну точку в области поиска, в то время как определение решения должно осуществляться на всем пространстве. Поэтому имитационное моделирование при решении указанных задач используется в совокупности с другими методами — математическими, статистическими, эвристическими и т. д. Такое совместное использование различных подходов к решению некоторой задачи принятия решения приводит к появлению и развитию гибридных систем. В гибридных системах сочетание различных подходов позволяет получить новое качество при решении сложных задач.

В монографии проведен анализ теории и практики ЭМ. Рассмотрены пути развития ЕС и интеллектуальных ИС на основе моделей различных эволюций. Установлена связь между эволюцией и синергетикой, при этом рассмотрены вопросы определения порядка из хаоса в моделях ИС. Описаны эволюционные методы и ГА. Приведены новые архитектуры генетического поиска и инструментальные средства ГА. Рассмотрены гибридные

системы, включающие комбинированные методы эволюционного и имитационного моделирования, построение многоагентных и многомодельных систем. Описаны новые перспективные технологии генетического поиска для решения оптимизационных задач, позволяющие частично решать проблемы предварительной сходимости алгоритмов. Авторы рассматривают такую новую область ЭМ, как искусственная жизнь, а также решение оптимизационных задач с применением подходов, разрабатываемых в этой области.

Монография структурирована следующим образом.

В начале книги рассматриваются различные подходы к эволюционному развитию природы, как к первоисточнику, из которого исследователи заимствуют стратегии, методы и подходы, разрабатываемые и применяемые в эволюционном моделировании. Авторы не дискутируют с противниками эволюционной теории естественных систем, так как многочисленные примеры эффективной работы методов эволюционного моделирования при решении задач науки и техники являются достаточно серьезной причиной к их дальнейшему исследованию и развитию. Из многочисленных теорий эволюционного развития авторы выделяют и рассматривают следующие, с их точки зрения наиболее продуктивные для стоящих перед ними целей, пять моделей (видов) эволюции:

- **модель эволюции Ч. Дарвина** — процесс, посредством которого особи некоторой популяции, имеющие более высокое функциональное значение (с сильными признаками), получают большую возможность для воспроизведения потомков, чем «слабые» особи. Такой механизм часто называют методом «выживания сильнейших»;
- **ламаркизм** или **модель эволюции Ж. Ламарка**. Им предложена теория, основанная на предположении, что характеристики, приобретенные особью (организмом) в течение жизни, наследуются его потомками. В отличие от простого ГА данная модель оказывается наиболее эффективной, когда популяция имеет тенденцию сходимости в область локального оптимума;
- **сальтационизм (модель эволюции де Фриза)**. В основе этой модели лежит моделирование социальных и географических катастроф, приводящих к резкому изменению видов и популяций. Эволюция, таким образом, представляет собой последовательность скачков в развитии популяции без предварительного накопления количественных изменений в эволюционных процессах;
- **модель К. Поппера**, который рассматривал эволюцию как развивающуюся иерархическую систему гибких механизмов управления, в которых мутация интерпретируется как метод случайных проб и ошибок, а отбор — как один из способов управления с помощью устранения ошибок при взаимодействии с внешней средой;
- **синтетическая теория эволюции**, описанная Н. Дубининым (интеграция различных эволюций, в том числе Ч. Дарвина, Ж. Ламарка и де Фриза). Ее кардинальным положением является признание стохастичности процессов мутации и больших резервов рекомбинационной

изменчивости. Условия внешней среды — не только факторы исключения неприспособленных особей из популяции, но и формирующие особенности самой синтетической теории эволюции.

В завершении первой главы представлены примеры сложных искусственных систем (производственных и информационных), которые не были специально задуманы и спроектированы как таковые. Их современные формы — это результат эволюционного увеличения сложности, приобретения и накопления новых качеств и свойств. В результате эволюции эти системы адаптируются к изменению внешней среды, демонстрируют интеллектуальные свойства и, как следствие, повышают эффективность своего функционирования.

Одним из современных направлений науки, занимающейся перспективными вопросами, связанными с организационными структурами и управлением, является синергетика. Эволюционные и синергетические подходы (глава 2) не могут быть разделены, когда мы рассматриваем задачи анализа и проектирования сложных систем. Они тесно взаимосвязаны с системными исследованиями в области искусственного интеллекта. Рассмотрена новая идеология исследований в области ИИ на основе системного подхода и многоагентно-ориентированной парадигмы, включая исследования генезиса и эволюции интеллектуального поведения, которая приводит к возникновению гибридного (синергетического) ИИ.

Синергетические процессы позволяют биологическим системам трансформировать энергию, предварительно преобразованную на молекулярном уровне, в ее макроформы. В этой связи можно считать, что эволюция — это синергетический процесс образования все новых и новых макроструктур (т. е. новых видов). В природе и ИС имеется иерархия, означающая, что система состоит из очень большого числа подчиненных подсистем. На основе анализа основных принципов синергетики сделан вывод о том, что она занимается изучением временной эволюции систем. В открытых системах, обменивающихся с внешней средой энергией, веществом, информацией, возникают процессы самоорганизации, т. е. процессы рождения из хаоса некоторых устойчивых упорядоченных структур с новыми свойствами систем.

Далее во второй главе анализируются принципы гомеостатики, изучающей механизмы поддержания динамического постоянства неизменно важных параметров, функций, ритмов и этапов эволюционного развития. Рассмотрены четыре составляющие гомеостатики: внутренние противоречия; иерархическая организация; иерархия гомеостатов; реализация в управлении принципа регулируемого противоречия. Использование этих составляющих позволяет моделировать условия, необходимые для устойчивого функционирования системы, содержащей неустойчивые компоненты.

Сложные ИС имеют фрактальную структуру, они строятся из универсальных строительных блоков, которые повторяются в различных масштабах. Такие системы инвариантны к анализируемому объекту, способны к самоподобному размножению на различных пространственно-временных

уровнях и передаче информации о нарушении устойчивости структурного состояния; обладают свойствами адаптации к внешнему воздействию. В монографии рассмотрен механизм агрегации, описывающий создание фракталов. Согласно ему определенная разновидность фракталов может быть получена в процессе неупорядоченного роста.

В ЭМ особо выделяют генетические оптимизационные алгоритмы как наиболее хорошо исследованный и находящий широкое применение класс методов, сочетающих в себе элементы случайного поиска и эвристических подходов. Эвристики представляют собой некоторую формализацию правил, по которым человек принимает решения и которые являются обобщением его опыта и интуиции. Поэтому теория ГА не является завершенной, она постоянно совершенствуется и развивается. Тем не менее, достигнутые на этом направлении успехи привлекают все большее число исследователей и практиков к разрабатываемым здесь методам. В связи с этим в третьей главе книги рассмотрены основы теории ГА. Началом возникновения ГА считается модель биологической эволюции и методы случайного поиска. ГА — это не просто случайный поиск. Они эффективно используют информацию, накопленную в процессе эволюции. ГА позволяют абстрактно и формально объяснить адаптацию процессов в ЕС и ИС, спроектировать ИС, дают много преимуществ при решении реальных задач. В третьей главе подробно рассмотрен простой ГА, на базе которого создаются все другие типы этих алгоритмов. Сформулирована теория ЭМ, приведены основные правила и аксиомы. Приведена модифицированная теорема ГА, показывающая асимптотическое число схем, выживающих в следующих поколениях.

В главе четыре рассмотрены инструментальные средства ЭМ, в которых используется ряд методов одномерного градиентного поиска, а также статистических методов оптимизации. В рамках одномерного поиска описаны следующие методы: пассивный; последовательный; дихотомии; Фибоначчи; золотого сечения; поиск в глубину, в ширину, с возвращением, с использованием И-ИЛИ деревьев и метод горизонта. Последовательный поиск выполняется путем перебора значений целевой функции (ЦФ) для нахождения оптимального значения. Метод дихотомии реализуется за счет механизма обычного перебора возможных точек разрыва. Он аналогичен методу деления отрезка пополам для нахождения точки, в которой ЦФ имеет локальный оптимум. Для определения глобального оптимума ЦФ в оптимизационных задачах в главе предлагается:

- использовать комбинированные методы одномерного и градиентного поиска и статистических методов оптимизации;
- учитывать знания о решаемых задачах, позволяющие связать возможные значения ЦФ с известными значениями в точках реализованных испытаний;
- изменять начальную точку спуска методом проб и ошибок, градиентными методами, совместными методами оптимизации и локального поиска;

- анализировать использование различных архитектур эволюции и статистические методы поиска с адаптацией, позволяющие находить несколько областей локальных экстремумов.
- использовать совместные схемы поиска с ГА.

Рассмотренные методы могут выполняться не только на основе случайного поиска, но и с использованием эвристик. Основными стратегиями взаимодействия поиска и ЭМ служат стратегии «поиск–эволюция», «эволюция–поиск», «поиск–эволюция–поиск», «эволюция–поиск–эволюция» и другие. Элементы поиска в указанных стратегиях можно наращивать иерархически в зависимости от наличия вычислительных ресурсов и времени, заданного для получения окончательного решения.

Вопросам построения гибридных систем, включающих подсистемы эволюционного моделирования, блоки оптимизации, взаимодействующие с имитационными моделями, экспертными системами и другими системами поддержки принятия решений, посвящена пятая глава. Создание имитационных моделей выступает здесь как одно из направлений развития подходов интеллектуального имитационного моделирования. В данной главе описаны подходы и модели многоагентных систем, различного уровня интеллектуальности и их дальнейшая эволюционная форма — модели искусственной жизни. Как примеры гибридных систем с эволюцией приводятся моделирование развития популяции простейших автоматов и многомодельные системы.

В шестой главе исследуются оптимизационные задачи на графах, такие как задачи разбиения, размещения, раскраски вершин графов, нахождения пути коммивояжера, построения деревьев Штейнера, трассировки соединений, построения клик, независимых подмножеств, покрывающих деревьев, определения планарности и изоморфизма графов. Для решения указанных задач применяются различные методы ЭМ. Рассмотрены нечеткие модифицированные алгоритмы их решения. Описаны итерационные методы парных и групповых перестановок, методы последовательного приближения, релаксации, поиска в глубину и ширину, направленного перебора и другие, а также эвристики, использующие различные методы статистической оптимизации. Эти эвристики представлены методами отжига, генетического поиска и их модификациями. Применяется группировка сильно связанных вершин в кластеры с дальнейшим сбором этих кластеров в заданные части разбиения, а также использование фрактальных множеств для группирования сильно связанных вершин. Рассмотрены последовательный ГА разбиения, алгоритмы разбиения графов на части на основе модифицированной агрегации фракталов. Проанализировано использование итерационного разбиения гиперграфа и ГА дихотомического разбиения графа. При решении проблем разбиения графов на части рассмотрены задачи группирования элементов, обладающих одинаковыми свойствами.

При решении задач, связанных с размещением вершин графа на плоскости, применены классический ГА и принципы метагенетической оптимизации. При размещении вершин графа на плоскости в решетке заданной конфи-

гурации выделены два подхода применения ЭМ. Первый подход использует методологию последовательного размещения элементов, одинаковых по высоте и различных по ширине, а второй — построение минимальных и квазимиимальных кластеров и агрегацию фракталов. Описан эффективный ГА размещения вершин графа в линейке с минимизацией суммарной длины соединений.

При построении покрывающих (остовных) деревьев графа с дополнительными вершинами (деревьев Штейнера) предложена упрощенная эвристическая процедура, основанная на горизонтальных и вертикальных «столбах» Штейнера. На основе этой процедуры описан ГА построения прямоугольных деревьев Штейнера.

Рассмотрена постановка задачи и описаны различные алгоритмы проектирования БИС и печатных плат — трассировки соединений. Основное внимание уделено задаче двухслойной канальной трассировки. Рассмотрены модифицированные ГА двухслойной канальной трассировки для цепей стандартной и различной ширины.

Проанализированы алгоритмы решения задачи о коммивояжере. Разработана стратегия решения таких задач на основе ГА и жадных эвристик.

Рассмотрена задача раскраски графа и описан гибридный ГА ее решения. Основной стратегией для задач раскраски графа являются последовательные и жадные эвристики, которые дают с первой попытки результаты с локальным оптимумом.

Описаны четыре стратегии построения независимых подмножеств и клик графа. Первая стратегия основана на построении и анализе строительных блоков для создания независимых подмножеств или выделения клик графа. Вторая стратегия использует алгоритмы поиска в ширину. Третья стратегия предусматривает последовательный анализ всех вершин в глубину до получения семейства независимых подмножеств или клик графа. Четвертая стратегия является комбинацией первых трех.

Приведены основные критерии определения планарности графов. Описаны эвристические методы определения планарности и плоской укладки графов. Проанализирован генетический подход для решения данной задачи. Описан метод кодирования альтернативных решений и приведены операторы кроссинговера, мутации и инверсии, ориентированные на знания об исследуемых графах. Для непланарных графов рассмотрен ГА определения максимально планарной части графа и минимизации пересечений ребер. Временная сложность алгоритма близка к линейной зависимости.

Рассмотрены алгоритмы распознавания изоморфизма графов. Основная идея заключается в разбиении исследуемых графов на предполагаемо изоморфные подграфы. После получения подмножеств разбиения в подмножествах наибольшей мощности выполняются модифицированные генетические операторы. Наилучшие результаты показывают новые генетические операторы, основанные на жадной стратегии, методе дихотомии, минимального кластера, методе золотого сечения и методе Фибоначчи. Они позволяют параллельно анализировать все подмножества вершин и снижают время нахождения результата.

Методы исследования рассмотренных в шестой главе задач и их результаты описаны в главе семь. Здесь приведены количественные и качественные оценки ГА при проведении различных серий испытаний. Приведены экспериментальные исследования комплексов эволюционных алгоритмов. Выполнено некоторое сравнение решений, полученных с использованием ЭМ со стандартными решениями.

В восьмой главе рассмотрены примеры решения практических задач на основе методов ЭМ. Среди них рассмотрены примеры решения задач плоского раскроя материалов, одномерной и двумерной упаковки блоков, определения пути коммивояжера, трассировки сверхбольших интегральных схем. Применение разработанных триединых концепций принятия решений позволяет получать решения указанных задач с локальными оптимумами за полиномиальное время. Далее в главе приводятся результаты исследований, связанные с применением гибридных систем на основе ЭМ и имитационного моделирования для решения задач плоского и двумерного раскроя, динамического планирования поставок продукции на склад, краткосрочного планирования работы производственного участка.

Можно считать, что целью монографии является исследование развития нового научного направления — ЭМ и эффективности его применения для решения оптимизационных задач.

ЭВОЛЮЦИЯ ЕСТЕСТВЕННЫХ И ИСКУССТВЕННЫХ СИСТЕМ

Все сущее во вселенной подчинено
одним и тем же вечным, неизменным
законам эволюции.
А. Клизовский

1.1. Эволюция Дарвина

Эволюция (лат. *evolutio* — развертывание, развитие), как принято считать в биологии, это необратимое историческое развитие естественных и искусственных систем [1]. Обычно эволюцию противопоставляли революции — быстрым и значительным по масштабу изменениям. В настоящее время стало ясно, что процесс развития ЕС и ИС складывается из изменений как постепенных, так и резких, как быстрых, так и длящихся много поколений.

Основные характерные черты биологической эволюции это: преемственность; возникновение в эволюционном процессе целесообразности; усложнение и совершенствование структур.

Согласно второму началу термодинамики все совершающиеся в природе процессы направлены в сторону разрушения структур, снижению уровня сложности, увеличения доли беспорядка (энтропии) во всех системах. А в процессе эволюции происходит лишь местное усложнение системы, которое достигается ценой лишней затраты энергии на развитие организма [2, 3].

Впервые термин эволюция был использован в биологии швейцарским ученым Ш. Бонне в 1782 году. Под эволюцией понимают медленные постепенные количественные и качественные изменения объекта. При этом каждое новое состояние объекта должно иметь по сравнению с предыдущим более высокий уровень развития и организации.

Теорию эволюции — науку о закономерностях и причинах эволюционного процесса — называют эволюционным учением. В настоящее время существует большое число вариантов различных концепций эволюции. Основное их различие в том, какую изменчивость они берут за основу эволюции — определенную направленную приспособительную или же неопределенную ненаправленную и оказывающуюся приспособительной только случайно.

В биологии эволюция определяется наследственной изменчивостью, борьбой за существование, естественным и искусственным отбором [4, 5]. Эволюция приводит к формированию адаптаций (приспособлений) организмов к условиям их существования, изменению генетического состава от популяции видов, а также отмиранию неприспособленных видов. Под адаптацией понимается процесс приспособления строения и функций орга-

низмов и их органов к условиям окружающей среды. В науке под адаптацией понимают процесс накопления и использования информации в системе, направленный на достижение ее (системы) оптимального состояния, при первоначальной неопределенности и изменяющихся внешних условиях [5, 6]. О том, что явление адаптации имеется в живой природе, было известно биологам прошлых веков. В настоящее время генетика или теория генетики утверждает [6, 7], что адаптация не является какой-то внутренней сущностью, заранее приданной организму, но она всегда возникает и развивается. Такое развитие осуществляется под воздействием четырех основных признаков: наследственности, изменчивости, естественного отбора, искусственного отбора.

С развитием теории эволюции ее идеи все больше используются при моделировании мышления и поведения человека, создании современных компьютеров и т. д. В связи с этим возникают целые новые отрасли знаний и науки. Например, бионика (греч. *bion* — элемент жизни) — наука пограничная между биологией, генетикой и техникой, решающая инженерно-технические задачи на основе генетики и анализа структуры жизнедеятельности организмов [8]. Генетика — наука о законах наследственности и изменчивости организмов. Основная задача генетики — это разработка методов управления наследственностью и наследственной изменчивостью для получения нужных форм организмов или для управления их индивидуальным развитием [6, 7].

Известно, что Г. Мендель в 1866 г. сделал величайшее открытие, сформулировав три закона наследственности:

- **Закон однородности и реципрокности** (взаимность, эквивалентность). Первое гибридное поколение оказывается полностью однородным.
- **Закон расщепления**. Он относится ко второму поколению и касается расщепления признаков в потомстве в отношении 1:2:1.
- **Закон независимой комбинации**. Относится к потомкам родителей, отличающихся более чем одной парой признаков, и говорит о том, что признаки наследуются независимо друг от друга.

Таким образом, он описал общий закон природы и вывел рациональные формулы расщепления признаков в гибридном потомстве. Г. Мендель пришел к выводу, что наследственность прерывиста (дискретна), что наследуется не большая совокупность свойств, а отдельные признаки. Он предположил, что в половых клетках есть какие-то материальные структуры (позже их назвали генами), ответственные за формирование признаков. Одна из основных заслуг Г. Менделя — его гипотеза о материальных задатках, которые в двойном комплекте находятся в клетках и которые зародыш получает от обоих родителей [6, 7].

Важную роль в явлениях наследственности играют хромосомы — нитевидные структуры, находящиеся в клеточном ядре. Каждый вид характеризуется вполне определенным числом хромосом, во всех случаях оно четное, и их можно распределить попарно. При делении половых клеток и в процессе оплодотворения каждая хромосома ищет себе подобную. Они сближаются и располагаются параллельно друг другу и почти сливаются

ся. Затем они расходятся. Но во время контакта почти все хромосомы обмениваются своими частями. Получившаяся клетка (зигота), многократно делясь, образует зародыш, из которого развивается организм.

В клетках существует сложный и важный механизм перераспределения генетического материала. Каждая клетка организма имеет одинаковое число хромосом. Потомки имеют то же самое число, причем ровно половина от отца, а половина от матери. При таком обмене передаются наследственные признаки.

Ч. Вильсон в 1900 г. определил, что гены находятся в хромосомах. Две гомологичные хромосомы (одна от отца, а другая от матери) сближаются при созревании половых клеток и обмениваются частями. Это явление назвали кроссинговер. Оно происходит между разными генами случайным образом с разной частотой. Модель хромосомы в настоящее время — это нить, на которую, словно бусины, нанизаны гены [6].

Ч. Дарвин — первый ученый, который определил в живой природе существование общего принципа — естественного отбора. В эволюционном учении различают две стороны: учение о материале для эволюции и учение о ее факторах, ее движущей силе. Движущая сила эволюции — естественный отбор. В основе наследования лежат неделимые и несмешиваемые факторы — гены. Имелю через отбор и происходит направленное влияние условий жизни на наследственную изменчивость. Сама по себе наследственная изменчивость случайна. Под воздействием окружающей среды отбираются признаки, которые лучше других соответствуют условиям жизни.

Суть эволюции состоит в реализации целенаправленного процесса размножения — исчезновения, при котором размножению соответствует появление новых объектов, а гибели — удаление объектов из процесса в соответствии с определенным критерием естественного отбора (или селекции). Считается, что элементарной единицей эволюции являются популяции, и в ее основе лежат изменения наследственных структур, называемых мутациями. Причем только отбор превращает мутационные сдвиги и изменения в адаптацию. Следовательно, отбор является одним из ведущих факторов адаптивной организации ЕС и ИС. Адаптация позволяет накапливать и использовать информацию в ИС, достигать ее оптимального состояния, при первоначальной неопределенности и изменяющихся внешних условиях [4–7].

Наследственные изменения отдельных генов де Фриз назвал мутациями. Мутации и служат элементарным материалом для эволюционного процесса. Они закономерно возникают в природных условиях.

Изменчивость — разнообразие признаков и свойств у особей и групп особей любой степени родства. Различают изменчивость направленную и ненаправленную.

Направленная, или определенная, изменчивость обычно массовая и приспособительная. В данном случае наследуется не изменение признака, а способность к изменению, но в разной степени. Такие изменения называют модификациями. Определенная изменчивость — продукт эволюции, способность к ней возникает в результате отбора в течении многих поко-

лений. Но само изменение признака под влиянием какого-нибудь фактора внешней среды исчезает с гибелью организма, потомки должны обрести его заново. Только в этом смысле определенная изменчивость ненаследственна. Ее нельзя называть изменчивостью: это наследственность, проявляющаяся в фенотипе не всегда, а лишь при воздействии определенного фактора внешней среды.

Ненаправленная, или неопределенная, изменчивость возникает независимо от природы вызвавшего ее фактора, причем изменяющийся признак может изменяться и в сторону усиления, и в сторону ослабления. При этом она не массовая, а единичная. Различают два типа неопределенной изменчивости — комбинативную и мутационную. На основе комбинативной изменчивости при образовании потомства во время мейоза возникают новые сочетания материнских и отцовских хромосом. При этом хромосомы иногда обмениваются частями (кроссинговер), так что число комбинаций генов в каждом новом поколении резко возрастает. Мутационная изменчивость — процесс изменения генетической структуры организма, его генотипа. При этом изменяется число хромосом, или их строение, или же структура слагающих хромосому генов. Как и комбинативная изменчивость, мутационная — процесс ненаправленный (признаки могут при ней изменяться случайным образом), немассовый (одновременное возникновение какой-нибудь одной мутации у целого ряда особей в популяции невозможно) и неприспособительный (мутации могут и повышать и понижать жизнеспособность их носителей). Неопределенная изменчивость — материал для процесса эволюции. Изменения организмов, по Ч. Дарвину, определяются факторами внешней среды. При этом с большей вероятностью выживают и оставляют потомство носители полезных в данной среде признаков, возникших в результате мутации или рекомбинации определяющих эти признаки генов.

Понятия ген и генетика ввел датский ученый В. Иогансен. Приведем факторы, которые меняют генетический состав природной популяции: мутационный процесс, изоляция, «волны жизни», отбор.

Особое положение генов состоит в их уникальности. В хромосомном наборе каждый ген представлен только один раз. В 30-е годы было доказано, что генная мутация — это небольшое химическое изменение. Следовательно, ген имеет химическую природу, являясь молекулой или частью большой молекулы.

В 1953 г. с работ М. Уоткина и Ф. Крика началась новая наука — молекулярная генетика [10]. Биохимические особенности живых организмов наследуются по законам, которые открыл Г. Мендель. В генах «записаны» планы строения белков — планы всех наследственных признаков. Генетический код оказался общим для всех естественных систем на нашей планете. Он практически расшифрован. Каждая хромосома уникальна морфологически и генетически и не может быть заменена другой либо восстановлена при утере. При потере хромосомы клетка, как правило, погибает. Каждый биологический вид имеет определенное, постоянное число хромосом. В процессах наследования признаков определяющую роль играет поведе-

ние хромосом при делении клеток. Существует два основных типа деления клеток: митоз и мейоз. Митоз — прямое деление клеток тела, это механизм точного распределения хромосом между двумя образующимися дочерними клетками. Мейоз — механизм редукции (уменьшения числа хромосом вдвое) [6].

Классическая генетика к началу 1940-х годов пришла к пониманию дискретности таких качеств, как наследственность и изменчивость. Это стало возможным в первую очередь благодаря формированию теории гена в работах школы Т. Моргана. Основные положения этой теории можно сформулировать следующим образом [6, 10]:

- все признаки организмов находятся под контролем генов;
- гены — элементарные единицы наследственной информации, они находятся в хромосомах;
- гены могут изменяться — мутировать;
- мутации отдельных генов приводят к изменению отдельных элементарных признаков, или фенотипов.

Сейчас считают, что ген — реально существующая независимая комбинирующаяся и расщепляющаяся при скрещиваниях единица наследственности, самостоятельно наследующийся наследственный фактор. Ген определяют как структурную единицу наследственной информации, неделимую в функциональном отношении. Его рассматривают как участок молекулы ДНК, кодирующий синтез одной макромолекулы или выполняющий какую-либо другую элементарную функцию. Совокупность генов составляет генотип. Фенотип — совокупность всех внешних и внутренних признаков. Комплекс генов, содержащихся в наборе хромосом одного организма, обозначают геном.

Генетическая рекомбинация подразумевает несколько типов перераспределения наследственных факторов [5–7, 10]:

- рекомбинация хромосомных и нехромосомных генов;
- рекомбинация целых негомологичных (неоднородных) хромосом;
- рекомбинация участков хромосом, представленных непрерывными молекулами ДНК.

Основой регулярной (общей) рекомбинации является кроссинговер, т. е. обмен гомологичными участками в различных точках гомологичных хромосом, приводящий к появлению нового сочетания сцепленных генов.

Расщепление при независимом наследовании и при кроссинговере определяет изменчивость организмов вследствие комбинаторики существующих генов (аллелей). Аллелями называют определенное химическое состояние гена. Мутации — это возникновение качественно новых генов (аллелей), хромосом и наборов хромосом. Сочетание обоих типов изменчивости вызывает общую изменчивость генотипа.

Генетический материал обладает такими универсальными свойствами, как дискретность, непрерывность, линейность и относительная стабильность, выявляемыми в ходе генетического анализа. Повышение разрешающей способности генетического анализа возможно с помощью изучения

большого числа особей, применения селективных методов, ускорения мутационного процесса.

Все эти методы имеют важное значение в ЕС и могут найти применение в построении ИС. Увеличение числа особей приводит к росту разнообразия генетического материала, что означает увеличение исходного набора контролируемых генами функций, а это, в свою очередь, позволяет провести более широкий отбор функций. Ускорение мутационного процесса ведет к получению все более разнообразного генетического материала.

В ЕС и ИС роль мутаций заключается в том, что именно они генерируют новые функции, затем происходит дупликация, закрепляющая обе функции, а после этого начинается отдельная эволюция исходной и новой функции. Эта эволюция и показывает, что новая или возникшая в результате мутации функция обладает более высокими адаптационными качествами либо прежняя функция выполняет эту роль лучше.

В общем же эволюция стремится к усреднению (так как происходит все более однородное смешение разного по качеству с нашей точки зрения генетического материала). Поэтому в качестве одного из методов для получения наилучших результатов развития используется селекция. Селекция представляет собой форму искусственного отбора. Селекция, как наука, создана Ч. Дарвином, который выделял три формы отбора [1]:

- естественный отбор, вызывающий изменения, связанные с приспособлением популяции к новым условиям;
- бессознательный отбор, при котором в популяции сохраняются лучшие экземпляры;
- методический отбор, при котором проводится целенаправленное изменение популяции в сторону установленного идеала.

Движущими силами эволюции по Ч. Дарвину являются [1, 4]:

- неопределенная изменчивость, т. е. наследственно обусловленное разнообразие организмов каждой популяции;
- борьба за существование, в ходе которой устраняются от размножения менее приспособленные организмы;
- естественный отбор — выживание более приспособленных особей, в результате которого накапливаются и суммируются полезные наследственные изменения и возникают новые адаптации.

Дарвинизм адаптацию объясняет эволюцией. В результате естественного отбора вновь возникающие мутации комбинируются генами уже прошедших отбор особей, их фенотипическое выражение меняется и на их основе возникают новые адаптации. Следовательно, отбор — основной фактор эволюции, обуславливающий возникновение новых адаптаций, преобразование организмов и видообразование. Отбор проявляется в трех основных формах [4–7]:

- движущий (ведущий). Он приводит к выработке новых адаптаций;
- стабилизирующий. Он обеспечивает сохранение в неизменных условиях среды уже сформировавшихся адаптаций;

- дизруптивный (разрывающий). Он обуславливает возникновение полиморфизма при разнонаправленных изменениях среды обитания популяции.

Отбор идет по общей приспособленности организма, а не по какому-нибудь отдельному признаку.

При движущем отборе большую вероятность оставить потомство имеют особи, изменившиеся по каким-нибудь признакам по сравнению со средней для данного вида величиной (нормой). При этом отбирается один тип отклонений от нормы.

Стабилизирующий отбор сохраняет в популяции среднее значение признаков (норму) и не пропускает в следующее поколение наиболее отклонившихся от этой нормы особей. Это делает сохранения видов неизменными.

При дизруптивном, или разрывающем, отборе отбирается не один тип отклонений от нормы, а два или больше. Это путь дробления предкового вида на дочерние группировки, каждая из которых может стать новым видом.

Эволюционные процессы, протекающие внутри вида и завершающиеся видообразованием, называют микроэволюцией. Макроэволюцией называется развитие групп организмов надвидового ранга. Задачами макроэволюции являются анализ соотношения индивидуального и исторического развития организмов, анализ закономерностей направления эволюционного процесса. Естественный отбор приводит к эволюции процессов онтогенеза — взаимозависимостей развивающихся органов, названных И. Шмальгаузенom онтогенетическими корреляциями [5].

Шмальгаузен разработал и привел концепции целостности организма в индивидуальном и историческом развитии. Он исследовал механизмы эволюционного процесса и индивидуального развития организмов как саморегулирующихся систем, и изложил эволюционную теорию с позиций кибернетики.

В учении об отборе Ч. Дарвин доказал, что главной движущей силой эволюции является отбор наилучших форм, требующий для успеха таких условий: правильный выбор исходного материала, точная постановка цели, проведение селекции в достаточно широких масштабах и возможно более жесткая браковка материала, отбор по одному основному признаку [1].

Вид — основная структурная единица в ЕС, качественный этап их эволюции. Новые виды возникают в результате межвидового скрещивания. Вид — это группа популяций, особи которых могут скрещиваться в естественных условиях, но изолированы от других видов. Вид подразделяется на несколько популяций, каждая из которых эволюционирует самостоятельно. Процесс перехода одного вида в другой не скачкообразен и генетическая изоляция между ними может не возникать.

Основой эволюционных процессов в ЕС служит популяция. Популяция — это многочисленная совокупность особей определенного вида, в течение длительного времени (большого числа поколений) населяющих определенный участок географического пространства, внутри которого осуществля-

ется та или иная степень случайного свободного скрещивания. В популяции нет абсолютно тождественных особей. Каждая особь является носителем уникального генотипа, который управляет формированием фенотипа. Существование каждой особи ограничено некоторым временным интервалом, по истечении которого особь погибает. При этом генотип особи исключается из генофонда популяции, но при жизни особь может передать наследственную информацию.

Известны три механизма передачи наследственной информации при рождении потомства: бесполое размножение, половое размножение, промежуточная (между бесполом и половым) форма размножения [6, 7]. Устойчивая передача генов от родителей к потомкам зависит в первую очередь от способности молекул ДНК к репродукции и авторепродукции. Каждая особь в течение жизни подвергается воздействиям внешней среды. В некоторых случаях эти воздействия могут привести к перестройкам молекул ДНК, переносящих наследственную информацию. Изменение первоначальной последовательности генов в молекулах ДНК приводит к изменению свойств этой молекулы, а следовательно, и наследственной информации.

Рассмотренные выше факторы по степени влияния на эволюцию можно упорядочить по убыванию влияния:

- естественный отбор,
- изоляция популяции,
- колебание численности популяции,
- мутационные процессы.

Пусковой механизм эволюции функционирует в результате совместного действия эволюционных факторов в пределах популяции. В результате действия эволюционных сил в каждой популяции многократно возникают элементарные эволюционные изменения. Со временем некоторые из них суммируются и ведут к возникновению новых приспособлений, что и лежит в основе видообразования.

Направленная молекулярная эволюция подобна искусственному отбору. Если надо создать молекулу, обладающую каким-либо химическим свойством, следует выбирать из большой популяции молекул те, которые в наилучшей степени выражают это свойство, и произвести из них дочерние, в разной степени похожие на родителей. Этот процесс отбора и дупликации повторяется до тех пор, пока не будет достигнут нужный результат [10].

Согласно теории Ч. Дарвина, эволюция осуществляется во взаимодействии трех повторяющихся вновь процессов: отбора, амплификации, мутации. Амплификация — процесс производства потомков или более точно, копирование особей, действует в природе совместно с отбором [1]. Критерий отбора подобен библейскому: «плодитесь и размножайтесь». Стратегия повторяющейся рандомизации может заставить молекулы эволюционизировать в направлении улучшения функциональных характеристик.

Приведем математическую модель процесса эволюции, описанную Г. Дульневым [11]. Рассмотрим ИС, в которой изменение во времени t

некоторого параметра $\dot{q} = dq/d\tau$ пропорционально величине этого параметра. В качестве q фигурирует число хромосом в популяции. Простейшее эволюционное уравнение имеет вид

$$\frac{dq}{d\tau} = \alpha q,$$

где α — параметр, определяющий как скорость, так и характер изменения процесса эволюции.

Решение данного уравнения имеет вид:

$$q = q_0 e^{\alpha\tau},$$

где q_0 — постоянная интегрирования, равная значению параметра q в начальный момент времени $\tau = 0$, α может быть больше или равно 0.

Для ИС будет характерно свойство стохастичности. Для них вводят член $f(\tau)$, учитывающий флуктуации во времени:

$$\frac{dq}{d\tau} = \alpha q + f(\tau).$$

Например, пусть рост популяции $\frac{dP}{d\tau}$ пропорционален числу скрещиваний между хромосомами p_1, p_2 ($p_1, p_2 \in P$). Тогда

$$\frac{dP}{d\tau} = \alpha p_1 p_2,$$

$0 < \alpha < 1$, ($|P| = |p_1| + |p_2|$). Если примем $|p_1| \equiv |p_2| \equiv P/2$, то можно получить [6]

$$n(\tau) = \frac{4}{\alpha} \times \frac{1}{(\tau_f - \tau)},$$

где $n(\tau)$ — размер популяции P в момент времени τ . Если $\tau = \tau_f$ (режим обострения), тогда $|P| = \infty$, т. е. популяция будет содержать бесконечное число хромосом. Следовательно, существует спектр путей развития, по которым может пойти эволюция Ч. Дарвина. Возможный путь развития здесь определяет случайность. ИС может сама себя организовать, но нужна случайность как спусковой механизм.

Итак, подведем краткие итоги. Ч. Дарвин впервые дал строго научную теорию эволюции. Она состоит из следующих положений [1, 4]:

- в ЕС все подвержено неопределенной наследственной изменчивости, потомство производится, отличающееся по многим признакам. В настоящее время считается, что изменчивость — результат мутаций, т. е. изменения в строении генов ДНК;

- все организмы в ЕС размножаются таким образом, что число появляющегося на свет потомства превосходит число взрослых особей, но численность всех организмов в среднем остается более или менее постоянной, она колеблется около средней величины.
- основой отбора является принцип «выживают сильнейшие».

Неопределенная наследственная изменчивость поставляет материал для естественного отбора, который осуществляют условия внешней среды. В этом принципиальное отличие дарвинизма от всех предшествующих эволюционных теорий. Ч. Дарвин принципом естественного отбора ненаправленных (случайных) наследственных изменений объяснил и приспособленность всех организмов к условиям внешней среды, и расхождение признаков в процессе эволюции, и усложнение, совершенствование организации (прогрессивную эволюцию) [1, 4–7, 12].

В настоящее время основой современного эволюционного учения является результат синтеза дарвинизма, генетики и молекулярной биологии. Принцип эволюции — это главный биологический фактор, объединяющий все организмы в историческую цепь событий. Каждый объект в этой цепи является результатом серии «случайностей», которые происходили под влиянием селективных факторов окружающей среды. В течение многих поколений случайные изменения и естественный отбор придавали определенные очертания поведению элементов, чтобы как можно лучше приспособиться к изменяющейся окружающей среде. Такие изменения могут быть достаточно неожиданными, подтверждающими, что эволюция созидательна. Эволюция не имеет строго определенной внутренней цели, влияющей на популяцию индивидов и на каждый индивид в отдельности. Тем не менее, она создает решения проблемы выживания, уникальные для каждого индивида.

Идеи применения знаний о живой природе для решения задач в ИС принадлежат еще Леонардо да Винчи, который спроектировал и пытался построить летательный аппарат с движущимися крыльями как у птиц. Наблюдения в области адаптации живых организмов приводили, приводят, и будут приводить к идеям, позволяющим наделить указанными свойствами ИС [8, 11, 13–15].

Моделирование эволюции может предоставить алгоритмические средства для решения комплексных задач науки и техники (с использованием хаотических возмущений, вероятностного подхода, нелинейной динамики), которые нельзя было решить традиционными методами. В общих чертах, эволюция может быть описана как многоступенчатый итерационный процесс, состоящий из случайных изменений и последующей затем селекции. Таким образом, достаточно просто обнаружить взаимосвязь между таким определением эволюции и оптимизационными алгоритмами [8, 16–19].

Отметим, что в последнее время проявляется тенденция использования естественных аналогов при создании моделей, технологий, методик, алгоритмов для решения тех или иных задач проектирования, конструирования и производства, стоящих перед человечеством. В большинстве случаев

использование естественных аналогов дает положительные результаты. Как правило, это объясняется тем, что аналог, взятый из природы, совершенствовался в течение многих лет эволюции и имеет на данный момент самую совершенную в своем роде структуру.

1.2. Эволюции Ламарка, де Фриза, Попшера и синтетическая теория эволюции

Первая группа концепций и гипотез учения об эволюции связывается с именем Ж. Ламарка. В 1809 г. он предположил, что все живые организмы целесообразно приспосабливаются к условиям среды. Ж. Ламарк предложил свою концепцию эволюции до теории эволюции Ч. Дарвина о естественном отборе. Его теория основана на предположении, что характеристики, приобретенные организмом в течение жизни, наследуются потомками. Он предложил ее как средство, при помощи которого организм передает специальные черты для выживания в среде, и это положение известно как теория эволюции Ж. Ламарка или ламаркизм [20, 21].

Причинами эволюции Ж. Ламарк считал стремление всех живых организмов к прогрессу, развитию от простого к сложному (учение о градации (уровень, ступень)), а также целесообразные изменения организмов, направленные на приспособление к внешним условиям. Эти изменения, как утверждал Ж. Ламарк, вызываются прямым влиянием внешней среды, упражнением органов и наследованием приобретенных при жизни признаков. Его считают основоположником телеологической (греч. *teleos* — результат, цель и *logos* — наука) теории эволюции, согласно которой способность организмов приспособляться к окружающей среде есть врожденное свойство жизни.

Ж. Ламарк объясняет одну из особенностей эволюции органического мира приспособляемостью. Прогрессивную эволюцию, появление форм, более сложных и совершенных, он объяснял «законом градаций» — стремлением живых существ усложнять свою структуру. Приспособительные изменения, возникнув один раз, далее, по мнению Ж. Ламарка, способны передаваться по наследству (концепция «наследования благоприобретенных признаков»). Его концепция не дает четкого объяснения эволюции. Согласно ей виды эволюционируют, приспособляясь и усложняясь, потому что у них существуют свойства — приспособляться и усложняться. Причины направленных изменений объясняются различно, но их можно свести к двум:

- направленное влияние внешней среды;
- способность самого организма.

Такие гипотезы называют телеологическими. Телеологический взгляд на протекающие в природе процессы высказал Аристотель. Согласно Аристотелю, причина развития — будущая цель [22]. Так и в эволюции по Ж. Ламарку — большая приспособленность потомков возникает в результате целенаправленной изменчивости предков. Согласно ламаркизму предполагается, что живые организмы способны сами находить верное реше-

ние, как себя улучшить, и, более того, сами же способны свое решение осуществлять [20]. Зная, как сложно устроена ЕС, легко понять, что это затруднительно. Ламаркизм не объясняет большинство эволюционных преобразований.

Ж. Ламарк попытался создать целостную эволюционную теорию. Согласно ей, переход от низших форм жизни к высшим — градация. Она происходит в результате имманентного и всеобщего стремления ЕС к совершенству. Ламаркизм эволюцию объясняет адаптацией. Разнообразие видов на каждом уровне организации объясняется модифицирующим градацию воздействием условий среды. Ж. Ламарком выведены два закона:

1. Упражнение органов приводит к их прогрессивному развитию. Неупражнение органов приводит к редукции.
2. Результаты упражнений и неупражнений органов при достаточной продолжительности воздействия закрепляются в наследственности организмов и далее передаются из поколения в поколение вне зависимости от вызвавших их воздействий среды.

Направленная изменчивость в эволюции Ж. Ламарка не причина, а всегда результат эволюционного процесса. Способность к ней — такое же приспособление, возникающее на протяжении многих поколений. Поэтому направленная изменчивость не может быть доказательством правоты ламаркизма. В биологии считается, что его законы основаны на ошибочном представлении о том, что природе свойственно стремление к совершенствованию и наследованию организмом приобретенных свойств.

Эволюция Ж. Ламарка является мощной концепцией искусственной эволюции, применимой в науке и технике, несмотря на то, что она дискредитирована в биологии. Авторы предлагают использовать некоторые принципы этого типа эволюции для построения интеллектуальных ИС и решения оптимизационных задач. Подобно эволюции в ЕС, интеллектуальные ИС используют простейшие преобразования генотипов и фенотипов и перестановка фенотипов с соответствующими генотипами является часто возможной. В случае, когда генотипы являются своими собственными фенотипами, не требуется преобразований между ними. Это позволяет оптимизировать фенотип для решения конкретных проблем среды. Подобная оптимизация отражается в соответствующем генотипе для последующего наследования потомками. Следовательно, эволюция Ж. Ламарка в ЕС устанавливает связь между генотипом и фенотипом [17, 23–25]. Ламаркизм обычно дает локализованный поиск форм в пространстве структур фенотипов. Локализованный поиск при решении сложных практических задач большой размерности более подходит для исследования локальных областей популяции в отличие от глобального перебора.

Ян Лотси, У. Бетси, де Фриз, В. Гольдшмидт, В. Кордюм в своих работах [7, 12, 26] отклоняли интегрирующую роль естественного отбора, а в процессах появления мутаций видели непосредственно факторы видообразования. Они и развили идеи сальтационизма, по которым основным путем происхождения видов служат внезапные скачки без предваритель-

ного накопления количественных изменений в процессах эволюции. Такой механизм эволюции иногда называют эволюцией катастроф. Он проявляется ориентировочно один раз в нескольких тысячах поколений. Основная идея его состоит во внесении глобальных изменений в генофонд на момент катастрофы. Будем называть такую эволюцию — эволюцией де Фриза. Появление видов путем внезапных скачков — это реальный процесс.

Мутационный процесс, обладая собственными закономерностями, поставляет сырой материал для эволюционного процесса в виде новых, мутационно возникающих генов. Эти гены вступают в состав генофонда вида, который сохраняет их вне зависимости от их эволюционного значения в пределах, дозволяемых естественным отбором. Согласно закону Гарди (равновесие популяции) и закону Пирсона (стабилизирующее скрещивание), если на популяцию не действуют внешние факторы, то закономерности популяции поддерживают наследственное строение в неизменном состоянии [4, 6]. На рис. 1.1 приведена одна из возможных схем механизма эволюционного процесса.

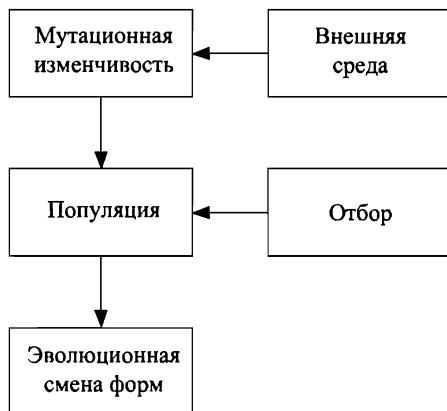


Рис. 1.1. Механизм эволюционного процесса

Мутационный процесс поставляет материал для эволюции, обладает своими собственными закономерностями, но не определяет закономерности эволюционного процесса.

Внешняя среда в конечном итоге определяет эволюцию, но не в простой форме связи между наследственной изменчивостью организмов и средой как в эволюции Ж. Ламарка, а в более сложной форме. Согласно Н. Дубинину [7] модернизированная схема механизма эволюционного процесса с учетом эволюции де Фриза имеет вид (рис. 1.2).

Эволюция Дарвина на основе естественного отбора на каждой из своих ступеней создает все новые условия генотипической среды. Естественный отбор закрепляет только те генотипы, которые дают приспособленный соответствующий данной внешней среде фенотип. Отбор — это, по форме, селектирующий фактор, а, по существу, — творческий процесс создания ЕС или ИС. Дарвин говорил лишь о категории неопределенной изменчи-

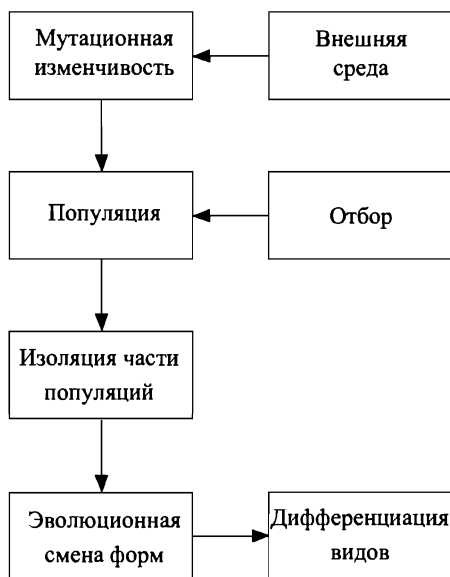


Рис. 1.2. Модернизированный механизм эволюции Ламарка

ности. В настоящее время ее подразделяют на мутационную и комбинационную или комбинативную. С момента разделения изменчивости на две категории возникают предпосылки для создания новой, синтетической, теории эволюции. Она является синтезом дарвинизма, классической генетики и теории популяций. Суть ее в следующем: вновь образующиеся изменения генов, а также появляющиеся при этом в результате скрещивания комбинации генов подвергаются отбору под воздействием факторов внешней среды. Синтетическая теория эволюции объединила признанные положения эволюций Ч. Дарвина, Ж. Ламарка, де Фриза и учение о генетике популяций. Основные положения синтетической теории эволюции согласно Н. Дубинину [7] следующие:

- эволюция невозможна без адаптации организмов к условиям внешней среды. Фактором, формирующим приспособленность строения и функции организмов, выступает естественный отбор, который использует случайные мутации и рекомбинации;
- естественный отбор, опираясь на процессы преобразования генетики популяций, создает сложные генетические системы. Их модификации закрепляются стабилизирующим отбором;
- в популяциях наследственная изменчивость имеет массовый характер. Появление специальных мутаций свойственно лишь отдельным особям;
- наиболее приспособленные особи оставляют большое количество потомков. Обратная связь от фенотипов к генотипам идет не на основе эволюции Ж. Ламарка путем адекватного унаследования благоприят-

ретенных признаков, а через преобразование генотипа по законам генетики популяций;

- специальные виды эволюций идут путем фиксации нейтральных мутаций на основе стохастического процесса;
- реальным полем эволюции являются интегрированные генетические системы. Только при наличии разрывов между видовыми генетическими системами могла возникнуть и развиваться эволюция;
- виды происходят путем эволюции популяций. Источником этой эволюции служат постоянная интеграция генотипов или появление макромутаций (внезапное видообразование). Среди макромутаций известны гениные мутации, хромосомные перестройки, перенос от одного вида к другому разными механизмами репродукции отдельных генов или их блоков;
- самодвижение представляет собой внутреннее необходимое самопроизвольное изменение системы, определяемое ее противоречиями;
- противоречия между случайным характером наследственной изменчивости и требованиями отбора определяет уникальность видовых генетических систем и видовых фенотипов.

Полная модифицированная схема синтетической теории эволюции показана на рис. 1.3.

Процессы, происходящие при реализации синтетической теории эволюции по Н. Дубинину, включают:

- случайные колебания численности отдельных популяций;
- периодические колебания численности популяций;
- скачкообразные колебания численности популяций;
- колебания численности популяций, обусловленные процессом миграции.

Отметим, что признание единства факторов эволюции в виде наследственности, изменчивости и естественного отбора не исключает существование разных форм эволюций. И. Дубинин выделяет четыре основные формы осуществления внутренне единого эволюционного процесса:

- микроэволюция (процессы внутривидовой эволюции);
- фаза нарастающего эволюционного усовершенствования;
- переломные моменты эволюции;
- эволюции основных особенностей в организации ЕС.

Эволюционный процесс связан с двумя типами адаптаций. Один тип адаптации основан на выработке приспособлений к тем условиям внешней среды, в которых вид существует в настоящее время. Другой тип адаптации связан с выработкой таких особенностей в структуре, которые должны обеспечить его соревнование с другими видами во времени.

Основная задача синтетической теории эволюции — определение природы противоречий или постепенной эволюции, т. е. разных форм противоречий между наследственностью и постоянно меняющимися потребностями в приспособлении. Итак, в ЕС можно выделить пять видов эволюции [7, 12, 26, 27]:

- синтетическая теория эволюции (модифицированные эволюции Ч. Дарвина, Ж. Ламарка, де Фриза);
- ламаркизм или эволюция Ж. Ламарка (адекватное унаследование влияния факторов среды);
- сальтационизм (эволюция де Фриза, сведение эволюции только к скачкам);
- номогенез (действие специальных внутренних факторов, обеспечивающих генетическое развитие);
- прогрессивная эволюция.

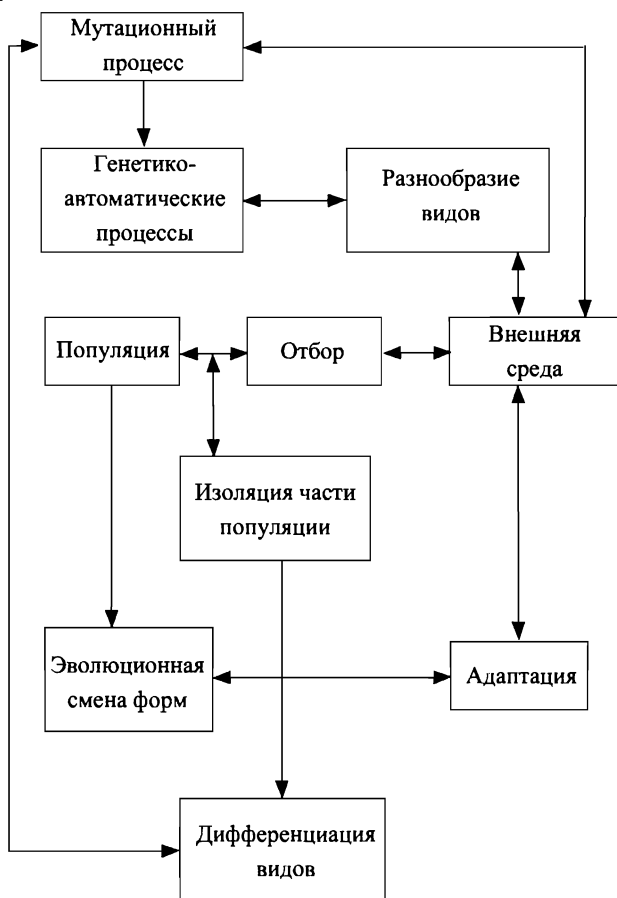


Рис. 1.3. Полная схема механизма эволюционного процесса

Эволюция Ч. Дарвина частично объясняет прогрессивную эволюцию. Сложные высокоорганизованные формы побеждают в жизненной борьбе более простые не потому, что они более сложные, а потому, что они лучше приспособлены к окружающим условиям. Там, где условия внешней среды

не дают особого преимущества сложным формам, они и не возникают. А при упрощении внешней среды наблюдается не прогресс, а, наоборот, регресс — движение назад, в сторону упрощения, потери ненужных для вида структур и признаков.

Кардинальное положение синтетической теории эволюции — признание стохастичности процессов мутаций и больших резервов рекомбинационной изменчивости. Условия внешней среды — не только факторы исключения неприспособленных особей, но и формирующие особенности самой синтетической теории эволюции. В этой связи в эволюции ЕС и ИС важное значение имеет объединение всех видов и форм эволюции в синтетической теории эволюции.

Кроме рассмотренных основных видов эволюций интерес представляет концепция К. Поппера. Им предложена эволюционная эпистемология [28], интерпретированная в виде триады: дедуктивизм—отбор—критическое устранение ошибок. Будем называть ее обоженной эволюцией К. Поппера; она излагается в виде двенадцати тезисов, основанные из которых следующие:

- все ИС и ЕС решают проблемы;
- проблемы всегда решаются методом проб и ошибок;
- устранение ошибок может осуществляться либо путем полного устранения неудачных форм, либо в виде эволюции механизмов управлений, осуществляющих модификацию или подавление неудачных форм поведения или гипотез;
- популяция использует тот механизм управления, который выработался в процессе эволюции ее вида;
- популяция представляет собой «головной отряд» эволюционного вида к которому она принадлежит. Она сама является пробным решением, анализируемым в процессе эволюции, выбирающим окружающую среду и преобразующим ее;
- эволюционная последовательность событий представляется в виде схемы $F_1 \rightarrow TS \rightarrow EE \rightarrow F_2$, где F_1 — исходная проблема, TS — пробные решения, EE — устранение ошибок, F_2 — новая проблема.

В отличие от эволюции Ч. Дарвина, где существует одна проблема — выживания сильнейших, в эволюции К. Поппера существуют и другие проблемы: воспроизводство, избавление от лишнего потомства и т. д. Здесь важное значение имеет отличие F_1 от F_2 , что как-то объясняет творческую интеллектуальную эволюцию.

Согласно [28], информация не входит в ИС и ЕС из окружающей среды, а ИС и ЕС исследуют окружающую среду и активно получают из нее информацию. Знания при этом — это любые формы приспособления или адаптации ИС или ЕС к условиям окружающей среды. Причем знание часто имеет характер ожидания. Всякое знание есть результат пробы и устранения ошибок — плохо приспособленных гипотез, адаптации, элементов.

Процесс выбора лучшей индивидуальности в эволюции К. Поппера может являться процессом отбора (селекции), а отбор из некоторого множества

случайных событий не обязан быть случайным. Он также предлагает триаду миров: мир физических вещей (модели эволюции), мир индивидуального мышления (знания, модели и оптимизационные задачи), мир объективного содержания мышления (популяция альтернативных решений).

В его концепции эволюция рассматривается как развивающаяся иерархическая система гибких механизмов управления, в которых мутация интерпретируется как метод случайных проб и ошибок, а отбор — как один из способов управления с помощью устранения ошибок.

Итак, в эволюции К. Поппера процесс слепой изменчивости и избирательного сохранения лежит в основе прироста знания, возрастания приспособленности к внешней среде. В этом процессе выделяются три основные части (триада): механизмы изменчивости; согласованные процессы отбора; механизмы сохранения или распространения отобранных вариантов. Этим механизмам присущи противоречия и между ними необходимо устанавливать гомеостаз. Процессы, позволяющие сокращать случайную изменчивость, сами являются результатом индукции, так как сохраняют и используют знания о внешней среде.

1.3. Эволюционная кибернетика

Кроме приведенных выше моделей и видов эволюций определенный интерес представляют математические модели, связанные с идеями эволюционной кибернетики, описанные в работе [48]. Рассмотрим возможные схемы таких эволюций. К ним относятся модели квазивидов, гиперциклов, сайзеров и нейтральной эволюции.

В модели квазивидов анализируется возможный процесс эволюционного возникновения простейших макромолекул, кодирующих наследственную информацию. Здесь рассматривается эволюция цепочек РНК. В результате формируется квазивид — распределение цепочек РНК в окрестности некоторой оптимальной РНК.

Отдельная хромосома задается своим геном p_i . Популяция представляет собой множество $\{p_i\}$, состоящее из хромосом, где N_p — численность популяции. Геном p_i — цепочка из L символов, $|p_i| = L$, то есть, L — длина хромосомы. Считается, что длина L и численность популяции велики, то есть, $L, N_p \gg 1$. Вводится ограничение: L и N_p не изменяются в процессе эволюции.

Алгоритм эволюции на основе такой модели имеет вид:

1. Формирование начальной популяции $P^0 = \{p_1^0, p_2^0, \dots, p_{N_p}^0\}$. Для каждой хромосомы $i = 1, 2, \dots, N_p$ и поколения $t = 1, 2, \dots, T$ популяции выбираем символ p_i^t , полагая его равным произвольному символу алфавита.
2. Отбор.
 - 2.1. Определение ЦФ. Она представляет собой величину, характеризующую приспособленность хромосомы к данной популяции — величину $f(p_i)$, где $i = 1, 2, \dots, N_p$.

- 2.2. Формирование нового поколения P^{t+1} рассматриваемой популяции. Отбор хромосом с вероятностями, пропорциональными $f(p_i^t)$.
- 2.3. Реализация оператора мутации. Для каждого $i = 1, 2, \dots, N_p$ заменяем p_i^{t+1} на произвольный символ алфавита с вероятностью $P_i(\text{ОМ})$.
3. Организация последовательности поколений. Повторяем пункты 2, 3 для $t = 1, 2, \dots$
4. Конец работы алгоритма.

Геном определяется приспособленностью модельных организмов $f(p_i)$, считается что $f(p_i) \gg 0$. Приспособленность произвольной хромосомы p_i определяется расстоянием по Хеммингу $\rho(p_i, p_m)$ между p_i и p_m (числом несовпадающих компонент в этих наборах), причем $f(p_i)$ экспоненциально уменьшается с ростом $\rho(p_i, p_m)$, т.е. чем больше различие между p_i и p_m , тем меньше ее приспособленность. Новое поколение P^{t+1} получается из старого P^t путем отбора и мутации последовательностей. Отбор представляет собой формирование нового поколения в соответствии с приспособленностью хромосом $f(p_i)$. Чем больше приспособленность хромосомы, тем больше у нее будет потомков в новом поколении. Начальная популяция $P^t (t = 0)$ формируется из N_p случайных хромосом, формирование новой популяции выполняется на основе модели рулетки.

Если предположить, что при переходе от поколения к поколению численность видов изменяется незначительно, то процесс эволюции можно считать непрерывным. Тогда динамику популяции описывают уравнением [48]:

$$\frac{\partial x_k}{\partial t} = f(p_k)x_k + \sum_l w_{kl} - Ex_k \quad (1.1)$$

где x_k — численность хромосом k -го вида, $f(p_k)$ — селективная ценность хромосом k -го вида, E — параметр, характеризующий общее разбавление популяции, причем $\sum_k x_k = N_p = \text{const}$, w_{kl} — параметры, характеризующие мутационные потоки.

Если ввести частоты $g_k = x_k/N_p$, характеризующие вероятности нахождения хромосом разных видов в популяции, то (1.1) примет вид:

$$\frac{\partial g_k}{\partial t} = f(p_k)g_k + \sum_l w_{kl}g_l - Eg_k, \quad (1.2)$$

причем $\sum_l g_l = 1$. Для случая предельно малых мутаций ($w_{kl} = 0$) из выражения (1.2) получим уравнение Эйгена Фишера [48]:

$$\frac{\partial g_k}{\partial t} = (f(p_k) - f(p)_{\text{cp}})g_k. \quad (1.3)$$

где $f(p)_{\text{cp}}$ — средняя селективная ценность популяции.

Из выражения (1.2) можно получить вариант теоремы естественного отбора Фишера [48]:

$$\frac{\partial f(p)_{\text{cp}}}{\partial t} = \sum_k g_k (f(p_k) - f(p)_{\text{cp}})^2. \quad (1.4)$$

Из (1.3) и (1.4) следует, что селективная ценность в популяции растет, достигая стационарного значения в состоянии равновесия системы. Скорость изменения средней селективной ценности равна ее дисперсии и обращается в нуль при достижении указанного состояния равновесия. Среднее значение расстояния до оптимальной последовательности составляет

$$\rho_{\text{ср}} = \frac{P_1(\text{OM})}{\beta}. \quad (1.5)$$

где β — параметр, характеризующий интенсивность отбора.

Исходное распределение последовательностей случайно. При $\rho \gg 1$ оно принимает форму нормального распределения со средним значением $\rho_{\text{ср}} = N_p/2$ и дисперсией $D = N_p/4$. Выражение (1.5) имеет место при малых интенсивностях мутаций и отбора $1 \gg \beta \gg P_1(\text{OM})$, где $P_1(\text{OM})$ — вероятность однократных мутаций. Таким образом, в результате эволюции отбирается не отдельный вид, а квазивид — распределение видов. Выражение (1.5) используют для оценки максимальной интенсивности мутаций, при которой эволюция определяет оптимальную с нашей точки зрения хромосому.

Модель квазивидов основана на существовании единственного максимума приспособленности $f(p_k)$.

Для многоэкстремальной функции приспособленности $f(p_k)$ используют модель спиновых стекол Шеррингтона–Кирпатрика [48]. Такая модель описывает систему попарно взаимодействующих спинов и сводится к следующему:

1. Имеется система P из n спинов. $P = \{p_1, p_2, \dots, p_n\}$, $n \gg 1$. Спины принимают значения 1 или -1 , то есть, $p_i \in \{1, -1\}$.
2. Взаимодействия между спинами случайны. Энергия спиновой системы определяется формулой: $E(P) = - \sum_{i < j} J_{i,j} p_i p_j$, где $i, j = 1, 2, \dots, n$; $J_{i,j}$ — элементы матрицы взаимодействий между спинами. Величины $J_{i,j}$ нормально распределены.

Модельный геном P состоит из последовательности элементов $p_i \in P$, которые случайно попарно взаимодействуют в соответствии с матрицей $J_{i,j}$. Максимальной энергией $E(P)$ будут обладать «организмы», имеющие такую комбинацию p_i , которая обеспечивает максимальную кооперативность взаимодействия между спинами p_i для данной матрицы $J_{i,j}$.

При селекции особи выбираются в новую популяцию с вероятностями, пропорционально их ЦФ. При мутациях происходят случайные повороты спинов ($p_i \rightarrow -p_i$) с вероятностью $P_i(\text{OM})$ для каждого спина любой последовательности. Основное отличие от модели квазивидов состоит в том, что параметр $P_c(\text{OM})$ (вероятность смены знака символа последовательности) отличается от $P_i(\text{OM})$: $P_c(\text{OM}) = 0,5 P_i(\text{OM})$.

Совместно с эволюционным процессом здесь рассматривается последовательный поиск минимумов энергии спинового стекла. Задается система из n спинов. Случайно выбирается спин p_i , поворачивается ($p_i \rightarrow -p_i$)

и анализируется увеличилась или уменьшилась энергия. При уменьшении энергии принимается новое значение спина, при увеличении — возврат.

Итак, на основе понятия спиновых стекол строится модель эволюции для хромосом, геномы которых состоят из множества случайно взаимодействующих между собой элементов. Эволюция рассматривается как процесс поиска таких комбинаций элементов, которая обеспечивает эффективную кооперацию элементов генома.

Модель гиперциклов М. Эйгена и П. Шустера [48] интерпретирует гипотетическую стадию эволюции, которая может следовать за квазивидами. Гиперцикл — это самовоспроизводящаяся макромолекулярная система, в которой РНК и ферменты кооперируются. Макромолекулы кооперативно обеспечивают трансляцию так, что информация, закодированная РНК-последовательностями, передается в структуру ферментов аналогично механизму трансляции в биологических клетках. Циклическая организация гиперцикла обеспечивает его структурную стабильность.

Модель сайзеров аналогична модели гиперциклов. Это — универсальная модель самовоспроизводящейся системы, подобной структуре самовоспроизводящихся автоматов Дж. фон Неймана [48]. Такие автоматы состоят: из ленты, хранящей информацию; из автомата А для изготовления произвольного другого автомата согласно информации ленты; из автомата В для копирования ленты; автомата С, координирующего процесс отделения нового автомата от автомата родителя. Модель адаптивного сайзера показывает процесс эволюционного возникновения простой системы управления.

В. Редько [48] рассматривает упрощенный процесс эволюции как цепочку: Квазивиды → Гиперциклы → Сайзеры → Протоклетки → Простейшие организмы. Очевидно, что описанные модели — это элементы и возможные сценарии предбиологических систем.

М. Кимура [48] предложил модель нейтральной эволюции с нейтральным отбором. Идея такого отбора основана на эволюционном алгоритме (игре) следующего вида:

1. Задана популяция больших и малых строительных блоков.
2. Эволюция представляет собой последовательность поколений. Каждое поколение в своем развитии проходит два этапа. На первом этапе дублируются все строительные блоки: большой блок имеет два больших потомка, маленький — имеет два потомка. На втором этапе из популяции случайным образом удаляется ровно половина строительных блоков с равной вероятностью для больших и малых.

Показано, что рассматриваемый процесс всегда сходится к одному из поглощающих состояний (все строительные блоки большие или все — маленькие). При $n > 1000$, где n — общее число элементов в популяции или исходное число элементов рассматриваемого объекта, характерное число поколений T_n , требуемое для сходимости к какому-либо из поглощающих состояний, равно $2n$. Данный эволюционный процесс чисто нейтральный, в результате эволюции выбирается только один вид.

1.4. Эволюция сложных систем

Согласно концепции общей теории систем сложность — это совокупность огромного числа различных объектов, функционирующих вместе и взаимодействующих непростым способом. Сложность есть взаимодействие, более того, взаимозависимость, т. е. поведение одного или нескольких элементов воздействует на поведение других элементов. Сложность зависит не только от взаимозависимости, но и от числа взаимодействующих компонентов.

Сложные системы можно анализировать, концентрируя внимание либо на объектах, либо на процессах. Для моделирования выгоднее рассматривать систему как упорядоченную совокупность объектов, которые в процессе взаимодействия друг с другом обеспечивают функционирование системы как единого целого. На системном уровне при решении ряда задач сложную систему можно рассматривать как дискретную, т. е. состоящую из отдельных элементов (подсистем), взаимодействующих между собой в определенные моменты времени.

Моделирование сложных крупномасштабных систем — более трудная задача, чем моделирование физических систем:

- имеется мало фундаментальных законов, относящихся к рассматриваемой системе;
- многие взаимосвязи элементов в системе с трудом поддаются количественному описанию и формализации;
- трудно количественно описать поведение входных элементов;
- важную роль играют стохастические процессы;
- неотъемлемой частью таких систем является процесс принятия решений (человеком, компьютерной программой).

Рассматривая сложные системы, Дж. Касти в первую очередь выделяет проблемы, связанные с внутренней структурой сложных систем [29]. Он подчеркивает, что не может быть единственной модели данной системы: существует множество моделей, каждая из которых обладает характерными математическими свойствами и пригодна для изучения определенного класса вопросов, связанных со структурой и функционированием системы.

Структурная связность является, по-видимому, наиболее существенной качественной характеристикой сложной системы. Она связана с двумя важными свойствами системы:

- математической структурой неприводимых компонентов (подсистем);
- способом, которым эти компоненты связаны между собой.

Отсюда следует, что сложность присуща самой системе, а тот факт, что она все же связана с взаимодействием исследователя и системы, отступает на второй план.

Помимо структурной, или статической сложности, включающей связность и структуру подсистем, существует динамическая сложность, обусловленная поведением системы во времени. Эти два вида сложности могут быть относительно независимы, иначе говоря, структурно простая система может быть динамически сложной, и наоборот.

Выводом из сказанного служит тот факт, что даже в элементарных системах могут возникать совершенно неожиданные (и неприятные) явления, если сложность взаимосвязей не изучена должным образом. Другой важный вывод состоит в том, что в отличие от обычных представлений такое парадоксальное поведение системы вызывается не наличием нелинейности, стохастичности и других подобных факторов, а порождается исключительно структурой системы, имеющимися связями и ограничениями, присущими компонентам системы.

Сложные системы являются управляемыми. Исследователь играет активную роль в функционировании сложной системы, ибо он имеет возможность генерировать определенные внешние воздействия на систему, пытаясь заставить вести ее необходимым для него (оптимальным) образом.

Помимо внешнего управления сложная система, как правило, включает в себя одну или несколько подсистем, выполняющих функции управления. Эта своя собственная система управления совсем не обязательно функционирует в полном соответствии с внешним управлением, что следует из свойств сложных систем.

Поэтому, говоря о сложности управляемых систем, следует рассматривать кроме структурной и динамической сложности еще и сложность управления.

Первые исследования, посвященные анализу разнообразия компонент и нормального функционирования сложной системы, были проведены У. Эшби [14]. Им сформулирован принцип необходимого многообразия: многообразие может быть разрушено только многообразием. Смысл этого принципа состоит в том, что *система может реализовать заданный тип поведения при воздействии внешних помех, лишь увеличив множество управляющих параметров*. Исследования систем различной природы и в различных аспектах показали, что жизнеспособность и возможность нормального функционирования системы имеют место при достаточной разнородности подсистем [30, 31]. Дальнейшее развитие этот принцип получил в работах по информатике, посвященных математическому описанию степени хаотичности и организованности системы [9].

Все сложные системы имеют тенденцию к развитию во времени, при этом во многих случаях сложная система оказывается результатом развития работающей простой системы. В результате взаимодействия отдельных частей системы возникают новые свойства, не присущие этим частям в отдельности. На эти процессы оказывает влияние и среда, которая не является статически целостной, а динамична, так как подвержена внешним и внутренним возмущениям.

Целесообразно определить некоторые общие свойства систем, которые свидетельствуют о сложности их динамики. В качестве таких общих свойств сложных динамических систем можно назвать следующие:

- эволюционность развития;
- неравновесность, проявляющаяся в постоянном обмене с внешней средой энергией и информацией;

- самоорганизация и самовоспроизведение;
- нарушение законов симметрии.

Первым серьезным вкладом в изучение эволюции, бесспорно, является теория Дарвина, согласно которой в результате флуктуаций (мутаций), происходящих под воздействием внешней среды и естественного отбора, производимого также внешней средой, происходит зарождение новых структур в живой природе. Это открытие Ч. Дарвин смог сделать, акцентируя внимание не на отдельной особи того или иного вида, а изучая их популяции. С тех пор появились другие выдающиеся работы, посвященные изучению эволюции в системах различной природы: П. Тейяра де Шардена в области палеонтологии и эволюции биосферы земли, Л. Гумилева в области этнологии и эволюции этносов, Д. Панина в области философии, С. Вайберга и А. Салама в области физики и эволюции материи и др. Несмотря на то, что указанные работы посвящены исследованию систем различной природы, в них выявляется общий механизм эволюции, уточняющий и углубляющий тот, который был описан Ч. Дарвином. В частности, установлено, что ведущая роль в эволюции принадлежит не внешней среде, а внутренним процессам системы. Как флуктуации, так и механизм отбора новых состояний обусловлены собственной динамикой системы, и в некоторых случаях — внешней средой. В этих работах определен важный принцип эволюции, заключающийся в увеличении информации, содержащейся в вышестоящих эволюционных структурах. Иными словами, *процесс эволюции есть процесс увеличения сложности*. Исследование неравновесных процессов в химических, биологических, технических, организационных и социальных системах также выявляет одинаковый механизм поддержания неравновесного состояния, связанный с необратимостью и нарушением временной симметрии.

Флуктуации в системах различной природы также осуществляются по аналогичным механизмам и играют сходную роль в процессах самоорганизации. Флуктуация случайно появляется как отклонение от среднего и имеет локальный масштаб. Вследствие диссипативных процессов флуктуация распространяется в пространстве и приводит к макроскопическим корреляциям, т. е. осуществляется согласованность и упорядочение отдельных компонентов системы. Механизмом отбора в случае появления нескольких устойчивых состояний является нарушение симметрии.

Становление сложной динамики, обуславливающей свойства сложных систем, можно определить в общих чертах следующим образом. Особенность сложных систем — появление новых ветвей решений в результате бифуркации, происходящей вследствие потери устойчивости стандартного состояния, вызванной нелинейностями. Таким образом, нелинейная система за счет бифуркации имеет возможность множественного выбора. Выбор одной из ветвей решения осуществляется с помощью флуктуаций, имеющих место в диссипативных системах. Возможность реализации только одной ветви решения и неосуществимость состояний, связанных с другой ветвью, приводит к пространственной или временной асимметрии. Нару-

шенная симметрия играет роль «устройства» отбора развивающихся структур [32].

Таким образом, несмотря на существование различных концепций сложных систем и отсутствие общепризнанного определения этого понятия, анализ свидетельствует о наличии у сложных систем разной природы общих характерных признаков. При этом феномен сложности во всех случаях связан с эволюцией системы.

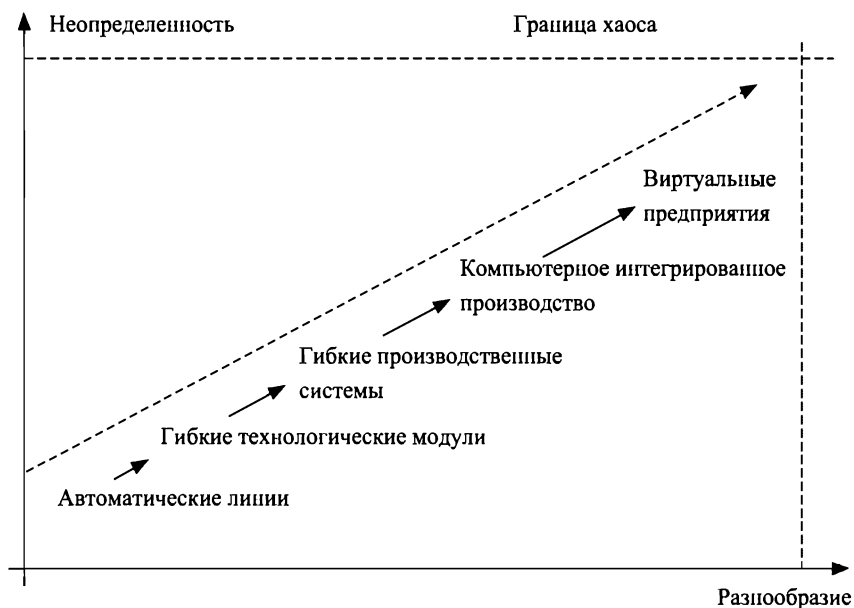


Рис. 1.4. Процесс эволюции производственных систем

Рассматривая, в качестве примера, развитие производственных систем как класса сложных систем в координатах «степень неопределенности» и «разнородность», можно получить картину эволюции этих систем от достаточно простых к очень сложным (рис. 1.4). При этом с увеличением сложности производственных систем увеличивается и их эффективность, что хорошо согласуется с принципами эволюции Ч. Дарвина. На рисунке приведены лишь часть этапов развития современных производственных систем. В левом нижнем углу расположены автоматические линии, достаточно однородные по составляющим их элементам и работающие в хорошо определенных условиях. Более сложными и более эффективными являются гибкие технологические модули, состав которых уже достаточно разнороден — станки, роботы, системы числового управления. Далее мы имеем гибкие производственные системы, в которые помимо основного оборудования входят складское, транспортное, системы планирования и управления, локальная сеть ЭВМ и т. п. Их функционирование имеет гораздо более стохастический характер, чем у предшествующих им систем и протекает

в условиях отсутствия полной и точной информации как о внешней среде, так и о процессах, протекающих в ней.

Таким образом, производство эволюционирует как сложная система. Следовательно, можно считать, что свойство эволюции является наиболее глобальным для сложных производственных систем, аккумулирующем в себе другие более частные свойства сложных систем, отмечаемые различными авторами.

Увеличивая сложность производственных систем, мы увеличиваем их эффективность за счет получения таких свойств системы, которыми не обладают ее составляющие части (эффект эмерджентности). Компьютерное интегрированное производство представляет собой еще более сложную систему, чем гибкие производственные системы, включая в свой состав системы проектирования, автоматизированной подготовки производства, локальную вычислительную сеть, систему информационной поддержки жизненного цикла изделий и многое другое. Дальнейшее развитие производства как организации ведет от компьютерного интегрированного производства к таким видам производства (организациям посттейлоровского типа) как виртуальные, расширенные, плоские, интеллектуальные, фрактальные и другие виды предприятий [33, 34].

Абстрактное описание сложной производственной системы и процессов в ней могут получаться с использованием различного математического аппарата. Однако по мере приближения к границе хаоса (рис. 1.4, верхний правый угол) сложность взаимодействия подсистем возрастает настолько, что лишь имитационное моделирование позволяет получать результаты, удовлетворительные с точки зрения решения задач анализа и управления.

Аналогичную картину мы будем наблюдать, если рассмотрим, например, развитие информационных систем (рис. 1.5).

Здесь, как и в случае производственных систем, можно наблюдать (рис. 1.5), как в процессе эволюции образуются все более сложные системы, которые никто и никогда целенаправленно не проектировал и не создавал. Так, не существует организации или какой-либо корпорации, которая поставила своей целью создание INTERNET и осуществила данный проект. «Всемирная сеть» появилась в процессе эволюции информационных систем. При этом следует отметить, что она функционирует на грани хаоса (в условиях большой неопределенности и непредсказуемости), но, тем не менее, миллионы пользователей ежеминутно успешно работают в данной системе. Она продолжает развиваться по некоторым законам, которые мы попытаемся далее рассмотреть в настоящей книге. Возникающие аналогии из биологии, использование терминов, принятых при описании биологических сообществ, не являются случайными. Между органическим миром, созданным природой, и миром, создаваемым человеком, существует много общего.

Законы эволюции являются общими для живого и неживого в природе. Это относится не только к сходству в структуре и функционировании, но также и к вопросам управления и принятия решений в сложных системах различной природы. Сложная система, возникшая эволюционным путем,

не может управляться каким-либо единым центром управления. Она требует распределенной системы управления, в которой существует большое число локальных подсистем управления, принимающих самостоятельные решения на основе знаний и механизмов логического вывода. Данные подсистемы образуют некоторое сообщество, в котором они объединяются общими целями и используют общее множество ограниченных ресурсов для достижения этих целей. Все это способствует возникновению и развитию многоагентных систем и интеллектуальных организаций.

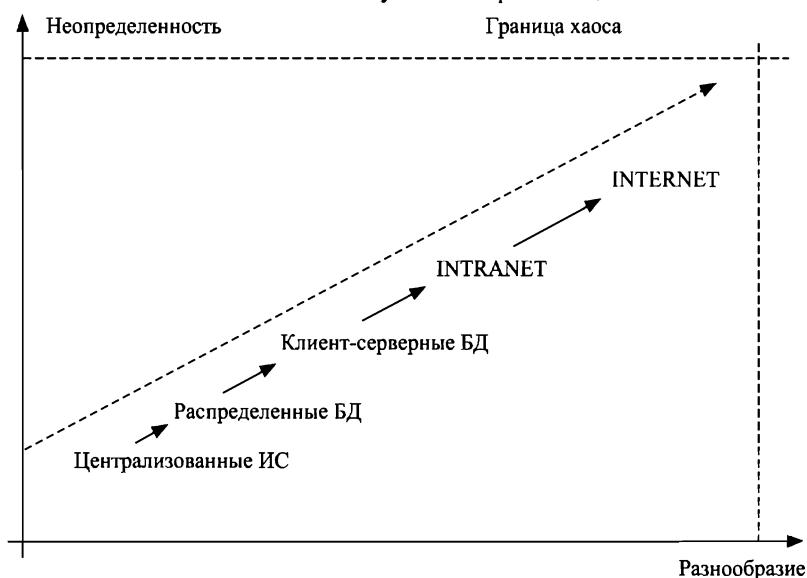


Рис. 1.5. Процесс эволюции информационных систем

С развитием человеческой цивилизации число таких систем постоянно растет. При этом возникает большое число общесистемных задач, связанных со спецификой сложных систем с эволюцией, управлением, обеспечением устойчивости их функционирования, повышением эффективности работы и т. д. [47].

1.5. Тейлоровские и посттейлоровские организации

Эволюция сложных производственных систем (организаций) ведет от компьютерного интегрированного производства к производству посттейлоровского типа. Здесь, как уже было отмечено выше, мы имеем дело с такими типами предприятий, как виртуальные, расширенные, плоские, интеллектуальные, фрактальные и другие. Следуя В. Тарасову, рассмотрим некоторые особенности и закономерности этого процесса [33, 34].

В теории Ф. Тейлора выделяют следующие понятия:

а) горизонтальной специализации (в пределе речь идет о разбиении общей задачи на максимально возможное число простейших подзадач и по-

ручении отдельному исполнителю выполнение одной-единственной подзадачи или функции);

б) вертикальной специализации — установления четких разграничений между постановкой и решением задачи или между сферами управления и исполнения, что выражается в том, что декомпозицию задачи и распределение обязанностей между исполнителями осуществляет руководитель; он же проводит обратную операцию композиции полученных решений и оценку результатов исполнителей.

Тейлоровская организация целесообразна в случае статической, стабильной внешней среды, хорошей разложимости исходной задачи на подзадачи, которые исполнители могут решать практически независимо друг от друга, установившейся номенклатуры и массового характера производимой продукции или услуг. Такая организация, будучи монолитной и замкнутой, состоит из однородных структур, включающих однотипных исполнителей, решающих довольно простые подзадачи, практически не взаимодействуя между собой. В ней доминируют вертикальные связи и отношения субординации, которые порождают реактивные стратегии организационного поведения, она развивается на основе строгого планирования и адаптации к одному виду сред.

В открытых динамически трансформируемых средах ранее неоспоримое достоинство тейлоровских организаций — устойчивая структура, инвариантная к внешней среде, оборачивается недостатком, поскольку жесткая, инерционная организация чаще всего не позволяет адекватно и своевременно реагировать на изменения внешней среды. Возрастание уровня неопределенности и динамика среды (рис. 1.4) делают малоэффективными монолитную структуру и жесткое централизованное управление в организациях. В этих условиях все большие и большие ресурсы управляющего органа затрачиваются на «внутренние нужды» организации и все меньше ресурсов остается на решение поставленной задачи, а также связанную с этим поисковую и адаптационную деятельность. Поэтому требуются новые подходы к организациям.

На смену тейлоровским организациям в процессе эволюции приходят посттейлоровские. Сравнительный анализ характеристик этих организаций приведен в табл. 1.1.

Таким образом, посттейлоровская организация — это открытая, сетевая, распределенная организация с прозрачными и довольно нечеткими границами, помещенная в сложную динамическую среду, которая состоит из автономных неоднородных организационных единиц различных типов совместно работающих для достижения некоторой цели.

Эволюция организации предполагает единство таких понятий, как:

- самоорганизация (в первую очередь, это аспект автономии);
- реорганизация (аспект преобразования, трансформации);
- экоорганизация (аспект открытости и наличия адаптивных взаимосвязей со средой).

В целом эволюция организаций происходит в направлении интеллектуальных организаций. Поведение интеллектуальной организации связы-

Таблица 1.1

Проектные критерии	Тейлоровские организации	Посттейлоровские организации
Внешняя среда	Статическая, стабильная	Динамическая, быстро меняющаяся
Организационная единица	Однородное функциональное подразделение	Автономная междисциплинарная рабочая группа (агентство)
Вид организации	Монолитная, замкнутая	Открытая, распределенная, сетевая
Взаимодействия	Вертикальные (субординация)	Горизонтальные (координация)
Структура	Иерархия (дерево)	Гетерархия (неоднородная сеть)
Связи	Постоянные	Гибкие, переменные
Управление	Централизованное	Децентрализованное
Задачи исполнителей	Простые (специализация)	Реинтегрированные (деспециализация)
Адаптация	Телогенез (адаптация к заданному состоянию)	Арогенез (расширение набора сред существования)
Развитие	Жесткое планирование	Самореорганизация

вается с наиболее эффективной и оперативной адаптацией к среде, что предполагает прогрессирующую обратимость организационных связей и приоритет кооперативных стратегий взаимодействия.

Интеллектуальные организации определяются процессами интенсивной циркуляции и переработки знаний, преобразованием интеллектуальных ресурсов в продукты и услуги, обеспечивающие их выживание и конкурентную способность. Их уровень интеллектуальности тесно связан со свойствами соответствующей сетевой структуры. Создание сетевых организаций представляет собой один из наиболее продуктивных методов стратегического управления. Центральная идея подобной организации состоит в организации системы различных взаимодействий между подсистемами, которые часто не являются постоянными и регулярными, а, скорее, вариabильны и зависят от ситуации взаимодействия. Поэтому сети образуются, развиваются и трансформируются в зависимости от потребностей подсистем и преследуемых ими целей.

Эффективность интеллектуальной организации также обусловлена построением достаточно полной и адекватной модели внешнего мира, причем организационное управление определяется не столько внутренними критериями, сколько внешними.

С другой стороны, интеллектуальность организации обеспечивает гибкое и в то же время устойчивое равновесие ее поведения путем постоянной структурной самореорганизации (обновления) в условиях сложной дина-

Таблица 1.2

Название организации	Критерии	Значение
Распределенная (горизонтальная)	• Структура организации и управления • Вид связей	• Максимально «плоская» и децентрализованная • Максимализация числа горизонтальных связей в сети
Ресурсосберегающая	Оптимальное управление ресурсами: • Стратегия обмена ресурсами «точно в срок» • Тотальное управление качеством • Параллельное выполнение	• Минимальные запасы • Максимальное качество работы для каждой подсистемы • Минимальное время цикла выполнения заказа
Расширенная	• Организация • Управление • Деятельность • Эволюция	• Открытая, развивающаяся сеть (максимум открытости) • Централизованное при наличии определенной децентрализации • Экстернализация (минимальная стоимость работы, максимальное приближение поставщика к потребителю) • Стратегия максимального сотрудничества и компромиссов
Фрактальная	• Автономия • Самовоспроизведение	• Деспециализация и максимальная независимость подсистем • Фрактализация
Холоническая (подвижная)	• Адаптация • Реакция • Структура	• Максимальное приспособление к среде • Максимально быстрая • Максимально гибкая
Обучающаяся	• Обучение • Самооценка • Прогнозирование	• Коллективное • Постоянная • Максимально долгосрочное
Виртуальная	• Организация • Интеграция • Коммуникация • Кооперация	• Открытая сеть (гетерархия) • Ресурсы, опыт • Максимально интенсивная телекоммуникация • Максимально скоординированное сотрудничество

мической среды. Чем богаче и разнообразнее формы влияния организации на среду, чем больше ассоциаций, альянсов или консорциумов при этом образуется, тем интеллектуальнее ее поведение.

В отличие от классических тейлоровских организаций, опирающихся на принцип «конкурентной рациональности», эволюция интеллектуальных организаций в большой степени определяется принципами коадаптации и коэволюции.

Итак, «коэффициент интеллектуальности» организаций зависит, главным образом, от таких факторов, как: 1) структура; 2) поведение; 3) эволюция; 4) степень адаптации к среде; 5) характер взаимодействий с клиентами; 6) истории и культуры организации; 7) распространения знаний.

С точки зрения эволюции сложных производственных систем (организаций) интересен сравнительный анализ базовых характеристик организаций (табл. 1.2) [34].

Понятие виртуальной организации (предприятия) вводится как обобщающее, позволяющее объединить и развить представление о естественных и искусственных посттейлоровских организациях. Они представляют собой сложные структуры, приводящие к синергетическим эффектам, которые способствуют возникновению у таких организаций новых функциональных возможностей.

Г Л А В А 2

ЭВОЛЮЦИЯ И СИНЕРГЕТИКА

Поступать в соответствии с природой
означает поступать спонтанно и в
соответствии со своей внутренней
сущностью, не нарушая тем самым
гармонии с окружающей средой и людьми.
Лао-цзы

2.1. Анализ и построение искусственных систем

В основе интеллектуальных ИС лежит понятие искусственного интеллекта. Проблемы ИИ тесно связаны с организацией знаний об окружающем мире в виде математических структур, таких как множества, графы, гиперграфы, фреймы, алгоритмы, которые отражают реальные связи и отношения между любыми объектами в природе (в частности, в некоторой предметной области).

К сожалению, пока не существует формального определения ИИ. В науке и технике используются адекватные определения и описания ИИ [35–45]. Например: ИИ — это новое научно-техническое направление, увеличивающее функциональные возможности технических систем и средств их проектирования; ИИ — системы, работающие со знаниями; ИИ — наука о концепциях в области информационной технологии; ИИ — средство решения интеллектуальных задач; ИИ — система, имитирующая некоторые стороны деятельности человека; ИИ — система, работающая с неформализованной и расплывчатой информацией; ИИ — это психическая способность к создаваемому нестереотипному поиску, построению адекватных форм мышления и целесообразных способов поведения и действия, основанных на опыте и знаниях субъекта и имеющих тенденцию к опережению событий; ИИ — способность автоматических систем брать на себя отдельные функции интеллекта человека. Термин интеллект происходит от латинского *intellectus* — что означает ум, рассудок, разум, мыслительные способности человека. ИИ — это синоним мышления, но, в отличие от него, определяющий качество этого процесса. Критериями качества считают эффективность, способность находить нестандартные решения, простоту в отношении когнитивной нагрузки.

Классические модели ИИ пропитаны рационализмом, поскольку в них интеллект связывается с рациональными методами решения задач с использованием эвристических процедур. Моделирование различных взаимодействующих агентов становится основным предметом ИИ в последнее время [42].

В ИИ выделяют три направления: логическое, конструкторское и эволюционное; три аспекта: управление неопределенностью, обучение, адаптация в процессе эволюции, а также три концепции в моделировании ИИ.

В первой концепции объектом исследования являются структура и механизмы работы мозга человека, а конечная цель — определение способов мышления. Вторая концепция в качестве объекта исследований рассматривает ИИ. Моделирование интеллектуальной деятельности реализуется с помощью ЭВМ. Третья концепция ориентирована на создание смешанных человеко-машинных или интерактивных ИС.

В настоящее время интенсивно развивается новая идеология исследований в области ИИ на основе системного подхода и многоагентно-ориентированной парадигмы. В [40–44] приведены основные стратегии анализа, синтеза и моделирования ИИ:

- исследования ИИ в иерархии взаимодействующих систем;
- исследования генотипа и эволюции интеллектуального поведения;
- исследования многомерности интеллекта;
- самоорганизация и саморазвитие ИИ;
- определения рекурсивных связей между ИИ и внешней средой;
- исследования приоритетов координаций;
- выделение системных единиц ИИ.

Разработка схем интеграции при системном подходе, анализе и синтезе к построению ИИ приводит к возникновению гибридного (синергетического) ИИ. Основное здесь — единый, системный подход, который может быть выражен через следующие компоненты триады: агент — многоагентная система (МАС)—сообщество многоагентных систем.

Под интеллектуальным агентом в теории ИИ понимаются физические или виртуальные элементы, способные: действовать на любые другие элементы; стремиться к некоторым целям, общаться с другими агентами, накапливать и использовать собственные ресурсы; воспринимать среду и ее части, строить частичное представление среды; адаптироваться, самоорганизовываться, саморегулироваться, саморазвиваться и самовоспроизводиться [42]. Тогда любая МАС, подобно ИС, состоит из множества агентов, множества задач, внешней среды, отношений между агентами, множества действий и операций над объектами.

МАС есть популяция простых и независимых агентов, в которой каждый агент самостоятельно реализуется в локальной среде и взаимодействует с другими агентами. Связи между агентами являются горизонтальными, глобальное поведение агентов определяется на основе нечетких правил.

Базовая МАС есть кортеж длины три: $\langle \tilde{\Phi}, X, Y \rangle$, где $\tilde{\Phi}$ — график нечеткого соответствия; X — область отправления (вход); Y — область прибытия (выход), причем $X \times Y \subseteq M$, где M — множество, задающее область определения нечеткого соответствия. Набор таких базовых МАС может образовать иерархическую ИС, инвариантную к внешним воздействиям среды [42]. Тогда нечеткую эволюционирующую МАС определим как кортеж длины шесть: $\langle C, \tilde{\Psi}, I, K, S, E \rangle$, где $C = \{1, 2, \dots, n\}$ — множество агентов; $\tilde{\Psi}$ — семейство нечетких соответствий ($\tilde{\Psi} = \langle \tilde{\Phi}, X, Y \rangle$); I — множество действий агентов; K — множество коммуникационных связей; S — множество нечетких состояний; E — множество эволюционных нечетких стратегий.

На основе таких моделей можно анализировать процессы принятия решений с использованием ИС.

С понятием ИИ тесно связано направление «искусственная жизнь» (ИЖ) [46, 48]. Оно занимается исследованием интеллектуального поведения ИС в рамках адаптации, выживания, самоорганизации, построения децентрализованных систем. В основе ИЖ лежат функциональный анализ и синтез, моделирование, эволюция и имитация ЕС.

Уровень ИИ характеризуется: многосвязностью и мобильностью ассоциаций, проявляющихся в процессе поиска, быстротой анализа возникающих перспективных решений, адаптацией к внешней среде, способностью ориентироваться в неформально, неточно и некорректно сформулированном задании. Иногда говорят об интеллекте как феномене действительности, включающим в себя человека и технологии [49, 50].

И. Кант и Г. Гегель [51–56] считали, что интеллект ассоциируется с разработкой категорий разума и рассудка. При этом можно считать, что природа ИИ двойственная, т. е. она одновременно логическая и биологическая. В этой связи ИИ — это определенная форма равновесия, к которой тяготеют все структуры, образующиеся на базе восприятия, мышления и адаптации. Мышление — орудие высшей ориентировки человека в окружающем мире и в самом себе. Условно можно считать, что ИИ — это имя, обозначающее высшие формы организации или равновесие когнитивных структур. В [38, 39] ИИ рассматривается как явление адаптации. Адаптация — то, что обеспечивает равновесие между воздействием организма на среду и обратным воздействием среды. Действие организма на среду называется ассимиляцией. Действие среды на организмы — аккомодацией. Тогда адаптация — это равновесие между ассимиляцией и аккомодацией, т. е. равновесие во взаимодействии субъектов и объектов [43].

Для повышения уровня «интеллекта» ИС и расширения ее возможностей, свободного формального и неформального взаимодействия с пользователем в нее закладываются знания, относящиеся к предметной области и внешней среде. Кроме того, ИС (ее математическое обеспечение) снабжается математическими моделями и методами для построения выводов и механизмов обучения и самообучения, преобразования и практического использования введенных знаний [43, 57, 58].

Укрупненная структура ИС по аналогии с другими интеллектуальными системами будет состоять из четырех подсистем: адаптивной, интерактивной, обрабатывающей и управления.

Первая подсистема состоит из нескольких нерархических уровней: микро-, макро- и метауровней. На каждом из этих уровней происходит моделирование эволюции и адаптация алгоритмов к окружающей среде. Такая трехуровневая адаптивная подсистема может быть представлена на рис. 2.1.

Вторая подсистема анализирует входные описания на языке пользователя на основе имеющихся знаний и формирует внутреннее неполное и нечеткое представление задачи.

Третья подсистема превращает неполное и нечеткое описание задачи в полное и четкое и снова передает его интерактивной подсистеме. Далее

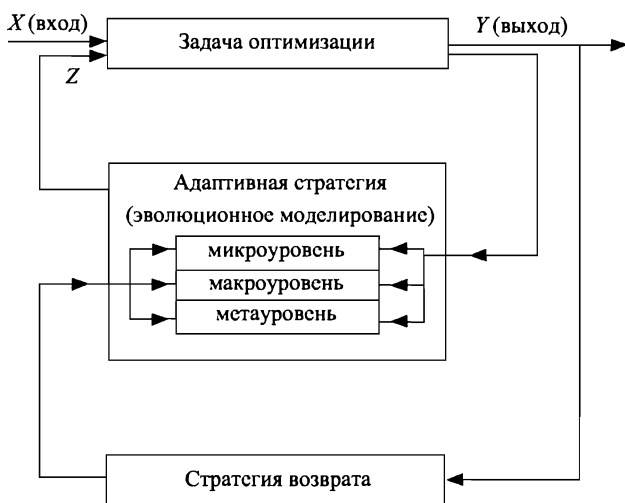


Рис. 2.1. Пример трехуровневой адаптивной системы

процесс происходит итерационно до получения удовлетворительного результата.

Четвертая подсистема, используя координирующие блоки и обратные связи, управляет процессом решения, взаимодействуя с первыми тремя подсистемами.

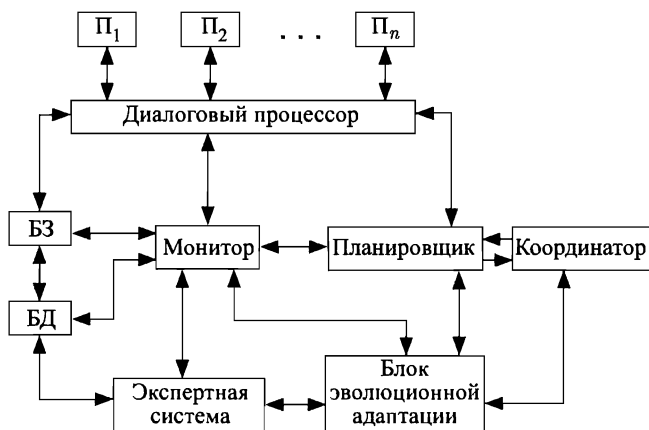


Рис. 2.2. Концептуальная схема искусственной системы.

БД — база данных, БЗ — база знаний

Приведем теперь одну из возможных концептуальных схем ИС (рис. 2.1.) [19, 40]. Здесь в качестве множества $\{P_1, P_2, \dots, P_n\}$ выступает множество пользователей. Диалоговый процессор является структурной единицей интеллектуального интерфейса ввода-вывода. Он обеспечивает заданный сценарий диалога в зависимости от степени подготовки пользователей и

интеллектуальности системы. Диалоговый процессор выполняет одновременный ввод–вывод графических, цифровых, речевых и других образов. Диалоговый процессор реализуется программным, аппаратным или программно-аппаратным способом. Его наилучшей формой в ИС является совместная программная и аппаратная реализация на основе нейронных сетей и их модификаций. Диалоговый процессор содержит набор сценариев диалога, меню диалога, генераторы случайного и направленного поиска, блоки перебора.

Пользователь во взаимодействии с диалоговым процессором на основе блочно-иерархического подхода и принципов декомпозиции производит разбиение заданной задачи большой размерности на совокупности функциональных подзадач меньшей размерности. Основная цель планировщика — поиск информации. Совместно с экспертной системой (ЭС) он осуществляет анализ преобразованных данных, полученных из диалогового процессора. В зависимости от значений внутренних, внешних и управляющих переменных, конструктивных ограничений, заданных режимов работы и т. п. выбирается несколько альтернативных путей решения. Каждому пути соответствует управляющий программный блок из монитора. Под управлением этого блока осуществляется выбор, редактирование, настройка и стыковка отдельных программных модулей, определяются типы и форматы всех данных, формируется рабочий комплекс программ. На рис. 2.1. координатор совместно с БЗ и блоком эволюционной адаптации через планировщик осуществляют реализацию процедур поиска для нахождения множества решений и выбор некоторого подмножества эффективных решений среди имеющихся.

Одним из основных, интеллектуальных, компонентов ИС является БЗ [35, 36, 57–59]. Понятие «знания» является расплывчатым неформальным понятием. Под знаниями понимают нечеткие и четкие множества фактов и сведений, описывающих информацию относительно определенной предметной области, позволяющих решать определенный класс задач. Обычно выделяют следующие типы знаний:

- физические — знания об окружающей среде и внутренних свойствах объектов;
- предметные — знания о данной предметной области;
- синтаксические — знания о правилах построения рассматриваемых структур;
- семантические — знания о конкретном смысле и значении элементов в структуре;
- декларативные — знания о фактах и данных;
- процедурные — знания об алгоритмах, методах, эвристиках;
- метазнания — обобщенные знания;
- «ИНЬ и ЯН» — знания о преобразованиях и переменах на основе принципов противоположности;
- нечеткие (синергетические) — знания об интеллектуальном опыте ЕС и ИС.

Основные характеристики знаний — это интерпретируемость, концептуальность, структурированность, именнованность, иерархичность, активность, рефлексивность, связность и степень согласованности.

Основная цель БЗ — получать, хранить, перерабатывать и создавать упорядоченные знания на основе данных и нечетких неупорядоченных знаний. Обычно в интеллектуальных ИС такая БЗ состоит из трех основных компонентов: правил, вопросов–подсказок, переводов. Структурная схема БЗ состоит из трех основных блоков: база общих знаний, база системных знаний, база прикладных знаний. В первом блоке хранятся общие знания, необходимые для решения всех оптимизационных задач. Во втором блоке — знания о всех внутренних связях самой системы. В третьем блоке — все прикладные знания, например, описание предметных областей, правила и ограничения, комплексы алгоритмов и т. п. В отличие от стандартных БД, которые также присутствуют в интеллектуальных ИС и взаимодействуют с БЗ, последние позволяют обрабатывать знания и в результате этого получать новые упорядоченные знания.

В настоящее время ЭС — это программно-алгоритмические комплексы, выполняющие разнообразные функции. Они могут консультировать пользователя; анализировать промежуточные и конечные результаты; обучать пользователя и ЭВМ; обучаться в процессе решения конкретной задачи; давать советы пользователю и ЭВМ; классифицировать рассматриваемую проблему; производить поиск на заданных математических моделях; принимать конкретное решение на любом этапе оптимизации; делать достоверные выводы из неполных и нечетких данных и знаний; взаимодействовать с другими ЭС; передавать и приобретать новые знания. Другими словами, ЭС помогают интеллектуальным ИС накапливать и обрабатывать различные виды глубинных и поверхностных знаний или их представлений в виде соответствующих моделей. Глубинные знания обычно помогают отображать разрабатываемую структуру. К ним относятся причинные модели, категории, абстракции, аналогии.

Аналогия — один из основных приемов описания интеллектуальных ИС. В [51–56] отмечено, что новая идея появляется в результате сравнения двух объектов, которые еще не сравнивались, а процесс познания есть отыскание аналогий. При этом важны не поверхностные аналогии, а изоморфизмы в математическом смысле, т. е. строгое соответствие между всеми элементами сравниваемых систем. На пути проведения аналогий между концепциями происходит целенаправленное развитие новой теории интеллектуальных ИС. Поверхностные представления знаний — это простые эмпирические ассоциации, которые формируются на основании условных правил, типа «Если X, то Y» или их модификаций. Экспертная система моделирует и интерпретирует действия пользователя по организации его знаний об объекте и делает из них выводы. В определенном смысле ЭС интеллектуальных ИС помогают ЭВМ применить опыт пользователя для принятия эффективных решений [57].

При построении моделей ИС важным является организация логического вывода. Если имеется множество посылок $\varphi_1, \dots, \varphi_n$, то из него

выводим заключение φ . Утверждается, что выполнен логический вывод, если найдена цепочка формул, заканчивающаяся той формулой, которую мы должны вывести, а каждый член этой цепочки есть либо аксиома, либо гипотеза, либо заключение, полученное по правилам дедуктивного вывода из предыдущих членов цепочки [60, 61].

Рассуждение — это получение заключения из посылок с использованием некоторых эвристических правил. Согласно [35, 36] принципиальные черты искусственного интеллекта есть выделение существенного в знании, способность к рассуждениям, рефлексия, выдвижение цели и выбор средств ее достижения, познавательная активность, адаптация к ситуации, формирование обобщений и обучения, синтез познавательных процедур.

Д. Поспелов [40, 115] определил 10 горячих точек в области ИИ, связанных с разработкой ИС:

- переход от вывода к аргументации;
- проблема оправдания;
- порождение объяснений;
- поиск релевантных знаний;
- понимание текстов;
- когнитивная графика;
- многоагентные системы;
- сетевые модели;
- метазнания.

ИС могут быть описаны совокупностью знаков с отношениями между ними. Такие модели соответствуют ИС лишь частично, что делает отношение моделирования расплывчатым. Для формализации процесса используется семиотика — наука о знаковых системах. Прикладная семиотика основана на семиотическом моделировании. Оно описывает динамику жизнедеятельности ИС при изменении ее знаний об окружающем мире и способах поведения в нем. Прикладная семиотика позволяет моделировать процессы, протекающие в открытых или динамических системах [61, 66].

Понятийной основой прикладной семиотики является треугольник Фреге, показанный на рис. 2.3. Множество $X = \{x_1, x_2, x_3\}$ описывает представления человека или ИС. Здесь, например, x_1 — имя, x_2 — понятие, x_3 — представление.

Сущности реального мира (внешней среды) — это объекты, явления, процессы. Их называют денотаты x_4 . Денотат ИС недоступен, а представляется через модели путем установления соответствия. Как видно из рис. 2.3, треугольник Фреге — это неориентированный граф $G_\Phi = (X, U)$ с двумя подмножествами ребер, $U = \{u_1 \cup u_2\}$, $u_1 = \{1, 2, 3\}$, $u_2 = \{4\}$. Ребро 1 по имени объекта позволяет активизировать сведения об этом элементе. Ребро 2 позволяет найти информацию о свойствах рассматриваемого элемента. Ребро 3 устанавливает соответствие имени элемента с его моделью. На основе факторизации эту конструкцию можно представить в виде двух блоков X , x_4 . Информационная единица, структурой которой является множество X , называется в прикладной семиотике знаком. Вершина

$x_1 \in X$ определяет связи наследования. Отношения наследования обычно образуют иерархическую структуру в системе знаков. Очевидно, что между сетью знаков и сетью фреймов существует аналогия. Это позволяет определить операции на сетях из знаков — фреймов, выполнять операции вывода знаний и формирования новых знаний. Граф G_Φ соответствует триаде: синтаксис, семантика, прагматика. Синтаксис (x_1) определяет кодирование знака; семантика (x_2) — значение знака; прагматика (x_3) — процедуры и действия. В ИС такая совокупность процедур обычно реализуется в планировщиках и определяет эффективность работы ИС.

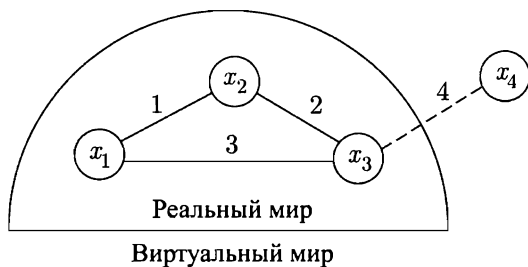


Рис. 2.3. Конструкция треугольника Фреге

В настоящее время в ИС также используются семиотические БЗ. Основными требованиями в них для объектов, называемых знаниями, являются именованность, структурированность, иерархичность, связность, активность и рефлексивность [59, 63]. БЗ удобно представлять в виде сети фреймов, гиперграфов или графов, причем каждый фрейм связывает воедино знания о данной ситуации и предсказывает, какие объекты будут обрабатываться, какие события могут произойти. В математических моделях интеллектуальных ИС фреймы являются совокупностью процедурных и декларативных знаний о прикладных моделях и состоят из множества слотов. Последние содержат знания о конкретной ситуации. Кроме того, слоты могут осуществлять вызов прикладных программ из программного обеспечения и т. д. Слоты одного фрейма взаимодействуют со слотами других фреймов, образуя в случае бинарных связей граф, а в случае n -арных связей — гиперграф с упорядоченным отношением на множествах вершин и ребер.

Важной задачей является поиск и преобразование информации в графовых и гиперграфовых моделях. Можно предложить совместное использование экспертной системы, методов эволюционной адаптации и синергетических подходов для эффективного решения таких задач. ЭС в интеллектуальных ИС позволяют разработать структурированную схему, отражающую весь ход процесса оптимизации. Эта схема дает возможность составить конечное множество вопросов, которые помогут пользователю эффективно провести весь процесс принятия решений. Если принятие решений определить как процесс преобразования потребности в результат, являющийся элементом в эволюционирующей технологической среде, то он интерпретируется как

локальный акт самоорганизации в интегрированной среде, который планируется и направляется самой средой.

В настоящее время в существующих ИС в основном реализуются бинарные связи. Бинарная парадигма согласно [64, 65] описывает ИС как совокупность парных отношений, утрачивая перспективу постижения целостности. Поэтому структура выбора «ИЛИ-ИЛИ» упрощена. В этой связи для комплексного представления ИС будем использовать триединый (триадный) подход. Триадой считается совокупность из трех взаимосвязанных элементов. В [64] различают три основных вида триад: линейные, переходные, системные. В линейных триадах все три элемента расположены на одной оси в семантическом пространстве. Переходные триады характеризуются формулой: тезис–антитезис–синтез. В системных (целостных) триадах единство создается тремя элементами одного уровня, каждый из которых может служить мерой совмещения двух других, так что потенциально все три равноправны. Системные триады обладают универсальным семантическим свойством, аналогичным природной способности человека мыслить одновременно понятиями, образами, символами. Семантическая формула системной триады: рациональность–интуитивность–эволюционность. Системная триада является простейшей структурной ячейкой синтеза. Третий элемент необходим для решения проблемы бинарных противоречий, как мера их компромисса, как условие существования. Результат синтеза можно представить как вершину тетраэдра, в основании которого системная триада.

Критерием целостности системной триады может служить принцип неопределенности–дополнительности–совместимости. Здесь каждая пара элементов находится в отношении дополнительности, а третий задает меру совместимости. Количественно эта закономерность видна на примере асимптотической математики. Она определяется триадой: точность–локальность–простота. Совмещение точности и простоты достигается по мере локализации ИС.

Порождающий (креативный) взгляд на структуру ИС может быть также описан триадой: *способ действия* $\perp$ *предмет действия* = *результат действия*. В общем случае известно большое число триад [64, 65]. Например: *закон природы* $\perp$ *материальная субстанция* = *феноменальный мир*; *замыслы* $\perp$ *потенция* = *произведение*. Использование и популярность триад связана с моделью содержательной, развивающейся, эволюционирующей ИС. В этой связи при моделировании ИС вместо диад (есть ровно одна причина и одно следствие) будем рассматривать триады, т. е. допускать множественность причин и следствий событий. Минимальная возможность и есть полный граф на три вершины (креативная триада). На его основе можно неоднозначно моделировать эволюцию. Это дает возможность генерировать множество сценариев развития событий, неоднозначность будущего и возможного прошлого. Например, выражение $f(x) = y$, также представляет собой триаду: f (функция) + x (аргумент) = y (значение функции). Следовательно, триадный подход может служить некоторой основой формализации простейших законов природы и позволяет создавать описание более сложных законов и структур.

Интеллектуальное поведение ИС можно также организовать на основе триады: цель–план–стратегия. Для адекватного описания ИС и интеллектуальной деятельности также требуется триединый механизм: индукции–дедукции–абдукции. Идея абдукции следующая. Даны группы фактов и выдвинуты некоторые гипотезы. Если эти гипотезы объясняют в некотором смысле эти факты, то они принимаются. Согласно [34, 60], абдуктивный подход позволяет среди множества возможных гипотез выбирать именно те, которые необходимы в процессе анализа и синтеза ИС. Механизм рассуждений в ИИ есть не только и не столько простая имитация человеческих рассуждений, сколько одновременно их имитация и усиление.

ИС, кроме рассмотренных БЗ, БД и других блоков, включают: решатель задач и интеллектуальный интерфейс. Решатель — это рассуждатель плюс вычислитель. Рассуждатель реализует синтез познавательных процедур. В результате обучения ИС с помощью БД производится расширение БЗ. Синтез познавательных процедур состоит из трех этапов: индукция, система связанных аналогий, абдукция.

Реализация ИС — это понятийное построение знаний, что предполагает преобразование идей в понятие. Идея — это терм, окруженный релевантным знанием. Пусть имеется терм T , обозначающий идею и характеризующийся множеством высказываний. Тогда если φ — отношение релевантности, то можно определить знания, входящие в терм, то есть, связать все те высказывания, которые имеют отношение друг к другу. Элементы a, b находятся в отношении релевантности φ , если они «имеют заданное нечеткое отношение друг к другу». Тогда, согласно [66], понятие T' — это идея T с упорядоченным значением такая, что: задано содержание создаваемой ИС; заданы отображения описаний ИС в множество моделей; задана структура реализации ИС. В ИС основная проблема — это обработка огромных массивов нечеткой информации. Поэтому структурированность и иерархичность, то есть, упорядоченность информации, упрощает эту задачу.

2.2. Иерархия в интеллектуальных ИС

Согласно теории В. Вернадского, Вселенная — единая саморазвивающаяся система. Во всех процессах, происходящих во Вселенной, присутствуют случайные факторы (стохастические). Они влияют на развитие процессов и придают им некоторую неопределенность. Поэтому возникновение новой структуры, в частности интеллектуальной ИС, может проходить по разным траекториям [8, 67, 68]. Новые качественные особенности интеллектуальной ИС появляются благодаря изменчивости. В эволюционирующей интеллектуальной ИС всегда существует зависимость от прошлого, т. е. от него зависят как настоящее, так и будущее. Эту зависимость условно называют наследственностью интеллектуальных ИС.

Принципы отбора позволяют выбрать из возможных виртуальных состояний некоторое множество разрешимых. Они допускают бифуркационные состояния в интеллектуальной ИС, из которых возможен переход во множество новых состояний.

ЕС дает возможность появиться новым формам организации материи, эти формы как бы потенциально ею заготовлены, но детали процесса непредсказуемы, аналогично может происходить и в интеллектуальной ИС.

Интеллектуальные ИС можно условно классифицировать:

- как изолированные (нет обмена с внешней средой);
- закрытые (есть только обмен энергией);
- открытые (есть все виды обмена).

В открытых системах, далеких от равновесия, возможны новые динамические состояния материи (диссипативные системы), приводящие к самоорганизации.

Во всех случаях в ЕС и ИС имеется иерархия, иначе говоря, рассматриваемые системы состоят из большого числа подчиненных подсистем. Отметим, что многоуровневое управление в ИС иногда бывает предпочтительнее централизованного подхода, так как управление может осуществляться быстрее и с меньшим требованием к объему памяти; система менее чувствительна к изменениям структуры взаимодействий — изменениям, которые могут иметь регулярный или случайный характер.

Приведем двухуровневое представление интеллектуальной ИС (рис. 2.4). Это простейшая двухуровневая иерархия. Здесь $m_1, \dots, \dots, m_n, m_{n+1}$ — входы; y — предварительный выход; G — глобальная функция качества; V — окончательный выход. Координатор может рассматривать независимость подпроцессов друг от друга и применить принцип согласования взаимодействий для их управления. Важная проблема в интеллектуальных ИС — реализация перехода с уровня на уровень и определение, как поведение системы на одном уровне влияет на системы, расположенные на соседних уровнях.

Каждый элемент здесь имеет собственную цель, которая зависит от параметра, получаемого от координатора. Координатор может иметь цель, отличную от глобальной цели, и выбирать параметры так, чтобы обеспечить выполнение своей собственной цели. Если зависимость между целями закономерна, интеллектуальная ИС может достигнуть глобальной цели. Интегральное поведение определяется действиями координатора и его стремлением добиться выполнения собственной цели. Таким образом, интеграция в интеллектуальных ИС достигается посредством координации и адаптации [67–69].

Приведем основные характеристики иерархических интеллектуальных ИС:

- последовательное вертикальное расположение подсистем, составляющих общую систему (вертикальная декомпозиция);
- приоритет действий или право вмешательства подсистем верхнего уровня в работу нижних уровней;
- зависимость действий подсистем верхнего уровня от фактического исполнения нижними уровнями своих функций.

Взаимодействие между уровнями в интеллектуальной ИС может проходить как между двумя близлежащими уровнями, так и между любым числом уровней. Входы и выходы могут быть распределены по всем уров-

ням. Качество работы всей интеллектуальной ИС обеспечивается обратной связью.

Интеллектуальные ИС обычно классифицируют по уровням: описания или абстрагирования, сложности, организации выполнения управления для решения задач. Интеллектуальная ИС задается семейством моделей, каждая из которых описывает поведение системы с точки зрения различных уровней абстрагирования. Для каждого уровня существует ряд характерных особенностей и законов, которые и описывают поведение интеллектуальной ИС. Для лучшего функционирования необходима как можно большая независимость моделей для различных уровней.

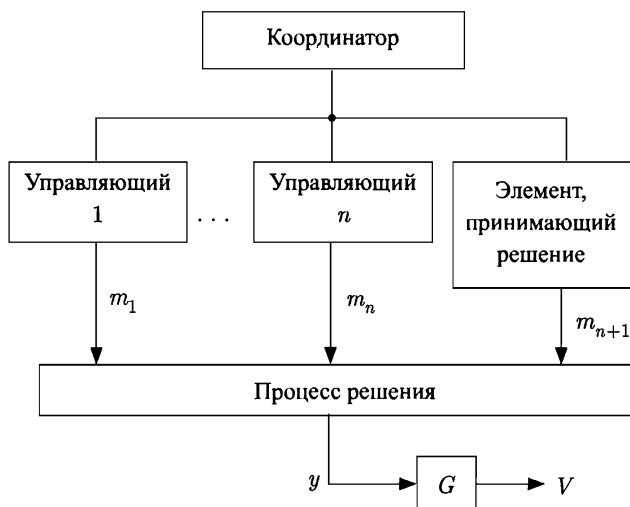


Рис. 2.4. Двухуровневое представление интеллектуальной ИС

Фундаментальную роль в интеллектуальных ИС играют стратифицированные или многоуровневые модели. Тогда, в зависимости от задачи, сначала можно ограничиться одним уровнем, а затем либо детализировать свои знания, двигаясь вверх, либо углубить их, двигаясь вниз. Цель в иерархических интеллектуальных ИС надо выбирать так, чтобы ее можно было развернуть в подцели и тогда появляется возможность оценить, приблизились ли мы к первоначальной цели. Например, на рис. 2.5 показана одна из возможных многоуровневых интеллектуальных ИС. Здесь ИИС₁ ÷ ИИС₈ различные детализированные модели подсистем интеллектуальной ИС.

Рассмотрим функциональную иерархию управления в интеллектуальной ИС в условиях полной неопределенности [67–69]. Здесь необходимо осуществить:

1. Выбор стратегии управления.
2. Уменьшение или устранение неопределенности.
3. Поиск предпочтительного или допустимого способа управления, удовлетворяющего заданным ограничениям.

Простую иерархию интеллектуальной ИС можно составить из трех слоев: слой выбора, слой обучения или адаптации, слой самоорганизации (рис. 2.5).

Задача слоя выбора — конкретизация множества неопределенностей. Выбор — это анализ альтернативных решений.

Основная цель слоя адаптации — обучение, т. е. определение возможностей сужения множества неопределенностей и упрощения работы слоя выбора. Адаптация — это управление с обратной связью на основе статистических методов.

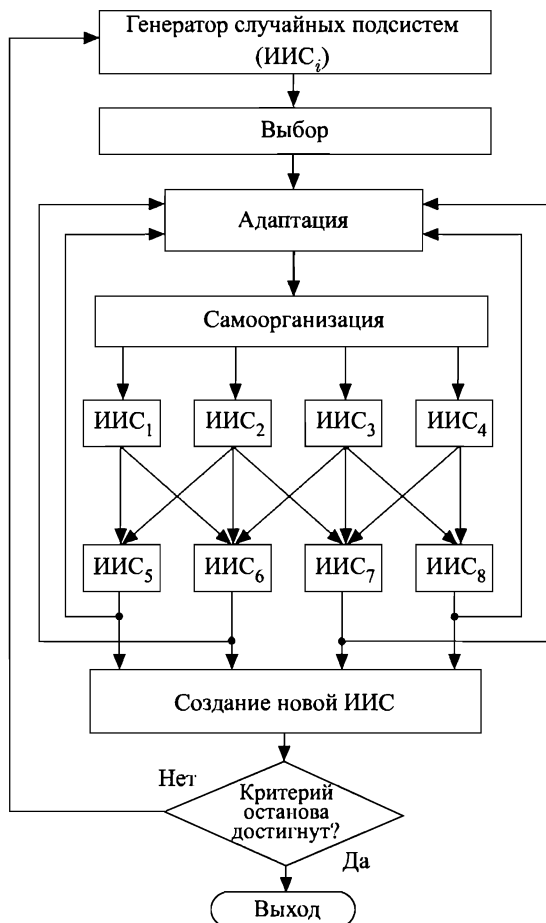


Рис. 2.5. Многоуровневая ИИС

Самоорганизация — эвристические, синергетические, гомеостатические методы поиска. Слой самоорганизации должен выбирать структуру, функции и стратегии для нижележащих слоев, чтобы по возможности при-

близиться к глобальной цели. Если общая цель не достигается, этот слой может изменить критерии на первом слое или стратегию обучения на втором в случае неудовлетворенности оценки. Всегда необходимо учитывать меняющиеся технологические условия и ограничения, налагаемые оборудованием и внешней средой. Задача управления в интеллектуальной ИС — удержать (в условиях неизбежных отклонений) соответствующие переменные около заранее заданных значений.

Приведем один из возможных примеров многоуровневой организационной иерархии в интеллектуальной ИС [8, 67–70]:

- интеллектуальная ИС состоит из семейства взаимодействующих подсистем;
- некоторые из подсистем являются элементами, принимающими решения;
- в интеллектуальной ИС элементы располагаются иерархически.

В процессе решения оптимизационной задачи элементам нижних уровней должна быть предоставлена некоторая свобода в выборе их собственных решений. Эти решения в частном случае могут быть такими, которые выбрал бы верхний уровень. Элемент верхнего уровня имеет дело с более крупными подсистемами или с более широкими аспектами поведения интеллектуальной ИС. Период управления для элемента верхнего уровня больше, чем для элементов нижних уровней. Элемент верхнего уровня имеет дело с более медленными аспектами поведения интеллектуальной ИС.

Проблема, решаемая элементом нижестоящего уровня, зависит от вышестоящего элемента и заключается в выработке значений определенного параметра. Обычно вводится приоритет действий вышестоящего элемента путем вмешательства до начала работы. Это основной способ координации. Вмешательство до начала основано на прогнозировании поведения как интеллектуальной ИС, так и окружающей среды. Вышестоящий элемент в ходе такого процесса определяет функции качества для оценки деятельности нижестоящего элемента. Он должен исправить посланные ранее элементами нижестоящего уровня инструкции, если допущения, выработанные на основе этих инструкций, неверны. Эти действия координирующего элемента после управления есть корректирующее или поощряющее вмешательство. Приведем следующие основные способы координации [69].

Координирование путем прогнозирования взаимодействий.

Здесь вышестоящий элемент посылают нижестоящим значения будущих связующих сигналов. Нижестоящие элементы начинают вырабатывать свои локальные решения в предположении, что связующие сигналы будут такими, какими их предсказал координирующий элемент.

Координирование путем оценки взаимодействия.

Вышестоящий элемент задает диапазон значений для связующих сигналов. Нижестоящие элементы рассматривают эти сигналы как возмущения, принимающие любое значение в заданном диапазоне.

Координирование путем «развязывания» взаимодействий.

В этом случае элементы нижестоящего уровня трактуют связующий сигнал как дополнительную переменную решения.

Координирование типа «наделение ответственностью».

Вышестоящий элемент снабжает нижестоящие элементы моделью зависимости между его действиями и откликом системы.

Координирование путем «создание коалиций».

Нижестоящие элементы знают о существовании других элементов, также принимающих свои решения на том же уровне. Вышестоящий элемент определяет, какого уровня связи разрешены между ними. Это приводит к коалиционным или конкурентным отношениям между нижестоящими элементами.

Координация имеет два аспекта: самоорганизации (изменения структуры) и управления (выбор координирующего вмешательства при фиксированной структуре). Самоорганизация может относиться к изменениям функций и взаимосвязей, используемых в процессе координации. Работа вышестоящего элемента сводится к выбору способа координации, модифицированию функций, координирующих воздействия, определяющих стратегии нижестоящего элемента.

Выбор модели для вышестоящего элемента основывается на признании того факта, что для него управляемый процесс описывается как взаимодействие семейства взаимосвязанных подсистем, каждая из которых преследует собственные цели. Иерархическое упорядочивание часто связано с процессом изменения структуры уже существующей интеллектуальной ИС для повышения эффективности ее работы.

Опишем интеллектуальную ИС как отображение $M: X \rightarrow Y$. Это отображение абстрактного множества X в Y ; пара $(x, y) \in M$, когда y является решением определенной задачи, конкретизация которой осуществляется посредством задания x . Тогда M — это модель интеллектуальной ИС. Пусть $g: X \rightarrow V$ функция, отображающая произвольное множество X в множество V , частично или полностью упорядоченное отношением «меньше» или «равно». При изменении определенных условий (управляющих параметров) в системах образуются качественно новые структуры в макроскопических масштабах. Интеллектуальная ИС обладает способностью переходить из одного нечеткого состояния покоя в неоднородное, но хорошо упорядоченное состояние или даже в одно из нескольких упорядоченных состояний. Система может совершать случайные движения, что считается хаосом [11, 19, 64, 65]. Такие процессы самоорганизации в макромасштабах могут приводить к возникновению качественно новых интеллектуальных ИС.

2.3. Порядок и хаос в моделях ИС

Науку, изучающую взаимодействие подсистем внутри системы, в результате которого в процессе самоорганизации возникают новые упорядоченные структуры в макроскопических масштабах, называют синергетикой [70–77].

Введение общего понятия подчинения как одного из основных принципов самоорганизации принадлежит Г. Хакену [70]. Используя это понятие,

в сложных системах можно исключить большое число переменных и свести задачу к задаче меньшей размерности. Дифференциация клеток в биологии и процесс эволюции могут служить примерами самоорганизации. Общие принципы, управляющие возникновением самоорганизующихся структур и (или) функций, — это основной вопрос синергетики.

ЕС дают большое число примеров упорядоченных и организованных структур. В биологических системах ничего не происходит без коопераций их частей на высоком уровне. Синергетические процессы позволяют биологическим системам трансформировать энергию, предварительно преобразованную на молекулярном уровне, в ее макроформы. В этой связи можно считать, что эволюция — это синергетический процесс образования все новых и новых макроструктур (т. е. новых видов). Модели эволюции биомолекул основаны на математической формулировке принципа Ч. Дарвина, т. е. выживании более приспособленного вида. Биомолекулы, как считают, размножаются автокаталитически. Такой механизм обуславливает отбор, приводящий в сочетании с мутациями к эволюционному процессу [7].

В процессе управления интеллектуальной ИС решающую роль играет динамика. Это — как бы обобщенный дарвинизм, действие которого распространяется на органический и неорганический мир. По Г. Хакену, возникновение макроструктур обусловлено рождением коллективных «мод» под действием флуктуаций, их конкуренции и выбора наиболее приспособленной моды. Решающую роль играет параметр t — время. Следовательно, синергетика исследует эволюцию систем во времени. Временные, пространственные и пространственно-временные структуры возникают, а не накладываются на систему извне. Процессы, приводящие к возникновению таких структур, Г. Хакен называет самоорганизацией. В ходе эволюции систем может происходить множество процессов самоорганизации. Глобальное воздействие на систему окружающей среды может вызвать увеличение числа компонент интеллектуальной ИС, изменение управляющих параметров и самоорганизацию. При медленном изменении системы под воздействием окружающей среды она в критических точках может переходить в новые состояния, отличающиеся более высоким порядком или структурой.

Структуры могут возникать в результате самоорганизации, когда интеллектуальная ИС из некоторого начального (неупорядоченного или однородного) состояния переходит в другое конечное состояние.

Синергетика занимается изучением временной эволюции систем. В синергетике управляющие параметры изменяются непредсказуемым образом, а далее изучаются состояния, в которые система переходит под воздействием управления. Переломный критический момент неопределенности будущего развития интеллектуальной ИС можно назвать точкой бифуркации, т. е. точкой «разветвления» возможных путей эволюции системы. Флуктуации обретают решающее значение в тех точках, в которых происходят бифуркации [70–77]. Создание структур различной природы синергетика связывает с такими условиями существования систем, как открытость, нелинейность, неравновесность. Открытость — это свободный обмен информацией (энергией) с внешней средой. Синергетика рассматри-

вает нестационарные состояния, динамику, взаимопереходы, разрушения и созидания интеллектуальных ИС. Согласно [75–77], образ мира в синергетическом описании следующий: мир открыт и сложноорганизован, он не «ставший, а становящийся», непрерывно возникающий и изменяющийся, он эволюционирует по нелинейным законам, связанным с выбором дальнейшего развития. Каждый период в истории приводит к своей модели ЕС. Существует еще ряд определений. Синергетика — наука об универсальных законах эволюции в природе и обществе. Синергетика — наука о самоорганизации физических, биологических и социальных систем; наука о коллективном, когерентном поведении систем различной природы. Примером синергетических систем в биологии считаются высокоупорядоченные структуры, образованные при морфогенезе. Известно, что в таких системах могут возникать как упорядоченные, так и хаотические колебания.

В начале XX века считалось, что природа развивается согласно второму началу термодинамики, т. е. ее развитие сопровождалось ростом энтропии, что приводит к выравниванию всех градиентов в природе, и мир смещается к состоянию равновесия, к серому однородному хаосу. Это положение называется «тепловая смерть Вселенной» [9, 74]. Развитие живого мира подчиняется прямо противоположным законам — энтропия может понижаться, увеличивается рост порядка, растет разнообразие форм.

В ЕС, как правило, должны соблюдаться единые законы эволюции. Общий эволюционный процесс как процесс самоорганизации, несмотря на стихийность, обладает направленностью, так как идет рост разнообразия форм и сложности структур. ИС, по определению П. Анохина, комплекс избирательно вовлеченных компонентов, у которых взаимное действие и взаимоотношения принимают характер взаимодействия компонентов для получения фиксированного результата [78].

Известны [75–77] два основных фундаментальных свойства ИС:

1. Обмен с окружающей средой энергией, веществом и информацией.
2. Взаимодействие, то есть, когерентность поведения между компонентами.

Согласно Л. Больцману [64, 65] природа стремится к переходу от состояния менее вероятного к состояниям более вероятным. В [11] концепция хаос–порядок связана с понятием энтропии. Из статистической механики известно [63, 64, 76, 77], что в замкнутых системах энтропия может служить мерой относительной степени хаотичности. Существуют теоремы Больцмана о возрастании энтропии в процессе эволюции к равновесному состоянию. Они позволяют показать, что равновесное состояние соответствует не только максимуму энтропии, но и является устойчивым. В этой связи энтропия обладает совокупностью свойств, позволяющих использовать ее в качестве меры неопределенности при статистическом описании процессов в макроскопических системах.

Знаменитая H-теорема Больцмана формулируется следующим образом: при временной эволюции замкнутой системы к равновесному состоянию энтропия возрастает и остается неизменной при достижении равновесного

состояния. Теорема Гиббса утверждает, что при определенных условиях равновесное состояние отвечает максимуму энтропии и является наиболее хаотическим. Другими словами, согласно Д. Гиббсу, при одинаковости нормировки и средней энергии, энтропия равновесного состояния максимальна. Шеннон определил энтропию как меру неопределенности дискретных сообщений [76, 77]. Ю. Климонтович [79] предлагает использовать энтропию Больцмана–Гиббса–Шеннона в качестве меры неопределенности при статистическом описании процессов в произвольных открытых системах.

Согласно идеям И. Бернулли и И. Ньютона «природа всегда действует простейшим образом и ничего не делает напрасно. Она проста и не роскошествует излишними причинами вещей». Используя идеи «экономии», в природе возникли два основных экстремальных принципа. Они обладают лаконизмом, простотой и универсальным характером.

Экстремальный принцип I. Истинный путь светового луча отличается от всех возможных тем, что время движения вдоль него минимально.

Экстремальный принцип II. Утверждение об экстремуме некоторой величины — принцип наименьшего действия: «истинное движение отличается от всех возможных тем, что для него величина действия минимальна».

Согласно [75]: «Именно количество действия является истинной тратой природы; и именно оно [количество действия] выгадывается как можно менее при движении». Л. Эйлер утверждал: «Так как здание всего мира совершенно и выведено премудрым творцом, то в мире не происходит ничего, в чем бы не был виден смысл какого-нибудь максимума или минимума; поэтому нет никакого сомнения, что все явления мира с таким же успехом можно определить из причин конечных при помощи метода максимумов и минимумов, как из самих причин производящих». Предложение Л. Эйлера — идти к законам природы сверху путем дедукции от экстремальных принципов. Для этого необходимо:

- найти ту величину, которую экономит природа в данной области (целевую функцию);
- сформулировать соответствующий экстремальный принцип.

Из сказанного следует, что совершенству и целесообразности живой природы соответствуют экстремальные принципы. Авторы и другие исследователи считают, что аналогичные принципы возможны и для интеллектуальных ИС.

В работе [70] концепция хаос–порядок связана с понятием энтропии. Из статистической механики известно, что энтропия системы равна логарифму доступного ей объема фазового пространства, мерой которого является число N возможных микросостояний системы

$$S = k \ln N,$$

где k — постоянная Больцмана.

Тогда хаос, т. е. беспорядок, вносимый в макросистему, пропорционален относительному увеличению числа ее микросостояний:

$$dS = k \frac{dN}{N}. \quad (2.1)$$

Следовательно, если в системе под действием управления число N ее возможных состояний уменьшается, то есть, сжимается ее фазовый объем, то в системе увеличивается порядок. Когда в системе возможно лишь одно состояние ($N = 1$), ее энтропия равна нулю.

Говорят, что энтропия определяет состояние ЕС или ИС с позиции ее внутренней упорядоченности [75, 77]. Энтропия есть объективная мера нашего незнания, мера отсутствия информации о системе. Рост энтропии идет до $S = S_{\max}$, при котором возникает равновесное состояние. Формула (2.1) связывает энтропию с хаосом. Менее упорядоченное состояние (большой хаос) имеет большой статистический вес, так как оно может быть реализовано большим числом способов. Следовательно, энтропия есть мера неупорядоченности ЕС или ИС, причем порядок создается на основе эвристической деятельности ЛПР, а беспорядок — самопроизвольно, так как ему отвечает большая вероятность, большая энтропия.

В равновесии интеллектуальная ИС будет находиться в состоянии максимально возможной неоднородности. При этом в ней допустимы отклонения от наиболее вероятного значения, которые называются флуктуациями [70, 71]. Они тем менее вероятны, чем больше число элементов в интеллектуальной ИС. Следовательно, вероятность отклонения от наиболее ожидаемого состояния в интеллектуальной ИС убывает с ростом числа элементов. В эволюции интеллектуальной ИС каждая популяция ее элементов обладает наследственной изменчивостью, т. е. имеет место случайное отклонение от наиболее вероятного среднего значения. Обычно, когда отклонение мало, оно следует нормальному закону распределения случайных величин. Но наследственная изменчивость не затухает, как всякая флуктуация. Наследственные признаки закрепляются, если они имеют приспособительный характер, т. е. обеспечивают виду лучшие условия существования и размножения.

ЕС эволюционирует в направлении роста энтропии $S = S_{\max}$, если она изолирована, следовательно, ее развитие направлено в сторону равновесия в изолированной системе. ИС не несет памяти о своем эволюционном развитии, ЕС несет память об этом. В ЕС наследственность не затухает, а наследует и закрепляет те признаки, которые позволяют ей выжить. По Ч. Дарвину в ЕС происходит непрерывное рождение все более сложно организованных живых форм, структур и систем.

Общее изменение энтропии:

$$\Delta S = \Delta_i S + \Delta_e S,$$

где $\Delta_i S$ — внутренние изменения энтропии, а $\Delta_e S$ — приток или отток энтропии.

В открытой системе возможно состояние, когда энтропия уменьшается, следовательно, появляется принципиальная возможность самопроизвольного движения от хаоса к порядку. При большом оттоке энтропии в окружающую среду общая энтропия системы уменьшается. В этом случае уменьшается хаос в системе, то есть, в ней начинает возникать структурообразование [74, 75].

В результате эволюции прошло много туров естественного отбора и каждый раз отбирались «лучшие из лучших». Поэтому, очевидно, система должна быть в каком-то смысле оптимальной, экономной. В науке до сих пор остается открытым вопрос, что же экономит природа, создавая ЕС, — энергию? материалы? информацию? минимизирует энтропию? или что-то другое?

Существует биологический принцип оптимальности: принцип экономии энергии, $U = \min$. При этом ЕС имеет оптимально возможную конструкцию по отношению к экономии используемого материала и расходуемой энергии, необходимых для выполнения заданных функций.

Например, известно, что нейрон в своей работе решает лишь одну главную задачу — экономию энергии. Решая задачу об экономии энергии, совокупность нейронов проходит последовательные ступени самоорганизации. Если исходные параметры нейрона подобраны удачно, то в качестве «продуктов» этой эволюции в системе возникают такие свойства, как память, органы восприятия и воздействия на внешнюю среду, эмоциональные и интеллектуальные свойства.

Принцип оптимальной конструкции интеллектуальной ИС связан с ее поведением. Устойчивые формы поведения закрепляются, а конструкция многое предопределяет в поведении. Принцип экономии энергии должен сопровождаться дополнительными условиями: интеллектуальная ИС при функционировании не должна выходить за границы гомеостаза. Как отмечалось выше, мера неопределенности или мера разнообразия возможных состояний системы — это энтропия. Энтропия — мера свободы системы: чем больше состояний доступно системе, тем больше энтропия.

Принцип максимума энтропии:

$$H(x) = \sum_i P(x_i) \log \frac{1}{P(x_i)} = \max,$$

где $P(x_i)$ — вероятность достижения различных состояний.

Информация должна передаваться и обрабатываться с наименьшими затратами, за кратчайшее время, при наименьшем уровне помех, она должна быть наилучшим способом закодирована, представлена в оптимальной для восприятия форме.

Принцип максимума информации:

$$I(X, Y) = \max,$$

где X — стимул, условие внешней среды, воздействующей на интеллектуальную ИС; Y — реакция интеллектуальной ИС на стимул с целью получения полезного результата. Для достижения результата интеллектуальная ИС должна обеспечить максимум взаимной информации между условиями среды и реакциями:

$$H(X, Y) = H(X) + H(Y) - I(X, Y),$$

иначе говоря, чем больше взаимная информация, тем теснее связь, тем меньше энтропия $H(X, Y)$.

Приведем следствие из принципа максимума информации: ЕС и интеллектуальная ИС стремятся приспособиться ко все большему разнообразию условий внешней среды. Принцип максимума информации описывает не только поведение, но и процессы развития, адаптации, роста, приспособленности и т. д. Конечно, максимум информации условный, он зависит от ресурсов. Требование максимума информации (при условии ограничения ресурсов) можно заменить требованием максимума функции полезности или ЦФ. Принцип максимума информации обычно рассматривается как постулат. Он связан с эволюцией, с накоплением и отбором информации. Итак, принцип максимума информации состоит из принципа максимума энтропии и из принципа экономии энергии. Одна и та же закономерность часто повторяется на различных уровнях организации интеллектуальной ИС, при этом она обеспечивается различными механизмами эволюции.

Приведем основные свойства интеллектуальных ИС с точки зрения переработки информации:

- способность получать разнообразную информацию;
- возможность поддерживать высокое постоянство своей внутренней среды, несмотря на значительные изменения окружающих условий;
- экономичная структура хранения информации, опирающаяся на выделение признаков, характерных для объектов окружающей среды;
- многоуровневая структура переработки информации, которая способна самосовершенствоваться, создавая все новые и новые уровни управления.

Интеллектуальная ИС должна работать в виде «башни принятия решений». Нижний этаж осуществляет связь с внешней средой и переход на более высокие уровни осуществляется на основе результатов нижележащих уровней. На высшем этапе эволюции образуется также многоуровневая структура переработки информации. Процессы выработки критериев отбора информации, спускаемые на нижний уровень, должны быть основаны на всем опыте, накопленном интеллектуальной ИС. На высшем уровне материализуется многоуровневая информационная структура на основе морфологической эволюции (т. е. развитие конструкций, занимающихся переработкой информации).

Интеллектуальная ИС ориентируется в окружающей среде по корреляциям (статистическим связям) между различными признаками. Строя модель на основе учета коррелированной окружающей среды, интеллектуальная ИС уменьшает энтропию своей модели, следовательно, максимизирует информацию. В эволюции реализуется принцип максимума информации, чем в значительной мере и объясняется то совершенство и гармония, которую мы видим в окружающей среде.

Рост упорядоченности ЕС соответствует онтогенезу, а многообразие возникающих при этом форм описывается морфогенезом [12, 27]. В ходе развития неравновесных процессов в ЕС при некотором критическом значении внешнего потока энергии или вещества из неупорядоченных состояний из-за потери устойчивости могут возникать упорядоченности. Они

называются диссипативными структурами — это и есть самоорганизация. Неустойчивость означает, что флуктуации (возмущения) могут перестать быть просто шумом и превратиться в фактор, направленно действующий на глобальную эволюцию системы. Во всех процессах, происходящих во Вселенной, присутствуют случайные (стохастические) факторы. Они влияют на развитие процессов и придают им некоторую неопределенность. Эти процессы объясняются основными принципами синергетики [70–79]:

- предметом ЕС и ИС является не только общее, но и случайное;
- естественный порядок не является постоянным;
- для описания ЕС и ИС важны неформальные, нечеткие методы;
- детерминизм в описании ЕС и ИС не исключает случайность;
- развитие ЕС и ИС многовариантное и альтернативное;
- развитие ЕС и ИС проходит через неустойчивость — хаос не только разрушителен, но и конструктивен;
- процесс развития ЕС и ИС сочетает в себе дивергентные (рост разнообразия) и конвергентные (свертывание разнообразия) тенденции;
- развитие ЕС и ИС происходит по нелинейным законам;

Синергетика приближает нас к целостному образу мира, состоящего из хаоса и порядка, необходимости и случайности, динамизма и гомеостаза, организации и дезорганизации, гармонии и дисгармонии. Эта наука переносит акцент с проблем эволюции на проблемы организации и упорядочивания.

Существуют параллели между положениями теории самоорганизации и идеями древних восточных учений [51–56, 80]. Синергетика устанавливает связь между прошлым, настоящим и будущим интеллектуального и духовного опыта человека и позволяет говорить о диалоге человека и природы, определяет связь между историей человека и изменяющейся средой [75].

Путь к воссозданию равновесия связан с определением взаимосвязи современного естествознания и восточных учений. Согласно древней индийской и китайской философии мир (природа) — не атомарная совокупность предметов, а единая нерасчлененная, вовлеченная в бесконечное движение реальность, идеальная и материальная. Древняя китайская классическая «Книга перемен» утверждает [80]: перемены и преобразования представляют собой первичный аспект природы; структуры и симметрии, порожденные ими, нечто вторичное; человек следует законам земли; земля следует законам неба; небо следует законам Дао; Дао следует самому себе (законам своей внутренней природы). Необходимо решить проблему перехода от естественных принципов к количественным формализованным соотношениям. Для этого важно использовать законы, справедливые для всех форм существования материи и являющиеся инвариантными.

Смысл синергетики состоит в том, что в открытых системах, обменивающихся с внешней средой энергией, веществом, информацией, возникают процессы самоорганизации, т. е. процессы рождения из хаоса некоторых устойчивых упорядоченных структур с новыми свойствами. Синергетика помогает формированию нового синтеза технических наук, философии

и истории. на основе интуиции, образного восприятия, спонтанности мышления, нелинейности, поливероятности, неравновесности и бифуркационности. Она позволяет соединять два взаимодополнительных способа постижения мира через образы и через число. Такая наука содержит модель саморазвития человека в самоорганизующемся мире, причем миропорядок в ней многослойный, иерархический и целостный.

Существуют два фундаментальных свойства синергетических систем [70]: обмен с внешней средой и взаимодействие компонентов системы.

Известно, что для ЕС свойственно наличие некоторых областей притяжения — инвариантных многообразий в их пространстве состояний. Такие режимы называются аттракторами. Аттрактор — это притягивающее множество в пространстве состояний, т. е. асимптотически устойчивое множество. Аттракторы, отличные от состояний равновесия, называются «странные аттракторы». Внутри них траектории блуждают нерегулярным образом и весьма чувствительны к изменению начальных условий [71–73]. В ЕС качество их функционирования может даже повышаться при расширении разнообразия входящих в них подсистем. Перспективна попытка переноса этих свойств на ИС. Для ЕС основная цель функционирования состоит в стабилизации соотношений между их переменными состояниями. Возникновение новой структуры может проходить по разным траекториям.

Приведем стандартную универсальную схему эволюционного процесса для ИС. На начальном этапе развития происходит медленное изменение свойств ИС. Этот этап предсказуем до случайных флуктуаций. В какой-то момент времени ранее стабильное состояние теряет устойчивость и возникает возможность разных путей развития. Это и есть точка бифуркации. Среди различных ветвей эволюции есть траектория (аттрактор), которая отличается относительной устойчивостью и как бы притягивает к себе множество траекторий систем с разными состояниями. Если система попадает в этот коридор, то она неизбежно эволюционирует к этому относительно устойчивому состоянию.

В [70] считается, что аттрактор — это асимптотический предел ($\tau \rightarrow \infty$) решений, на который не оказывают прямого влияния начальные условия, причем если законы сохранения допускают несколько равновесных состояний (решений), то реализуется состояние движения, которому отвечает минимальный рост энтропии.

Существуют устойчивые образцы (схемы), в соответствии с которыми происходят все изменения в многообразном мире. Приемы переноса знаний используют аналоги и прототипы как логические алгоритмы описания законов самоорганизации. Единство мира и изоморфизм — важнейшая характеристика при построении интеллектуальной ИС. В процессе создания интеллектуальной ИС важная проблема — поиск изоморфизма (подобия), который есть следствие общих свойств систем разной природы.

Синергетика определяет три этапа синтеза интеллектуальной ИС [75]:

- структурный — определяются процессы и концепции построения упорядоченных интеллектуальных ИС из хаоса;

- модельный — создание целостной модели интеллектуальной ИС;
- иерархический — выработка представлений об интеллектуальной ИС как о составляющей процесса самоорганизации.

Теория самоорганизации объединяет в себе бифуркацию, флуктуацию, гомеостаз, энтропию, равновесие, неравновесие и эволюцию. Ключевыми понятиями теории самоорганизации ЕС и ИС (составляющими ее суть) являются хаос и порядок. Понятие «порядок» связывают с математическим отношением элементов множеств, графов и гиперграфов. Порядок содержит в себе хаос. Хаос, как предполагается, — то первичное состояние ЕС, которое предшествует ее упорядочению.

Всякая микроструктура когда-то возникла, она содержит в себе элемент нестабильности (хаоса) и поэтому должна поддерживаться. Хаос — есть бесформенное состояние ЕС, где все будущие потенции смешаны и не расчленены. В недрах хаоса заключены и начинают развиваться с началом эволюции два универсальных импульса бытия ЯН и ИНЬ [80]. Существует большое число определений и понятий хаоса. Приведем основные из этих определений:

- хаос — принцип становления, содержит единство противоположностей;
- хаос — первозданное, беспорядочное состояние элементов, но ему присуще и творчески оформляющее начало;
- хаос — принцип устойчивости и неизменности.

Согласно учениям древности [51–56, 80] начало порядка — это внесение «умом» предела в беспредельное. Все беспредельное, описываемое неопределенными характеристиками «больше», «меньше», не имеет конца, а значит, количества, числового исчисления. Два члена не могут быть сопряжены без третьего. Эту задачу выполняет пропорция золотого сечения. Здесь при любом среднем числе первое так относится к среднему, как среднее к последнему. Возникновение гармоничного порядка и всякой определенности происходит из «беспредельного, связанного пределом». Это есть согласие противоположных начал, определяемое числом. Математические модели помогают описывать хаотический и беспорядочный мир и переходить к миру вечного бытия, где царят порядок, гармония, симметрия. Модель ЕС, как замкнутого структурного и иерархического целого, может быть представлена как первая и основная форма рождения мира из хаоса [51–56].

Порядок создается из хаоса через операцию дифференциации исходных элементов, установление базовых оппозиций (активное–пассивное) и нахождение гармоничных пропорций и соотношений между ними. Порядок связан с мерным отношением элементов (исчисляемым математически). Построение порядка сопряжено с деструкцией и временной потерей гармонии.

Отметим, что важнейшим в развитии интеллектуальных ИС является принцип целого, или принцип синтезированной троицы, на любом иерархическом уровне, что следует из учений древности и современных исследований.

2.4. Гомеостаз

Постоянство внутренней среды системы называют гомеостазом (греч. *homoios* — подобный, одинаковый и *stasis* — состояние). Гомеостаз играет фундаментальную роль в эволюции ЕС и ИС, так как он определяет выживание системы, ее устойчивость в условиях непрерывных воздействий со стороны внешней среды. Существование любой системы возможно только на относительно небольшом диапазоне отклонений различных характеристик внутренней среды от средних значений. Под действием факторов внешней среды жизненно важные характеристики внутренней среды могут изменяться. Тогда в системе возникают реакции, направленные на их восстановление или предотвращение таких изменений. Эти реакции называются гомеостатическими.

Гомеостатические реакции организует управляющая система, которая регулирует активность внутренней среды. Механизмы одной и той же гомеостатической реакции и их эффективность могут быть различными и зависят от множества факторов, в том числе наследственных. Гомеостазом называют также сохранение постоянства видового состава и числа особей в сообществе (популяции), способность популяции поддерживать динамическое равновесие генетического состава, что обеспечивает ее максимальную жизнеспособность (генетический гомеостаз).

Гомеостатика изучает механизмы поддержания динамического постоянства неизменно важных параметров, функций, ритмов и этапов эволюционного развития. Гомеостаз — механизм поддержания постоянства, а гомеостатика — теория гармонии и дисгармонии [8, 67, 68, 81, 82].

В гомеостатике сосредоточены четыре составляющие: внутренние противоречия, иерархическая организация, иерархия гомеостатов, реализация в управлении принципа регулируемого противоречия.

Следуя В. Вернадскому [8], можно сказать, что важнейшую роль в построении ИС и ЕС играют гомеостатические принципы. «Независимо от природы систем они имеют определенную общность в своей организации и в механизмах управления». Использование идей гомеостатики позволяет моделировать процесс нарушения равновесия и его восстановления. Для разрешения возникающих противоречий целесообразно введение механизма с обратными связями по информации и управлению. Он позволит моделировать условия, необходимые для устойчивого функционирования системы, содержащей неустойчивые компоненты.

ЕС стремится сохранить свою стабильность — гомеостаз. При различных внешних условиях ЕС стремится не выходить из той области, которая обеспечивает возможность продолжения ее существования. В гомеостатике существует принцип доминанты: всякая деятельность ЕС обладает определенной устойчивостью, инерционностью и на слабый «побочный» стимул она отвечает компенсирующей реакцией, усиливая основную деятельность. Однако сильный побочный стимул вызывает срыв регуляции и переключение на новую деятельность.

Согласно [81, 82], между ЕС и внешней средой имеется граница, разделяющая две среды (мембрана). Хаосу и неупорядоченности внешней среды противопоставляется упорядоченность и организованность, т. е. низкая энтропия. Мембрана дает возможность взаимодействия ЕС с внешней средой, т. е. получать из нее вещество, энергию и информацию. В этой связи можно выделить два важных момента эволюции:

- стремление к гомеостазу; здесь максимизация информации становится возможной за счет уменьшения условной энтропии;
- максимизация информации за счет наибольшего разнообразия множества реакций ЕС и ИС и увеличения диапазона их действий.

Перенос управления на прежде неуправляемый набор параметров требует создания новой системы управления, которая характеризуется новым набором условий. Над первой управляющей системой надстраивается вторая, управляющая ею и т. д. Повторяясь многократно в процессе эволюции, такое построение ведет к созданию новых более совершенных интеллектуальных ИС с новым набором параметров, по которым производится управление. Тогда развитие обеспечивает каждому следующему поколению интеллектуальных ИС все большую свободу по отношению к окружающей среде, увеличивает разнообразие ее реакций.

Все системы в процессе эволюции стремятся к максимизации информации. Для этого необходимо:

- усложнять внутреннюю структуру интеллектуальных ИС;
- максимально увеличивать разнообразие реакций;
- сохранять и защищать структуры от внешних воздействий, сводя этот процесс к поддержке гомеостаза.

Отметим, что при построении интеллектуальных ИС необходима разработка гомеостатической модели и БЗ. Гомеостат определяется как универсальная информационная единица управления, инвариантная материальному носителю. Разработка информационных механизмов управления в гомеостатике базируется на следующих концепциях:

- противоречия;
- склеивания противоположностей;
- компенсационного гомеостата.

Гомеостатика анализирует информационную сущность механизма поддержания равновесия в интеллектуальных ИС. Фундаментальной концепцией гомеостатики является теория об организации взаимодействия частей любого объекта. В ее основе лежит представление о гомеостате как о системе из противоположностей (антагонизмов). Основными постулатами этой теории являются следующие:

- два противоположных элемента, а также их комбинации могут быть объединены в устойчивую интеллектуальную ИС, если выполняются необходимые и достаточные условия их объединения;
- необходимым условием объединения является «зеркальное» отображение противоположных элементов;

- достаточными условиями объединения является выполнение трех требований:
 1. Несимметрия заданий антагонизмам не должна превышать определенного предела.
 2. Несимметрия параметров, заданных антагонизмам, не должна превышать определенного предела.
 3. Степень неустойчивости потенциальных антагонистов не должна превышать определенного критического значения.

Структурным основанием гомеостата является организация информационных потоков при управлении процессами в интеллектуальных ИС. Эту структуру образуют три контура управления (рис. 2.6). Здесь при принятии решений один контур (система управления верхнего уровня) управляет целями в двух других контурах (системах управления нижнего уровня), имеющих общий объект управления. Цели поддержания гомеостата обеспечиваются согласованием процессов в этих контурах управления.

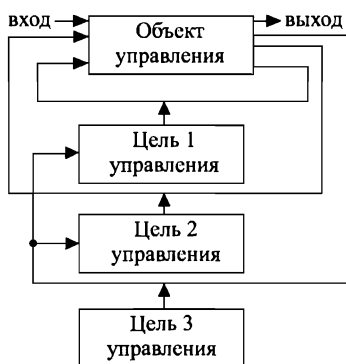


Рис. 2.6. Пример трехуровневой системы управления

Гомеостатические объекты воплощают принцип полярности. Это противоречие содержится в целях управления контуров нижних уровней. В гомеостатах этот принцип проявляется как распределенность противоречивых целей управления.

В гомеостатах третий контур отвечает за гомеостаз. Цель управления в нем — поддержание постоянства внутреннего противоречия. Способ осуществления поиска решения — это перераспределение целей в нижележащих системах. Этим поддерживается постоянное в изменяющемся, сохраняя существование целого. Другим важным принципом в гомеостатах является принцип триады.

Динамика процесса в гомеостатах отражает важные эволюционные принципы: движения, вибраций или ритма. Согласно этим принципам эволюция всякого естественного процесса разворачивается как непрерывная цепь вложенных друг в друга вибраций. Элементарным гомеостатом в человеческом организме является живая клетка.

Гомеостатика утверждает своими постулатами иерархию гомеостатических процессов. Фундаментальные принципы [8, 14, 81, 82] в гомеостатике следующие: противоречия; гармонии; аналогии.

В треугольнике контуров управления на рис. 2.6 гомеостаз поддерживается одним из них. Этим и создается иерархия. Принцип равновесия — это частный принцип гармонии. Динамическое постоянство достигается уравниванием противоположностей. В динамике равновесия возникает гармония. Выход состояний наблюдаемого процесса за пределы гармоничного отношения — это нарушение гомеостаза.

На основе биполярной модели логики ИНЬ–ЯН точка x на прямой никогда не достигнет предельного абсолютного состояния ЯН = 1 или ИНЬ = 0. Наличие в системе одновременно двух противоположных тенденций, одна из которых нарастает, а другая убывает, предполагает наличие зоны оптимальных состояний. Границы зоны определяются на основе золотого сечения. В соответствии с ним точка x делит отрезок, равный $[1;0]$, на две неравные части в отношении $0,382 : 0,618 = 0,618$ или $0,618 : 0,382 = 1,618$. Зона опасности определяется как $0,5 - 0,382 = 0,118$ и $0,118/2 = 0,059$. Величины $0,059$ на модели логики ИНЬ–ЯН определяют пределы точки x к состояниям ЯН = 1 и ИНЬ = 0. Существуют и другие конструкции, когда моделью логики ИНЬ–ЯН является полный граф на произвольное число вершин.

Следуя [87], рассмотрим реализацию нечетких отношений в гомеостатике. Пусть интеллектуальная ИС состоит из двух частей А и В.

Союзнические отношения. Выходные эффекты А и В объединяются так:

$$\tilde{\varphi}_1 = K_{\text{л}} \left(Y_{\text{А}} \cdot \tilde{\Phi}_{\text{А}} \bigcup Y_{\text{В}} \cdot \tilde{\Phi}_{\text{В}} \right), \quad 0 \leq K_{\text{л}} \leq 1,$$

где $K_{\text{л}}$ — коэффициент аддитивности, согласованности и союзнических взаимодействий; $\tilde{\varphi}_1$ — расплывчатое союзническое отношение; $Y_{\text{А}}, Y_{\text{В}}$ — области заданий расплывчатых отношений; $\tilde{\Phi}_{\text{А}}, \tilde{\Phi}_{\text{В}}$ — графики расплывчатых отношений.

Партнерские отношения:

$$\tilde{\varphi}_2 = K_{\text{м}}(Y_{\text{А}} \cdot Y_{\text{В}}), \quad K_{\text{м}} \geq 1,$$

где $K_{\text{м}}$ — коэффициент мультипликативности и партнерского взаимодействия.

Конкурентные отношения:

$$\tilde{\varphi}_3 = K_{\text{ж}} \left(Y_{\text{А}} \cdot \tilde{\Phi}_{\text{А}} \bigcap Y_{\text{В}} \cdot \tilde{\Phi}_{\text{В}} \right),$$

где $K_{\text{ж}}$ — коэффициент «жесткости» конкуренции; $K_{\text{ж}} \leq 1$, $K_{\text{ж}} = 1$ — жесткая и $K_{\text{ж}} \ll 1$ — «мягкая» конкуренция.

Предельное $\tilde{\varphi}_3$, при котором происходит нарушение системного гомеостаза, определяет отношение $\tilde{\varphi}_1$ ($\tilde{\varphi}_1 = \max \tilde{\varphi}_3$).

Конфликтные отношения. Здесь могут происходить нарушения внутренних гомеостазов А, В и системного гомеостаза ($\tilde{\varphi}_4 < \tilde{\varphi}_5$):

$$\tilde{\varphi}_5 = K_{\text{ж}} \left(Y_{\text{А}} \cdot \tilde{\Phi}_{\text{А}} \setminus Y_{\text{В}} \cdot \tilde{\Phi}_{\text{В}} \right).$$

Нейтральные отношения. В этом случае А и В не взаимодействуют:

$$\tilde{\varphi}_6 = K_{\text{ж}} \left(Y_A \cdot \tilde{\Phi}_A \bigcap Y_B \cdot \tilde{\Phi}_B \right) = \emptyset.$$

Отметим, что нейтральные отношения являются частным случаем конкурентных отношений. Все эффекты взаимодействия происходят во времени и сопровождаются двумя противоположными процессами: накопление (аккумуляция) и рассеивание (диссипация).

Для всех интеллектуальных ИС, организованных по принципу конкурентных отношений, важное значение имеют законы симметрии и гармонии. Многие интеллектуальные ИС, образованные объединением противоположностей, являются неравновесными и их существование (поддержание гомеостазов) обеспечивается за счет механизмов управления. При этом необходимо дополнение одного неустойчивого антагонизма другим, тогда образуется устойчивая гомеостатическая интеллектуальная ИС. При моделировании интеллектуальной ИС с позиции управления ее можно отображать двумя элементами: объектом, обладающим инерцией, и механизмом, стабилизирующим состояние этого объекта, либо вызывающим процессы, ведущие к потере устойчивости.

Антагонизмами считаются интеллектуальные ИС, действия которых имеют противоположно направленный характер. Для большинства интеллектуальных ИС характерна их инерционность и обратимость. Механизмы авторегуляции в диапазоне обратимости интеллектуальных ИС чаще всего реализуются по принципу интегрирования рассогласования между текущим состоянием и каким-то «оптимальным».

Анализ возможных комбинаций объединения двух антагонизмов, приведенный в [81], показывает, что для обеспечения устойчивости такой объединенной системы необходимо: зеркальное включение, когда выход каждого из них становится входом для второго; образование за счет зеркальных антагонистов двойной отрицательной обратной связи: каждый из антагонистов, в качестве которого выступает его зеркальный антагонист, охватывается отрицательной обратной связью. Выделяют четыре основных способа примитивного объединения:

- 1) компенсационный — симметрирование выходов антагонистов путем создания перекрестных информационных связей;
- 2) симметрирование по цепям обратной связи — симметрирование выходов антагонистов путем изменения их разбаланса $\Delta y = y_1 - y_2$ и передачи Δy на входы антагонистов по цепям обратной связи;
- 3) симметрирование на основе приращений — симметрирование выходов антагонистов путем организации работы на приращениях;
- 4) компенсация разбаланса — симметрирование входов антагонистов путем изменения разбаланса на входе $\Delta x = x_1 - x_2$ и использование Δx для компенсации разбаланса.

На рис. 2.7 приведена схема интеллектуальной ИС на основе модели с изменяющимися генерациями альтернативных решений. При нечетном

числе генераций $t = 1, 3, 5, \dots$ работа интеллектуальной ИС выполняется под управлением блока эволюционной адаптации с синергетическими принципами и внешней среды. При четном числе генераций $t = 2, 4, 6, \dots$ работа интеллектуальной ИС выполняется только под управлением внешней среды. Здесь P — набор входных альтернативных решений, а P' — выходных.

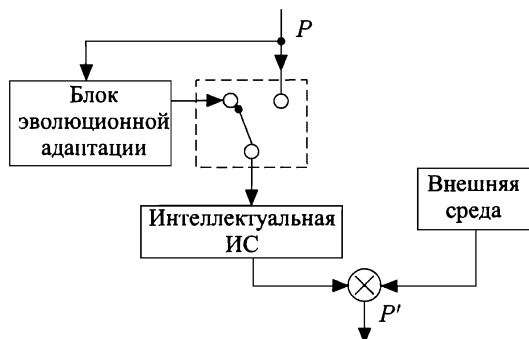


Рис. 2.7. Взаимодействия подсистем интеллектуальной ИС с изменяющимися генерациями

На рис. 2.8 приведена модель взаимодействия подсистем интеллектуальной ИС. Здесь $P_1, P_2, P_3, \dots, P_n$ — наборы входных альтернативных решений, а P' — набор выходных.

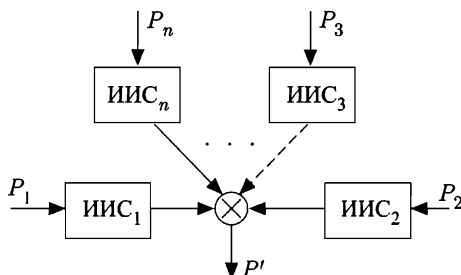


Рис. 2.8. Взаимодействие подсистем интеллектуальной ИС

На рис. 2.9 показана схема интеллектуальной ИС с относительным балансом элементов, на основе которого можно строить интеллектуальную ИС любой сложности. В рассматриваемой схеме элементы имеют различные свойства, но, будучи взаимосвязаны друг с другом, поддерживают относительный баланс. Нормальное функционирование системы (ее свойства и возможности) связано с равновесием в ней элементов ИНЬ–ЯН (блоки эволюционной адаптации 1 и 2 на рисунке). На рис. 2.9 P_1, P_2 — противоположные наборы входных решений.

Сложные способы информационного объединения согласно синергетическим принципам требуют образования иерархических структур подобно описанным выше.

типовыми процессам склеивания являются процессы расщепления. Для ЕС и ИС можно построить большое количество моделей гомеостатов.

На рис. 2.10 приведена модель интеллектуальной ИС с компенсационным гомеостатом, который реализуется в блоках обратной связи на основе блока эволюционной адаптации 1 и блока эволюционной адаптации 2. Здесь интеллектуальная ИС₁ реализуется случайным образом, а интеллектуальная ИС₂ — направленным с использованием знаний о решаемой оптимизационной задаче. В блоке W_Σ происходит объединение альтернативных решений и их редукция до стандартного размера. На рис. 2.10 P_Σ — окончательная популяция решений.

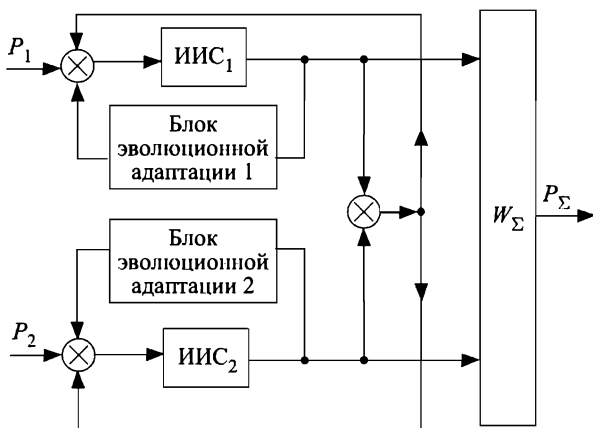


Рис. 2.11. Интеллектуальная ИС с симметрией по обратной связи

На рис. 2.11 приведена модель интеллектуальной ИС с симметрией по цепям обратной связи. Данная схема использует идею Ю. Горского о симметрировании выходов антагонистов путем измерения их разбаланса [81]. Она учитывает гомеостатические операции склеивания на информационном уровне по разбалансу на выходе.

На рис. 2.12 показана модель интеллектуальной ИС с симметрированием операций на основе приращений.

На рис. 2.13 приведена модель интеллектуальной ИС с компенсацией разбаланса на входах. Такая схема позволяет симметрировать разбаланс за счет склеивания антагонистов на информационном уровне.

Приведенные модели интеллектуальных ИС для создания многоуровневых иерархических интеллектуальных ИС во многих случаях помогают выходить из локальных оптимумов при решении оптимизационных задач.

Мерность гомеостата — это число объединяемых антагонистов. Будем использовать три основных способа построения многомерных гомеостатов:

- централизованный (склеивание двух антагонистических коалиций);
- автономизированный (склеивание n попарно объединенных антагонистов);
- планетарный (склеивание происходит с одним мощным антагонистом, представляющим собой ядро с большим числом антагонистов).

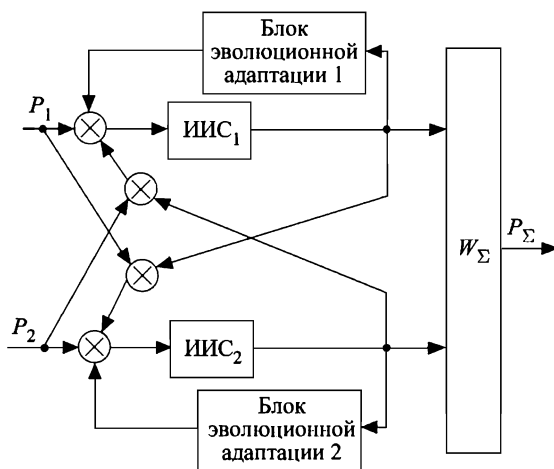


Рис. 2.12. Интеллектуальная ИС на основе приращений

В первом способе множество противоположностей разделено на две коалиции, находящиеся между собой в антагонистических отношениях. Очевидно, что чем больше устойчивых противоположностей входит в коалиции и при этом не нарушается симметрия, тем устойчивей многомерный гомеостат.

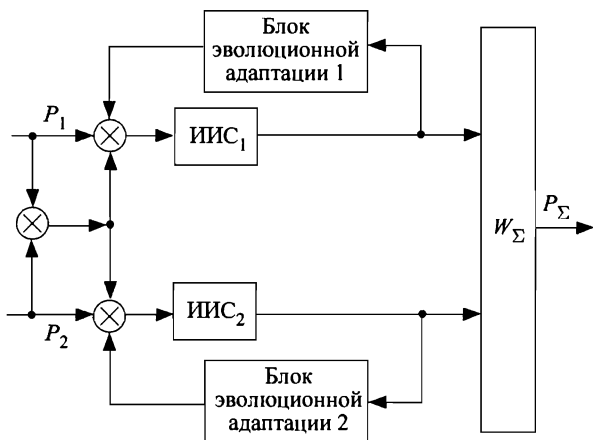


Рис. 2.13. Интеллектуальная ИС с разбалансом на входах

Во втором способе автономизированный n -мерный гомеостат состоит из нескольких гомеостатов, объединенных между собой посредством координационного управления (настройки). Чем больше количество устойчивых антагонистов входит в многомерный гомеостат, тем большей устойчивостью он будет обладать, а механизм сборки с управляющей настройкой придает ему ряд новых качеств. Отметим, что модели организации интеллектуальных ИС в упрощенном виде могут быть сведены к иерархиям

многомерных автономизированных гомеостатов, которые отражают интегродифференциальные принципы управления.

В третьем способе выделяется один мощный и устойчивый антагонист (центр) и $(n - 1)$ других неустойчивых антагонистов, отражающих работу других равноправных членов гомеостатического коллектива, направленность которых противоположна центру. Динамика интеллектуальных ИС реализуется на основе принципа равновесия или принципа гармонии. Поэтому взаимодействие входов и выходов оказывается динамически постоянным (гомеостатичным). В динамике равновесия возникает гармония. Гармонию определяют как отношение противоположностей, а в количественных мерах — как «золотое сечение» [81–83].

В управлении интеллектуальных ИС в качестве ЦФ иногда используют критерии Парето, критерии на основе чисел Фибоначчи и др. Если рассмотреть крайние состояния некоторого управляемого процесса в интеллектуальных ИС, где между ЛПР и внешней средой поддерживается динамическое равновесие, то областью гармоничных состояний будет та, которая образуется границами золотого сечения, отстоящими от входов и выходов. Гармония в таком выражении есть не число, а область или пространство. Следствием совместного действия каких-либо начал в управлении интеллектуальных ИС является принцип причинности. По этому принципу каждое состояние процесса эволюции проявляется как причина последующих и следствие предшествующих состояний — создается связанность всего со всем. Этот принцип — как бы надстройка по отношению к принципу наследственности. Наследственность в системном отношении проявляется в построении упорядоченности из хаоса развивающихся интеллектуальных ИС. Новые, более развитые интеллектуальные ИС всегда содержат в качестве элементов или в качестве формирующих принципов характерные черты систем менее развитых уровней.

Конструкцией целого выражается идея любого объекта, явления, процесса как составного целого. Основа этой идеи состоит в том, что полюса в любой интеллектуальной ИС, выражая единство его противоположных свойств, объективно притягиваются друг к другу. В этом притяжении определяется некоторая общая точка — точка синтеза. Она может рассматриваться либо как исток обоих полюсов, т. е. как причина их образования, либо как результат совместного проявления. Согласно этой идее рассмотрение любой интеллектуальной ИС, явления, либо процесса как целого приводит к тому, что в нем должны быть найдены три компоненты: два полюса и соединяющий их принцип (отношение), аналогично отношению между вершинами в треугольнике Фреге.

Развитие эволюции иногда представляют в виде совокупности объективных этапов, образующих один заверченный цикл. При этом происходит возникновение ряда отдельных изменений в интеллектуальных ИС под влиянием внешних и внутренних стимулов. Это — спонтанные мутации или пробные эксперименты. Кроме этого выполняется анализ результатов изменений на согласованность с внешней средой и происходит отбор ва-

риантов в соответствии с принципом гармонии. Остаются те варианты интеллектуальных ИС, которые соответствуют отношениям главных внешних и внутренних факторов, определяющих их развитие. ЛПР может выбрать ошибочный вариант развития интеллектуальной ИС. При этом эволюция исправляет ошибки. Негармоничные решения устраняются. При размножении изменений эволюция осуществляет самовоспроизведение: хромосомы делятся, размножаются спариванием, т. е. комбинированием признаков, а интеллектуальные ИС совершенствуются и дорабатываются. При расширении пространства создаются условия для механизма совершенствования интеллектуальных ИС. Совершенствуются отдельные свойства качественно изменившихся интеллектуальных ИС. В заключение происходит закрепление изменений, остановка преобразований и создание предпосылок для нового витка эволюции.

Принципы гомеостатики и иерархии проявляются и в управлении интеллектуальными ИС. При этом контур управления информационной цепи обратной связи руководит объектом управления. Адаптивная подсистема главенствует над основным контуром управления интеллектуальной ИС и т. д. Структура простейшей интеллектуальной ИС основывается на взаимосвязи информации нескольких видов: отображающей, внутренней, управляющей.

Внутренняя информация $I_{ин}$ образует совокупность сведений, знаний об интеллектуальной ИС, среде и цели. Следовательно, внутренняя информация интеллектуальной ИС определяет ее целенаправленное поведение. Отображающая информация $I_{от}$ — это сведения о внешней среде и об интеллектуальной ИС, характеризующие их в каждой конкретной ситуации. Следовательно, это ситуационная информация.

Количества $I_{ин}$ и $I_{от}$ должны быть соизмеримы, чтобы $I_{от}$ имела смысл. Управляющая информация $I_{упр}$ — совокупность сведений, передаваемых от управляющей подсистемы интеллектуальной ИС к объекту управления, и влияющих на его поведение. Она зависит не только от характера передаваемых сведений, но и от свойства объекта управления. Процесс управления в интеллектуальной ИС для установления гармонии и гомеостаза на информационном уровне может быть представлен в виде непрерывного потребления информации, т. е. приобретения $I_{от}$, ее передачи, формирования и передачи $I_{упр}$ [82]. Тогда интеллектуальную ИС можно также определить как динамическую систему, находящуюся под управлением. Она задается шестеркой $\langle T, Z, V, W, X, Y \rangle$, где T — множество дискретных моментов времени, Z — множество состояний динамической интеллектуальной ИС, V — множество мгновенных входных управлений, W — множество мгновенных входных возмущений, X — множество входов интеллектуальной ИС, Y — множество выходов.

Эти множества элементов связаны различными отношениями. Структура цели управления образуется на множестве $A \subseteq Z$ — множестве сведений о целевых состояниях интеллектуальной ИС, т. е. о тех состояниях, при которых достигается необходимый результат управления. Кроме того, для интеллектуальной ИС должно быть известно соответствие $\Gamma \subseteq T \times A$ как

некоторая стратегия перехода от одного допустимого состояния к другому. Образующие при этом сведения сопровождают динамику интеллектуальной ИС и свидетельствуют о ее переходе из одного состояния $z_i \in Z$ в другое $z_j \in Z$ под воздействием управления $U: T \rightarrow V$ и возмущения $m: T \rightarrow W$.

Получаемые сведения о функции $y_\sigma: \sigma \rightarrow y$, где $\sigma \in T$, используются для определения текущего состояния интеллектуальной ИС $z(t)$. Следовательно, $I_{\text{вн}}$ в интеллектуальной ИС представляет собой полученную из хаоса упорядоченную совокупность разнообразной информации. Этот порядок и создает в интеллектуальной ИС возможность анализа поведения управляемого объекта и установления такого поведения, которое требуется для достижения гармонии и гомеостаза. Если возникают отклонения от заданных свойств объекта управления, среды и интеллектуальной ИС, то управление становится невозможным, так как информации для принятия решений недостаточно. Возникает противоречие между существующими и необходимыми возможностями системы управления. Оно устраняется созданием механизма адаптации. В результате внешних и внутренних изменений в информационной структуре интеллектуальной ИС при управлении могут возникать неопределенности, относящиеся к любому виду сведений, составляющих внутреннюю информацию системы.

Устранение любой неопределенности в системе управления интеллектуальной ИС осуществляется с помощью другой системы управления, образующей по отношению к первой дополнительный адаптивный контур управления. В [82] выделяют два механизма принятия решений в условиях неопределенности — рефлексивный и нерефлексивный. Нерефлексивный способ устранения неопределенностей представляют как неуправляемое дополнение недостающих сведений из хранилища данных.

В интеллектуальных ИС в основном используется рефлексивный способ. Он заключается в анализе изменений и выработке решений на коррекцию информационного пространства в системе. При таком способе управление осуществляется за счет дополнительного контура, т. е. дополнительной обратной рекурсивной цепи. Эта дополнительная рефлексия обеспечивает устранение неопределенностей за счет нового информационного ресурса, который появляется в процессе принятия решений. В таком дополнительном контуре объектом управления становится некоторый информационный процесс в основном контуре. Управлению такой системой подвергается процесс, в котором возникла неопределенность.

Таким образом, образуется новая система для принятия решений, в которой объектом управления является старая система. Информационная структура дополнительного контура аналогична основному. Отличие состоит в содержании сведений, образующих информационную структуру. Возможны ситуации, когда априорная неопределенность и нечеткость возникает по отношению к данным в дополнительных контурах. В этих случаях каждый из этих контуров для лучшей адаптации тоже становится управляемым. В результате образуется структура с двумя и более иерархически связанными дополнительными адаптивными подсистемами.

В функциональном отношении такие интеллектуальные ИС отличаются способностью поддержания баланса, равновесия, т. е. гомеостаза. При этом в системе поддерживается динамическое постоянство основных функций и параметров интеллектуальных ИС при различных изменениях внутренней и внешней сред. Адаптационные способности гомеостатических систем достигаются за счет реализованного в них управляемого противоречия между целями управления контуров нижнего уровня.

Следуя [62–64, 81–83], простейшая организация информационных потоков в виде контура обратной связи, при которой в интеллектуальных ИС появляется возможность «понимать» текущую ситуацию, образует элементарный интеллект в системе управления. Добавление адаптивных подсистем к основному контуру управления позволяет распознавать все более сложные ситуации с изменяющимися свойствами. Это и есть процесс интеллектуализации ИС.

В [82] введено множество уровней рефлексии I_r , которыми могут обладать системы управления, реализующие ту или иную функцию, и множество взаимозависимых функций $F = F_1 \cup F_2 \cup \dots \cup F_k$, которые должны реализовываться в системе. Пересечение этих множеств описывает процессы принятия решений в системах управления. Каждый элемент декартова произведения $R \subseteq F \times I_r$ определяет уровень интеллектуальности системы. Глубину интеллекта в ИС определяют максимальным рангом рефлексии в них. Широту интеллекта определяют как суммарный размер хранилища данных некоторой фиксированной глубины. С каждой новой задачей добавляется новое знание, и объем общего интеллекта растет. Повышение уровня рефлексии в интеллектуальных ИС сопровождается адаптацией — усилением процесса приспособления интеллектуальных ИС к условиям внешней среды. После каждого нового шага интеллектуальная ИС становится способной реализовывать функции с учетом изменений; изменения изменений; изменения изменений и т. д.

В гомеостатических интеллектуальных ИС выделяют два фундаментальных свойства: полярность и иерархичность. В таких интеллектуальных ИС одним и тем же объектом управляют две системы с различными целями. Это — основа интеллектуальных ИС или элементарный гомеостаз. Как видно, здесь снова реализуется триединый подход на нижнем иерархическом уровне, причем цели управления в контурах нижнего уровня соотносятся как противоположные. Различие целей управления проявляется распределенностью противоречия в гомеостазе. В этой связи объекту управления удастся поддерживать постоянство процессов перераспределением заданий каждому из этих контуров управления при изменении внешней и внутренней среды.

Внутреннее противоречие между целями придает гомеостазу признаки целостности. Пересечение областей допустимости образует область, где противоречия как бы совпадают между целевыми функциями. При этом и возникает разнообразие способов поддержания гомеостаза. Принципы полярности и иерархичности задают устойчивость непрерывно изменяющимся

во взаимодействии полюсам интеллектуальной ИС. Этим поддерживается постоянство в изменяющемся, сохраняя существование целого.

Отметим, что для поддержания гомеостаза можно создать множество и множество множеств адаптивных подсистем вокруг основного контура управления. Тогда образуется устойчивая целостность, способная поддерживать постоянство процессов в объекте управления при существенных изменениях внешней среды. При этом, усложнив систему и организовав адаптацию гомеостатической интеллектуальной ИС по принципу многостажного гомеостаза, можно определенным образом моделировать процессы, происходящие в ней для поддержания устойчивости. Модель интеллектуальной ИС в какой-то мере аналогична геометрической модели функционального состояния человека, которая обладает свойствами гармонии. Она открывает путь к использованию общих принципов симметрии для анализа геометрических моделей функциональных состояний с целью выявления их инвариантов как количественной меры сохранения интеллектуальной ИС.

2.5. Фракталы

Все интеллектуальные ИС строятся на некой общей основе. Существует особый архитектурный каркас, некий строительный блок, для параметрических интеллектуальных ИС. Он затем достраивается, входит в сложные конгломераты конечным числом способов по нескольким определенным алгоритмам. Сложные интеллектуальные ИС имеют фрактальное строение из универсальных строительных блоков, которые повторяются в различных масштабах.

Итак, интеллектуальная ИС имеет внутренние предпочтения к определенным формам. Другие формы неустойчивы и очень быстро эволюционируют к устойчивым формам. Результаты синергетики возвращают нас к идеям древних учений о потенциальном и направленном. На упрощенных математических моделях в виде графов и гиперграфов можно видеть все поле возможных путей эволюции, все возможные пути развития данной интеллектуальной ИС. Интеллектуальные ИС имеют множество путей эволюции. Отсюда разнообразие форм. Однако реализуется одна из форм. Строительство на основе строительных блоков по образцу — это матричное дублирование. Путь отбора через хаос — это медленный путь эволюции, путь случайных вариаций и эволюционного отбора, постепенного перехода от простых интеллектуальных ИС к все более сложным. Универсальные строительные блоки имеют фрактальное строение и повторяются в различных масштабах.

В 1980 году Б. Мандельброт указал на фрактальную геометрию ЕС. Установлено, что фракталы выполняют роль информаторов, сигнализирующих о состоянии неустойчивости системы. Информационные свойства фракталов близки к свойствам живой клетки. Они инвариантны к анализируемому объекту, способны к самоподобному размножению на различных пространственно-временных уровнях и передаче информации системе о нарушении устойчивости своего структурного состояния; обладают

свойствами адаптации к внешнему воздействию путем самоперестройки фрактального множества в точках неустойчивости структуры системы. Для интеллектуальных ИС самоподобие структур, являющееся определяющим свойством фрактального множества, реализуется лишь в ограниченных масштабах. Тогда говорят о мультифракталах как о фрактальном множестве, содержащем фрактальные подмножества, связанные степенным законом. Они характеризуются не только размерностью, но и степенью однородности рассматриваемого множества. Для этого вводится вероятностная размерность фрактального множества.

Для моделирования самоорганизации интеллектуальных ИС типа фрактального множества перспективно использовать принцип подчинения, когда при переходе интеллектуальной ИС через точку бифуркации множество переменных подчиняется одной или нескольким переменным, называемым параметром порядка. Согласно [84–87], эволюционирующие системы имеют фрактальную природу, направленный нестихийный отбор и самосогласованную эволюцию.

Фрактальные объекты самоподобны, то есть, их вид не претерпевает существенных изменений при изменении масштабов их деятельности. Множества, имеющие такую структуру, считаются обладающими геометрической (масштабной) универсальностью. Преобразования, создающие такие структуры, — это процессы с обратной связью с большим числом итераций, когда одна и та же операция выполняется снова и снова. Здесь результат одной итерации является начальным условием для другой и требуется нелинейная зависимость между результатом и реальным значением, т. е. динамический закон $x_{k+1} = f(x_k)$. На рисунке 2.14 приведена схема реализации этого процесса.

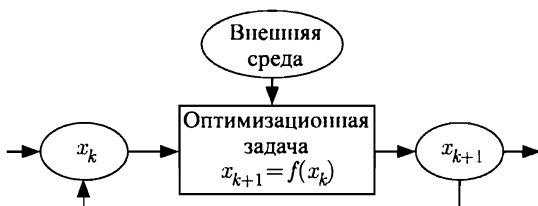


Рис. 2.14. Схема динамического процесса

Такой процесс аналогичен идеям Н. Винера о системах с обратной связью и идеям эволюции [14, 15]. Здесь входной параметр может быть постоянным, а может коррелировать с внешней средой. Результатом работы такого блока будет выходная последовательность $x_1, x_2, \dots, x_n$. Отметим, что здесь существует нелинейная зависимость между результатом и начальными условиями.

К фрактальным множествам относят множество Кантора и ковер Серпинского [11, 84–87]. Они обладают геометрической инвариантностью и называются «множества средних третей».

Простейший вариант множества Кантора строится следующим образом. Рассмотрим отрезок единичной длины $[0, 1]$ на вещественной оси.

Он делится на три равные части и средняя из них — открытый интервал $(1/3, 2/3)$ вырезается (рис. 2.15). Далее все происходит аналогично с каждым оставшимся из отрезков. Получаем последовательность отрезков все убывающей длины. На первом этапе — это один отрезок, на втором — два, на третьем — 4 и т. д., на k -м — $2k$. При $k \rightarrow \infty$ имеем множество точек, называемое множеством Кантора. Суммарная длина всех вырезанных отрезков равна 1.

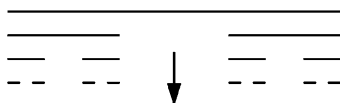


Рис. 2.15. Построение множества Кантора

Обобщение множества Кантора средних третей на случай плоских фигур приводит к ковру Серпинского. Например, возьмем квадратную матрицу и разделим ее на девять равных квадратов. При первой итерации удаляем центральный квадрат, аналогично поступим с каждым из оставшихся восьми квадратов и т. д. Пересечение полученных при $k \rightarrow \infty$ множеств — это ковер Серпинского [11]. Множество Кантора является фракталом. Для фракталов введено понятие фрактальной размерности. Для множества Кантора, которое состоит из $N = 2n$, разделенных интервалов длиной $\varepsilon = (1/3)^n$, фрактальная размерность равна:

$$\text{ФРР} = \frac{n \ln 2}{n \ln 3} \approx 0.63.$$

Для ковра Серпинского имеем фрактальную размерность $\approx 1,893$.

Примером фрактального объекта может быть схема снежинки. Если взять равносторонний треугольник и разделить каждую из его сторон на три части и по каждой из трех центральных третей построить по равностороннему треугольнику меньших размеров, то получим фрактальный объект, размерность которого есть фрактальная размерность $\approx 1,26$. Такой фрактальный объект иногда называют кривой Коха [84]. Отметим, что на основе множества Кантора и ковра Серпинского можно получать любые строительные блоки в интеллектуальных ИС.

Основные формы кооперативного поведения, свойственные ЕС, имеют свои аналогии и среди ИС. Хаотические динамические системы чрезвычайно чувствительны к внешним воздействиям. Стабилизация хаотического поведения осуществляется двумя способами. Первый обеспечивает выведение системы из хаотического на регулярный режим посредством внешних воздействий, реализованных без обратных связей. Второй способ называется контролируемым хаосом с обратной связью. Создание интеллектуальных ИС на основе применения хаотических систем базируется на использовании внутренней структуры управления. Хаотические множества, как правило, содержат бесконечное подмножество неустойчивых предельных циклов. Фракталами, как отмечалось выше, обычно называют множества, которые обладают масштабной инвариантностью, т. е. в любом масштабе они выглядят практически одинаково. В частности, странные аттракторы могут обладать геометрической инвариантностью, т. е. подобно фрактальным множествам их структура повторяется при последовательном увеличении масштаба.

Существует механизм агрегации (DLA), описывающий создание фракталов. Согласно DLA, определенная разновидность фракталов может быть получена в процессе неупорядоченного роста. Например, задан кластер (объект с максимальным числом внутренних связей), растущий следующим образом: с течением времени к нему присоединяется молекула и сразу прилипает к нему. Этот процесс называется агрегацией. Пусть частицы диффундируют к растущему кластеру абсолютно случайным образом. Агрегация частиц, протекающая в условиях случайного движения, это и есть DLA [64–65].

Приведем один из возможных механизмов решения оптимизационной задачи на основе агрегации фракталов. Он основан на четырех принципах:

- построение кластеров (массивов, определенным образом связанных между собой исследуемых объектов);
- факторизации кластеров, т. е. уменьшение размерности задачи путем представления кластеров в виде новых объединенных кластеров;
- DLA, т. е. агрегация, протекающая в условиях случайного, направленного и комбинированного присоединения элементов к кластерам;
- использование отдельных и объединенных моделей эволюций Ч. Дарвина, Ж. Ламарка, де Фриза, К. Поппера и синтетической теории эволюции.

Таким образом, между принципами гомеостатки, синергетики и эволюции много общего. Это как бы звенья одного общего эволюционного процесса развития ЕС и ИС. Общей концепцией интеллектуальных ИС может служить объединенное учение о гомеостазе, адаптации, самоорганизации, фракталах. Интеллектуальная ИС может определяться как регулирующая, адаптивная самообучающаяся система, проявляющая свойства гомеостатичности и регулируемости.

ГЛАВА 3

ГЕНЕТИЧЕСКИЕ АЛГОРИТМЫ

Научное мировоззрение, проникнутое
естествознанием и математикой, есть
величайшая сила не только настоящего,
но и будущего.
В. Вернадский

3.1. Определение и структура генетических алгоритмов

Исторически считается, что научное направление «эволюционные вычисления» является основным, использующим (моделирующим) эволюционные механизмы живой природы. Оно включает в себя три главных направления фундаментальных исследований: генетические алгоритмы, эволюционное моделирование (в некоторых работах говорят об эволюционных стратегиях) и эволюционное программирование [16–19, 23–25, 88–98].

В своих работах И. Букатова [93, 94] сформулировала новое направление «эвоинформатика». Эвоинформатика определяется как совокупность алгоритмических, программных и аппаратных средств, основанных на имитации механизмов ЕС для синтеза структур обработки информации. И. Букатова предложила архитектуру эволюционного поиска, состоящего из следующих компонент: моделирование цели, имитация основных законов дарвинизма, соревнование, отбор. Основу алгоритмов эволюционного и структурного поиска здесь составляют самообучающиеся и адаптивные алгоритмы. Самоорганизующиеся вычисления основаны на распараллеливании эволюционных процессов аналогично эволюции Ч. Дарвина, происходящих на основе механизмов синергетики. Такая эволюционно-вычислительная технология включает генерацию эволюционных систем, эволюционный синтез заданных моделей, поэтапное управление и интерпретацию моделей из БЗ.

Генетические алгоритмы — это новая область исследований, которая появилась в результате работ Д. Холланда и его коллег [88–90, 99–107]. ГА, описанные Д. Холландом, заимствуют в своей терминологии многое из естественной генетики. Впервые они были применены к таким научным проблемам, как распознавание образов и оптимизация. ГА представляет собой адаптивный поисковый метод, который основан на селекции лучших элементов в популяции, подобно эволюционной теории Ч. Дарвина.

Основой для их возникновения послужили модель биологической эволюции и методы случайного поиска. В [95] отмечено, что случайный поиск возник как реализация простейшей модели эволюции, когда случайные мутации моделировались случайными шагами поиска оптимального решения, а отбор — «устранением» неудачных вариантов.

Эволюционный поиск с точки зрения преобразования информации в интеллектуальной ИС — это последовательное преобразование одного конечного нечеткого множества промежуточных решений в другое. Само преобразование можно назвать алгоритмом поиска, или генетическим алгоритмом. ГА — это не просто случайный поиск. Они эффективно используют информацию, накопленную в процессе эволюции. Цель ГА состоит в том, чтобы:

- абстрактно и формально объяснить адаптацию процессов в ЕС и интеллектуальной ИС;
- смоделировать естественные эволюционные процессы для эффективного решения оптимизационных задач науки и техники.

В настоящее время используется новая парадигма решений оптимизационных задач на основе ГА и их различных модификаций. ГА осуществляют поиск баланса между эффективностью и качеством решений за счет «выживания сильнейших альтернативных решений», в неопределенных и нечетких условиях.

ГА, согласно [88–90], отличаются от других оптимизационных и поисковых процедур следующим:

- работают в основном не с параметрами задачи, а с закодированным множеством параметров;
- осуществляют поиск не путем улучшения одного решения, а путем использования сразу нескольких альтернатив на заданном множестве решений;
- используют целевую функцию, а не ее различные приращения для оценки качества принятия решений;
- применяют не детерминированные, а вероятностные правила анализа оптимизационных задач.

Для работы ГА выбирают множество натуральных параметров оптимизационной проблемы и кодируют их в последовательность конечной длины в некотором алфавите. ГА работает до тех пор, пока не будет выполнено заданное число генераций (итераций алгоритма) или когда на некоторой генерации будет получено решение определенного качества, или когда найден локальный оптимум, т. е. возникла преждевременная сходимость и невозможно найти выход из этого состояния. В отличие от других методов оптимизации ГА, как правило, анализируют различные области пространства решений одновременно и поэтому они более приспособлены к нахождению новых областей с лучшими значениями ЦФ.

Приведем некоторые понятия и определения из теории ГА. Все ГА работают на основе начальной информации, в качестве которой выступает популяция альтернативных решений P . Популяция $P = \{p_1, p_2, \dots, p_i, \dots, p_{N_p}\}$ есть множество элементов p_i . Здесь N_p — размер популяции. Каждый элемент этой популяции P_i , как правило, представляет собой одну или несколько хромосом, или особей, или индивидуальностей (альтернативных упорядоченных или неупорядоченных решений). Хромосомы состоят из генов (элементы, части закодированного решения), и позиции генов в хромосоме называются лоци или локус для одной позиции, т. е. ген — подэле-

мент (элемент в хромосоме), локус — позиция в хромосоме, аллель — значение гена.

Гены могут иметь числовые или функциональные значения. Обычно эти числовые значения берутся из некоторого алфавита. Генетический материал элементов обычно кодируется на основе двоничного алфавита $\{0, 1\}$, хотя можно использовать буквенные, а также десятичные и другие алфавиты. Примером закодированной хромосомы длины семь на основе двоничного алфавита может служить хромосома $p_i = (0001101)$. Элементы в ГА часто называют родителями. Родители выбираются из популяции на основе заданных правил, а затем смешиваются (скрещиваются) для производства «детей» (потомков). Дети и родители создают новую популяцию. Генерация, т. е. процесс реализации одной итерации алгоритма, называется поколением.

Эволюция популяции согласно [89–91] — это чередование поколений, в которых хромосомы изменяют свои значения так, чтобы каждое новое поколение наилучшим способом приспосабливалось к внешней среде. В интеллектуальной ИС общая генетическая упаковка называется генотипом. В ЕС организм формируется посредством связи генетической упаковки с окружающей средой и называется фенотипом.

Каждый элемент в популяции имеет определенный уровень качества, который характеризуется значением ЦФ (в литературе иногда называется функция полезности или пригодности *fitness*). ЦФ используется в ГА для сравнения решений между собой и выбора из них лучших. Основная задача генетических алгоритмов состоит в оптимизации целевой функции. Другими словами, ГА анализирует популяцию хромосом, представляющих комбинацию элементов из некоторого множества, и оптимизирует ЦФ, оценивая каждую хромосому. Генетические алгоритмы манипулируют популяцией хромосом на основе механизма натуральной эволюции.

Каждая популяция обладает наследственной изменчивостью. Это означает, что имеет место случайное отклонение от наиболее вероятного среднего значения ЦФ. Отклонение описывается нормальным законом распределения случайных величин, но, в отличие от ЕС, наследственная изменчивость не затухает, как всякая флуктуация. При этом наследственные признаки закрепляются, если они имеют приспособительный характер, т. е. обеспечивают популяции лучшие условия существования и размножения.

Так же как процесс эволюции начинается с начальной популяцией, так и алгоритм начинает свою работу с создания начального множества конкурирующих между собой решений оптимизационной задачи. Затем эти «родительские» решения создают «потомков» путем случайных и направленных изменений. После этого оценивается эффективность решений и они подвергаются селекции. Как и в ЕС здесь действует принцип «выживания сильнейших», наименее приспособленные решения «погибают», а затем процесс повторяется вновь.

Традиционные оптимизационные алгоритмы для нахождения лучшего решения используют большое количество допущений при оценке ЦФ. Эволюционный же подход не требует таких допущений. Это расширяет

класс задач, которые можно решать с помощью эволюционного моделирования. Согласно существующим исследованиям можно сказать, что эволюционные методы и ГА позволяют решать те проблемы, решение которых традиционными оптимизационными алгоритмами затруднительно.

ГА дает ряд преимуществ при решении практических задач. Одно из таких преимуществ — это адаптация к изменяющейся окружающей среде. В реальной жизни проблема, которая была поставлена для решения изначально, может претерпеть огромные изменения в процессе своего решения. При использовании традиционных методов все вычисления приходится начинать заново, что приводит к большим затратам машинного времени. При эволюционном подходе популяцией служит БЗ, которую можно анализировать, дополнять и видоизменять применительно к изменяющимся условиям. Для этого не требуется полный перебор. Другое преимущество ЭМ для решения задач состоит в способности быстрой генерации достаточно хороших решений.

При решении практических задач с использованием ЭМ, необходимо выполнить следующие четыре предварительных этапа:

- выбрать способ представления решения;
- разработать операторы случайных изменений;
- определить законы выживания решения;
- создать начальную популяцию.

Рассмотрим некоторые особенности выполнения этих этапов.

Для представления решения в виде, удобном для реализации на ЭВМ, требуется такая структура, которая позволит кодировать любое возможное решение и производить его оценку. Математически доказано, что не существует идеальной структуры представления, так что для создания хорошей структуры требуется анализ, перебор и эвристические подходы. Возможный вариант представления должен позволять проведение различных перестановок в хромосомах. Необходимо также определить способ вычисления ЦФ для оценки решений.

Достаточно сложным является этап выбора случайного оператора (или операторов) для генерации потомков. Их существует огромное число. В ЕС используются два основных типа размножения: половое и бесполое. При половом размножении два родителя обмениваются генетическим материалом, который используется при создании потомка. Бесполое размножение — это фактически клонирование, при котором происходят различные мутации при передаче информации от родителя к потомку. Эти операторы очень важны при ЭМ, однако в общем случае для интеллектуальной ИС можно применить и операции, которые не существуют в ЕС. Например, использовать материал от трех или более родителей, проводить голосование при выборе родителей. Фактически нет пределов в использовании различных операторов и нет никого смысла только слепо копировать законы природы и ограничиваться ими.

Успех ЭМ во многом зависит от того, насколько хорошо взаимодействуют между собой схема представления, операторы случайных изменений и способ определения ЦФ. Поэтому для определенного класса задач более

целесообразно использовать специальные определенные для этих задач операторы. То же можно сказать и о представлении, так как не существует универсального кодирования, которое можно использовать во всех оптимизационных задачах.

В качестве примера рассмотрим два способа представления перестановок при решении оптимизационных задач. В первом случае будем использовать одного родителя и получать потомка. Во втором операторе мы используем двух родителей, случайно выберем точку перестановки и для образования потомка возьмем первый сегмент у первого родителя, а второй сегмент — у второго. Первый оператор похож на бесполое размножение, а второй оператор — на половое размножение. Стоит отметить, что если первый оператор всегда генерирует реальное решение, то второй может генерировать недопустимые решения. Но это не мешает нам его использовать, просто требуется «восстанавливать» решения перед их оценкой. Например, можно использовать замещение. Как только это сделано для каждого повторяющегося решения, потомок будет восстанавливаться и будет соответствовать реальному решению.

На третьем из рассматриваемых этапов задаются правила выживания решений для создания потомства. Так же как и со случайными операторами, существует множество способов проведения селекции. Простейшее правило — это выживание сильнейших, т. е. когда только лучшие решения выживают, а все остальные устраняются. Однако такое правило часто оказывается малоэффективным при решении сложных проблем, когда лучшие решения могут происходить от худших, а не только от самых лучших. Однако логично использовать принцип, что вероятность выживания хорошего решения должна быть выше.

Последний предварительный этап заключается в создании начальной популяции. Если у нас недостаточно знаний о проблеме, то решения могут случайным образом выбираться из всего множества возможных. Это означает генерацию случайных перестановок, где каждая перестановка представляет собой определенное решение. С другой стороны, можно использовать некоторые знания о задаче при создании начальной популяции, например, эти данные могут быть получены из опыта решения этой же задачи другими алгоритмами. Если эти решения действительно ценные, то они выживут и произведут потомство, если же нет, то они погибнут вместе с другими слабыми индивидами.

Отметим, что популяция обязательно является конечным множеством. В каждой генерации хромосомы являются результатом применения некоторых генетических операторов. В простых ГА (ПГА) Д. Гольдберга существует три основных оператора (или функции, или шага, или процесса): селекция (репродукция), кроссинговер (скрещивание) и мутация [89].

Рассмотрим кратко эти основные операторы. Натуральная селекция — это процесс, посредством которого хромосомы, имеющие более высокое значение ЦФ (с сильными признаками), получают большую возможность для репродукции, чем слабые хромосомы. Элементы, выбранные для репродукции, обмениваются генетическим материалом, создавая аналогичные или различные потомки.

Существует много различных видов селекции. Рассмотрим основные из них:

- **Селекция на основе рулетки** — это самый простой и наиболее используемый в ПГА метод. При его использовании каждому элементу в популяции соответствует зона на колесе рулетки, пропорционально соразмерная с величиной ЦФ. Тогда при повороте колеса рулетки каждый элемент имеет некоторую вероятность выбора для селекции, причем элемент с большим значением ЦФ имеет большую вероятность для выбора.
- **Селекция на основе заданной шкалы**. Здесь популяция предварительно сортируется от «лучшей» особи к «худшей» на основе заданного критерия. Каждому элементу назначается определенное число и далее селекция выполняется согласно этому числу.
- **Элитная селекция**. В этом случае выбираются лучшие (элитные) элементы на основе сравнения ЦФ. Далее они вступают в различные преобразования, после которых снова выбираются элитные элементы. Процесс идет до тех пор, пока продолжают появляться элитные элементы.
- **Турнирная селекция**. При этом некоторое число элементов (согласно размеру «турнира») выбирается случайно или направленно из популяции, и лучшие элементы в этой группе на основе заданного турнира определяются для дальнейшего генетического поиска.

Кроме описанных, существует большое число других методов селекции, которые можно условно классифицировать на три группы [19]. К первой группе отнесем вероятностные методы селекции. Ко второй — детерминированные. К третьей — различные комбинации методов из первой и второй групп. Поиски оптимальных методов селекции непрерывно продолжаются. Дело в том, что основной трудностью решения оптимизационных задач с большим количеством локальных оптимумов является предварительная сходимости алгоритмов. Другими словами, происходит попадание решения в один, далеко не самый лучший, локальный оптимум. Различные методы селекции и их модификации как раз и позволяют в некоторых случаях решать проблему предварительной сходимости алгоритмов. Следует отметить, что исследователи ГА все более склоняются к мысли применять комбинированные методы селекции с использованием предварительных знаний о решаемых задачах и предварительных результатах.

Опишем теперь методы кроссинговера. В литературе часто их называют операторами кроссинговера (ОК). Основная функция ОК — создавать хромосомы потомков на основе различного скрещивания родителей. Существует огромное число ОК, так как их структура в основном и определяет эффективность ГА. Кратко рассмотрим основные ОК, известные в литературе, и их модификации.

Одноточечный (простой) ОК. Перед началом работы одноточечного ОК определяется так называемая точка ОК, или разрезающая точка, которая обычно определяется случайно. Эта точка определяет место в двух

хромосомах, где они должны быть «разрезаны». Например, пусть популяция P состоит из двух хромосом $P = \{p_1, p_2\}$, которые выступают в качестве родителей. Пусть первый и второй родители имеют вид $p_1 : (11111)$, $p_2 : (00000)$. Выберем точку ОК между вторым и третьим генами в p_1 , p_2 . Тогда, меняя элементы после или перед точкой ОК между двумя родителями, можно создать два новых потомка. В нашем примере получим:

p_1 :	1	1		1	1	1
p_2 :	0	0		0	0	0
p'_1 :	1	1		0	0	0
p'_2 :	0	0		1	1	1

Итак, одноточечный ОК в интеллектуальной ИС выполняется в три этапа:

1. Две хромосомы $A = a_1, a_2, \dots, a_L$ и $B = a'_1, a'_2, \dots, a'_L$ выбираются случайно из текущей популяции.
2. Число k выбирается из $\{1, 2, \dots, L - 1\}$ также случайно. Здесь L — длина хромосомы, k — точка ОК (номер или значение, код гена, после которого выполняется разрез хромосомы).
3. Две новые хромосомы формируются из A и B путем перестановок элементов согласно правилу

$$\begin{aligned} A' &= a_1, a_2, \dots, a_k, a'_{k+1}, \dots, a'_L, \\ B' &= a'_1, a'_2, \dots, a'_k, a_{k+1}, \dots, a_L. \end{aligned}$$

После применения ОК имеем две старые хромосомы и всегда получаем две новые хромосомы. Схематически простой ОК показывает преобразование двух хромосом и частичный обмен информацией между ними, используя точку разрыва, выбранную случайно.

В двухточечном ОК определяются две точки ОК, и гены обмениваются между ними. Например:

p_1 :	1	1	1		0	1		0	0
p_2 :	0	0	0		1	1		1	0
p'_1 :	1	1	1		1	1		0	0
p'_2 :	0	0	0		0	1		1	0

Отметим, что точки ОК в двухточечном ОК также определяются случайно. Существует много модификаций двухточечного ОК.

Развитием двухточечного ОК является многоточечный ОК или N -точечный ОК, он выполняется аналогично двухточечному ОК, хотя большое число «разрезающих» точек может привести к потере «хороших» родительских свойств.

Порядковый оператор кроссинговера. В порядковом ОК «разрезающая» точка также выбирается случайно. Далее происходит копирование

левого сегмента p_1 в p'_1 . Остальные позиции в p'_1 берутся из p_2 в упорядоченном виде слева направо, исключая элементы, уже попавшие в p'_1 . Например:

p_1 :	A	B	C	D		E	F	G	H
p_2 :	G	A	B	E		C	D	F	H
p'_1 :	A	B	C	D		G	E	F	H

Получение p'_2 может выполняться различными способами. Наиболее распространенный метод — копирование левого сегмента из p_2 , а далее анализ p_1 методом, указанным выше. Тогда имеем p'_2 : (G A B E | C D F H).

Частично-соответствующий ОК. Здесь также случайно выбирается «разрезающая» точка или точка ОК. Далее анализируются сегменты в обоих родителях и устанавливается частичное соответствие между элементами первого и второго родителей с формированием потомков. При этом правый сегмент p_2 переносится в p'_1 , левый сегмент p_1 переносится в p'_1 с заменой повторяющихся генов на отсутствующие (гены, находящиеся в частичном соответствии). Например:

p_1 :	A	B	C	D	E	F		G	H	I	J
p_2 :	E	C	I	A	D	H		J	B	F	G
p'_1 :	A	H	C	D	E	I		J	B	F	G

Аналогично можно получить p'_2 :

p_1 :	A	B	C	D	E	F		G	H	I	J
p_2 :	E	C	I	A	D	H		J	B	F	G
p'_2 :	E	C	F	A	D	B		G	H	I	J

Циклический ОК. Циклический ОК выполняет рекомбинации согласно циклам, которые существуют при установлении соответствия между генами первого и второго родителей. Например, пусть популяция P состоит из двух хромосом $P = \{p_1, p_2\}$. Первый и второй родители и их потомок имеют вид:

p_1 :	1	2	3	4	5	6	7	8	9	10
p_2 :	5	3	9	1	4	8	10	2	6	7
p'_1 :	1	3	9	4	5	8	10	2	6	7

При выполнении циклического ОК p'_1 заполняется, начиная с первой позиции, и копирует элемент из первой позиции p_1 . Элементу 1 в p_1 соответствует элемент 5 в p_2 . Следовательно, имеем первый путь в цикле (1, 5). Элементу 5 в p_1 соответствует элемент 4 в p_2 . Имеем второй путь в первом цикле (1, 5; 5, 4). Продолжая далее, получим, что элементу 4 в p_1 соответствует элемент 1 в p_2 . Следовательно, сформирован первый цикл

(1, 5; 5, 4; 4, 1). Согласно этому циклу элемент 5 переходит в пятую позицию p'_1 , а элемент 4 — в четвертую позицию. Сформируем теперь второй цикл. Элемент 2 в p_1 соответствует элементу 3 в p_2 . Продолжая аналогично, получим второй цикл (2, 3; 3, 9; 9, 6; 6, 8; 8, 2) и третий (7, 10; 10, 7) циклы. Следовательно, в p'_1 элемент 3 расположен во втором локусе, т. е. на второй позиции, элемент 9 — в третьем, элемент 6 — в девятом, элемент 8 — в шестом, элемент 2 — в восьмом, элемент 10 — в седьмом и элемент 7 — в десятом локусах. Циклический ОК и его модификации эффективно применяются для решения комбинаторно-логических задач, задач на графах и гиперграфах и других оптимизационных задач.

Универсальный ОК. В настоящее время он популярен для решения различных задач теории расписаний. Вместо использования разрезающей точки (точек) здесь определяют двоичную маску, длина которой равна длине заданных хромосом. Получение потомков выполняется на основе правил сложения булевой алгебры соответствующих генов родителей и маски ($0 + 0 = 0$, $0 + 1 = 1$, $1 + 1 = 0$). Для каждого элемента в p_1 , p_2 гены меняются, как показано на следующем примере:

p_1 :	0	1	1	0	0	1
p_2 :	0	1	0	1	1	1
	0	1	1	0	1	0
	— маска					
p'_1 :	0	0	0	0	1	1
p'_2 :	0	0	1	1	0	1

Маска обычно выбирается случайно с заданной вероятностью или на основе генератора случайных чисел. При этом чередование 0 и 1 в маске происходит с вероятностью $\approx 0,5$ (т. е. приблизительно в 50% случаев). В некоторых случаях используются параметризованный универсальный ОК, где маска может выбираться с вероятностью для 1 или 0 выше, чем 0,5. Такой вид маски эффективен, когда хромосомы кодируются в двоичном алфавите.

В оптимизационных задачах находят применение различного типа «жадные» операторы кроссинговера. Они основаны на анализе ЦФ решений после каждого шага «жадного» ОК. Он может быть реализован на двух и более хромосомах, а в пределе — на всей популяции. Приведем типичный модифицированный алгоритм «жадного» ОК:

1. Для всех хромосом популяции вычисляется ЦФ. Выбирается заданное число родительских хромосом, и случайным образом на одной из хромосом определяется точка «жадного» ОК.
2. В выбранной хромосоме для i -го гена, расположенного слева от точки «жадного» ОК, определяется частичная ЦФ, т. е. стоимость пути от i -го гена к рядом находящемуся гену. Аналогичные действия выполняются по определению стоимости пути во всех остальных хромосомах, выбранных для «жадного» ОК.
3. В хромосому «потомок» выбирают тот ген, у которого значение ЦФ выше (ниже) при максимизации (минимизации) ЦФ.

4. Процесс продолжается, пока не будет построена хромосома «потомок». Если в процессе реализации «жадного» ОК возникает цикл или тупик, то в потомок выбираются нерассмотренные гены с лучшей ЦФ.

Например, пусть популяция P состоит из трех родительских хромосом $P = \{p_1, p_2, p_3\}$, где $p_1:(abcde)$; $p_2:(bdeca)$; $p_3:(ebadc)$. Причем стоимость (ЦФ) для каждого гена в хромосоме задана матрицей:

	a	b	c	d	e
a	—	15	6	7	8
b	15	—	4	3	2
c	6	4	—	1	10
d	7	3	1	—	9
e	8	2	10	9	—

Согласно алгоритму выберем точку «жадного» ОК между генами b и c в хромосоме p_1 . Теперь выбор (b–c) дает значение ЦФ, равное 4; выбор (b–a) определяет ЦФ со значением 15, а выбор (b–d) определяет ЦФ, равную 3. При решении задачи минимизации ЦФ выберем путь (b–d). Продолжая далее, получим путь реализации «жадного» ОК (рис. 3.1). Итак, хромосома потомка p' : (b d c a e) имеет суммарную ЦФ, равную 18 ($3 + 1 + 6 + 8 = 18$), а ЦФ родителей для p_1 равна $15 + 1 + 1 + 9 = 29$, для p_2 равна $3 + 9 + 10 + 6 = 28$ и для p_3 равна $2 + 15 + 7 + 1 = 25$. Стратегию «жадного» ОК можно выполнять различными способами.

Следует отметить, что исследователи продолжают поиск оптимального ОК.

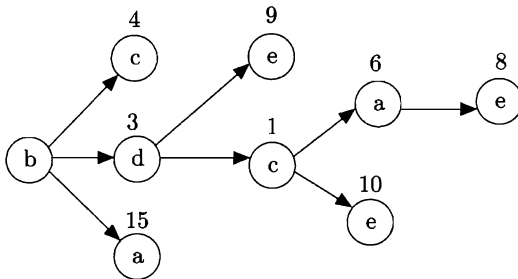


Рис. 3.1. Пример реализации «жадного» ОК

Рассмотрим кратко основные операторы мутации (ОМ). Мутация необходима потому, что предотвращает потерю важного генетического материала [88–92]. Обычно ОМ является одноточечным оператором, который случайно выбирает ген в хромосоме и обменивает его на рядом расположенный ген. Например, одноточечный ОМ имеет вид:

p_1 : 0 1 1 | 0 1 1 до ОМ,
 p'_1 : 0 1 0 | 1 1 1 после ОМ.

Двухточечный ОМ заключается в перестановке генов, расположенных справа от точек разрыва. Например:

p : A |B C D |E F до двухточечного ОМ,
 p' : A E C D B F после двухточечного ОМ.

Процесс преобразования в ЕС и ИС обычно происходит толчками. При этом важную роль играют толчковые мутации. Они не изменяют размера и строения хромосом, а изменяют взаимное расположение генов в хромосоме.

В оптимизационных задачах, с нашей точки зрения, интерес может представлять использование ОМ, основанных на знаниях о решаемой задаче. Такие ОМ называются «аргументированными знаниями». В них могут переставляться местами любые выбранные гены в хромосоме. Как правило, в таких ОМ точка или точки мутации определяются не случайно, а направленно. Например, в позиционном операторе мутации две точки мутации выбираются случайно, а затем ген, соответствующий второй точке мутации, размещается в позицию перед геном, соответствующим первой точке мутации. Например:

p : A |B C D |E F
 p' : A E B C D F.

Согласно Д. Холланду, ОМ выполняется следующим образом:

1. В хромосоме $A = (a_1, a_2, a_3, \dots, a_{L-2}, a_{L-1}, a_L)$ определяются случайным образом две позиции (например, a_2 и a_{L-1}).
2. Гены, соответствующие выбранным позициям, переставляются, и формируется новая хромосома. $OM - A' = (a_1, a_{L-1}, a_3, \dots, a_{L-2}, a_2, a_L)$.

Заметим, что в ПГА ОК — бинарная операция, а ОМ — унарная операция. Кроме того, ОК и ОМ соответствуют перестановкам элементов внутри заданного множества. Важным понятием в ОМ является шкала мутации, которая определяет, какой процент общего числа генов в популяции видоизменяется в каждой генерации. Для оптимизационных задач вероятность ОК обычно принимают равной $0,6 \div 0,99$, а вероятность ОМ от $0,6$ и меньше [16–19].

Кроме мутации, популярным и используемым в ЕС и ИС является **оператор инверсии**. В операторе инверсии случайным образом определяется одна или несколько точек инверсии, внутри которых элементы инвертируются.

Генетический оператор инверсии в ГА Д. Холланда [88] состоит из следующих шагов:

1. Хромосома $B = (b_1, b_2, \dots, b_L)$ выбирается случайным образом из текущей популяции.
2. Два числа y'_1 и y'_2 выбираются случайным образом из множества $\{0, 1, 2, \dots, L + 1\}$, далее определяем $y_1 = \min\{y'_1, y'_2\}$ и $y_2 = \max\{y'_1, y'_2\}$.
3. Новая хромосома формируется из B путем инверсии сегмента, который лежит справа от позиции y_1 и слева от позиции y_2 в хромосоме B .

Тогда, после применения оператора инверсии, получаем $B_1 : B_1 = (b_1, \dots, b_{y_1}, b_{y_2-1}, b_{y_2-2}, \dots, b_{y_1+1}, b_{y_2}, \dots, b_L)$.

Например, для двухточечного оператора инверсии получим:

p_1 : A B C | D E F | G H до оператора инверсии,
 p'_1 : A B C | F E D | G H после.

Для однотоочечного оператора инверсии запишем:

p_2 : A | B C D E F G H до оператора инверсии,
 p'_2 : A | H G F E D C B после.

Кроме простого оператора инверсии, в [89] описан специальный оператор инверсии. В нем точки инверсии определяются с заданной вероятностью для каждой новой создаваемой хромосомы в популяции. Оператор инверсии до последнего времени не получил широкого использования при решении оптимизационных задач.

Рассмотрим *оператор транслокации*. Он представляет собой комбинацию ОК и оператора инверсии. В процессе транслокации случайным образом производится один разрыв в каждой хромосоме. При формировании потомка p'_1 берется левая часть до разрыва из родителя p_1 и инверсия правой части до разрыва из p_2 . При создании p'_2 берется левая часть p_2 и инверсия правой части p_1 . Приведем пример оператора транслокации:

p_1 :	A	B	C	D	E	F
p_2 :	G	K	H	I	J	Q

p'_1 :	A	B	Q	J	I	H
p'_2 :	G	K	F	E	D	C.

Существует большое число других видов оператора транслокации. Отметим, что до последнего времени оператор транслокации не применялся в ГА, а также при разработке интеллектуальных ИС и решении оптимизационных задач.

Кроме описанных операторов, по мнению авторов, интерес может представлять *оператор сегрегации* и различные его модификации. Приведем один из примеров реализации оператора сегрегации. Отметим, что оператор сегрегации, как правило, реализуется на некотором наборе хромосом. Пусть имеется популяция P , состоящая из четырех родительских хромосом

$$P = \{p_1, p_2, p_3, p_4\} : p_1 : (1234); p_2 : (2431); p_3 : (3142); p_4 : (4321).$$

Тогда потомок p'_1 можно сформировать случайным образом, взяв первый элемент (один или несколько генов) из p_1 , второй из p_2 , третий из p_3 и четвертый из p_4 . При этом должны быть отброшены повторяющиеся решения, или решения, содержащие одинаковые элементы. Так, вариант $p'_1 : (1412)$ должен быть отброшен как содержащий две единицы, а вариант $p'_2 : (2143)$ является легальным (допустимым или реальным). Очевидно, что оператор

сегрегации можно реализовать различными способами в зависимости от способа выбора генов из хромосом.

Опишем *оператор удаления*. При реализации оператора удаления направленным или случайным образом определяется точка или точки оператора удаления. Далее производится пробное удаление генов или их ансамблей с вычислением изменения ЦФ. При достижении экстремума ЦФ оператор удаления реализуется. Элементы, расположенные справа от точки оператора удаления или между двумя точками оператора удаления, удаляются из хромосомы.

Опишем *оператор вставки*. При реализации оператора вставки направленным или случайным образом определяется точка или точки оператора вставки. Затем анализируются другие гены хромосом в популяции для определения альтернативных вставок. Далее производится пробная вставка генов или их ансамблей с вычислением изменения ЦФ. При достижении экстремума ЦФ оператор вставки реализуется. Новые гены или их ансамбли вставляются в хромосому справа от точки оператора вставки или между его двумя точками. Отметим, что оператор удаления и оператор вставки меняют размер хромосом. Для сохранения размера хромосом постоянным эти операторы необходимо применять совместно.

Рассмотрим теперь *понятие рекомбинации*. Функция рекомбинации определяет, как новая генерация хромосом будет построена из родителей и потомков. Другими словами, функция рекомбинации — это анализ и преобразование популяции при переходе из одной генерации в другую. Существует много путей выполнения рекомбинации [16–19, 89]. Один из них состоит из перемещения родителей в потомки после каждого генетического оператора (ГО). Другой путь заключается в перемещении некоторого процента популяции, используя потомков в течение каждой генерации. Обычно в ГА задается параметр $W(P)$, который управляет этим процессом. Так, $N_p(1 - W(P))$ элементов в популяции P , выбранных случайно, могут «выжить» в следующей генерации. Здесь N_p — размер популяции. Величина $W(P) = 1$ означает, что целая предыдущая популяция перемещается в новую в каждой генерации. При дальнейшей реализации алгоритма лучшие элементы из родителей и потомков будут выбираться для формирования новой популяции.

3.2. Простой генетический алгоритм

Эволюционный процесс реализуется за счет способности «лучших» хромосом оказывать большее влияние на состав новой популяции посредством длительного выживания и более многочисленного потомства. Основные этапы процесса эволюции, на основе которого создаются нужные схемы генетического поиска, согласно Д. Холланду следующие:

1. Конструирование начальной популяции. Ввести точку отчета поколений $t = 0$. Вычислить приспособленность хромосом популяции, а затем среднюю приспособленность популяции.
2. Установление $t = t + 1$. Выбрать двух родителей (хромосом) для крос-

синговера. Он выполняется случайным образом пропорционально приспособляемости родителей.

3. Формирование генотипа потомка. С заданной вероятностью произвести над генотипами выбранных хромосом кроссинговер. Выбрать с вероятностью 0,5 один из потомков $p(t)$ и сохранить его как члена новой популяции. Последовательно применить к $p(t)$ оператор инверсии, а затем мутации с заданными вероятностями. Полученный генотип потомка сохранить как $p'(t)$.
4. Отбор хромосомы случайным образом для исключения ее из популяции. Обновление текущей популяции заменой отобранной хромосомы на $p'(t)$.
5. Определение приспособленности (целевой функции) $p'(t)$ и пересчет средней приспособленности популяции.
6. Если $t = t_{\text{заданному}}$, то переход к 7, если нет, то переход к 2.
7. Конец работы.

Мы привели так называемый репродуктивный план Д. Холланда в несколько упрощенном виде. Заметим, что в технических задачах оптимизации часто вместо понятия «приспособленность» используют понятие «целевая функция». Биологи называют эту функцию — *fitness function*. Простой генетический алгоритм (ПГА) был впервые описан Д. Гольдбергом на основе работ Д. Холланда [88, 89]. Его механизм несложен. Он копирует последовательности и переставляет их части. Предварительно ПГА случайно генерирует популяцию последовательностей хромосом (альтернативных упорядоченных и неупорядоченных решений). Затем он применяет множество простых операций к начальной популяции и генерирует новые популяции.

ПГА состоит из трех операторов:

- репродукции;
- кроссинговера (ОК);
- мутации (ОМ).

Репродукция — процесс, в котором хромосомы копируются пропорционально их ЦФ. Копирование хромосом с «лучшим» значением ЦФ имеет большую вероятность для попадания в следующую генерацию. Рассматривая эволюцию Ч. Дарвина, можно отметить, что оператор репродукции, конечно, является искусственной версией натуральной селекции «выживания сильнейших». Оператор репродукции представляется в алгоритмической форме различными способами. Самый простой — создать колесо рулетки, в котором каждая хромосома имеет поле, пропорциональное его ЦФ. Рассмотрим пример Д. Гольдберга [89]: необходимо найти значение максимума функции $f(x) = x^2$ на целочисленном интервале $[0, 31]$. Традиционными методами мы будем изменять значения переменной x , пока не получим $\max f(x)$.

Для объяснения и реализации ПГА построим следующую таблицу (табл. 3.1).

Хромосомы столбца 2 в табл. 3.1 сгенерированы случайным образом. Значение ЦФ для каждой хромосомы определим как квадрат значения десятичного кода двоичного числа, которое приведено для хромосом в вто-

Таблица 3.1

Номер хромосом	Хромосома (двоичный код)	Десятичный код	Значение ЦФ	Значение ЦФ, в процентах
1	0 1 1 0 0	12	144	16,2
2	1 0 0 0 0	16	256	28,8
3	0 0 1 1 1	7	49	5,5
4	1 0 1 0 1	21	441	49,5

ром столбце таблицы. Тогда суммарное значение ЦФ = 890. Для селекции хромосом используется оператор репродукции на основе колеса рулетки. На рис. 3.2 поля колеса рулетки соответствуют значениям ЦФ в процентах. В одной генерации колесо рулетки вращается, и после останова ее указатель определяет хромосому, выбранную для следующего оператора. Очевидно, не всегда хромосома с большой ЦФ в результате оператора репродукции будет выбрана для следующего оператора. Будем считать для упрощения, что $16,2 = 16$; $49,5 = 50$; $5,5 = 5$; $28,8 = 29$.

Оператор репродукции выбирает хромосомы для применения ОК. После выполнения оператора репродукции оператор кроссинговера может выполняться в 3 шага одним из ОК, описанных выше. Точка разрыва k выбирается случайно между 1 и длиной хромосомы минус единица, т. е. в интервале $(1, L - 1)$. Длина хромосомы L — это число значащих цифр в ее коде. Длина всех хромосом в табл. 3.1 равна пяти ($L = 5$). Две новых хромосомы создаются путем обмена частей хромосом между позициями $(k + 1)$ и L соответственно. Например, рассмотрим хромосомы 1 и 2 из начальной популяции (табл. 3.1). Пусть $k = 1$. Тогда получим:

$$\begin{array}{l} p_1: \quad 0 \quad | \quad 1 \quad 1 \quad 0 \quad 0 \\ p_2: \quad 1 \quad | \quad 0 \quad 0 \quad 0 \quad 0, \end{array}$$

$$\begin{array}{l} p'_1: \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \\ p'_2: \quad 1 \quad 1 \quad 1 \quad 0 \quad 0. \end{array}$$

до применения оператора кроссинговера,

после применения оператора кроссинговера.

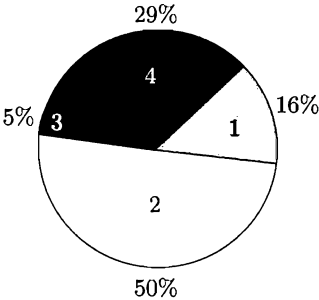


Рис. 3.2. Колесо рулетки для примера

Данные реализации ПГА для нахождения значения максимума функции $f(x) = x^2$ на целочисленном интервале $[0,31]$ приведены в табл. 3.2. Работа ГА начинается с репродукции. Мы выбираем хромосомы для следующей генерации, вращая колесо рулетки 4 раза, что соответствует мощности начальной популяции.

Величину отношения $f_i(x)/\sum f_i(x)$ называют вероятностью выбора копий (хромосом) при операторе репродукции и обозначают [88, 89]:

$$P_i(\text{OP}) = \frac{f_i(x)}{\sum f_i(x)},$$

где $f_i(x)$ — значение ЦФ i -й хромосомы в популяции, $\sum f_i(x)$ — суммарное значение ЦФ всех хромосом в популяции. Величину $P_i(\text{OP})$ также называют нормализованной вероятностью выбора. Ожидаемое число копий i -й хромосомы после оператора репродукции определяют так:

$$n_i = P_i(\text{OP}) \cdot N_p,$$

где N_p — число анализируемых хромосом.

Число копий хромосомы p_i , переходящее в следующее поколение, также иногда определяют на основе выражения (столбец 6 табл. 3.2):

$$n_i = \frac{f_i(x)}{\bar{f}(x)}.$$

где $\bar{f}(x)$ — среднее значение ЦФ по всей популяции.

Тогда для нашего примера ожидаемое число копий для первой хромосомы из табл. 3.2 равно $0,16 \cdot 4 = 0,64$ копий, для второй — $0,29 \cdot 4 = 1,16$ копий, для третьей — $0,05 \cdot 4 = 0,2$ и, наконец, для четвертой — $0,5 \cdot 4 = 2$. Смоделировать этот процесс можно, например, используя бросание монеты. В результате хромосомы p_1 и p_2 получают 1 копию, хромосома p_4 — 2 копии, и хромосома 3 не получает копий. Сравнивая этот результат с ожидаемым числом копий, получаем то, что ожидали: «лучшие» хромосомы дают большее число копий, «средние» остаются и плохие «умирают» после оператора репродукции (табл. 3.3).

Таблица 3.2

i	Начальная популяция	x	$f_i(x)$	Значение $\frac{f_i(x)}{\sum f_i(x)}$	Ожидаемое число копий	Число полученных копий
1	0 1 1 0 0	12	144	0,16	0,65	1
2	1 0 0 0 0	16	256	0,29	1,15	1
3	0 0 1 1 1	7	49	0,05	0,22	0
4	1 0 1 0 1	21	441	0,5	1,98	2
Суммарное значение			890	1,00	4,00	4
Среднее значение			222,5	0,22	0,88	1
Максимальное значение			441	0,5	2	2

Применяя к популяции после оператора репродукции (столбец 2 табл. 3.3) оператор кроссинговера, получим новую популяцию хромосом (5 столбец табл. 3.3). В принципе можно применять ОК заданное число раз.

Например, если в столбце 5 табл. 3.3 выбрать хромосомы p_2 и p_4 и между ними провести ОК (точка ОК выбрана случайно и равна 3), то можно получить:

p_2 :	1	0	1		1	1	(ЦФ = 23)
p_4 :	1	1	1		0	0	(ЦФ = 28)
p'_2 :	1	0	1	0	0		(ЦФ = 20)
p'_4 :	1	1	1	1	1		(ЦФ = 31)

Решение p'_4 случайно дало искомый наилучший результат.

После проведения одной генерации ПГА улучшились все показатели: среднее и максимальное значение ЦФ. Далее, согласно схеме классического ПГА выполняется оператор мутации.

Заметим, что ОК и ОМ соответствуют перестановкам элементов внутри заданного множества. Очевидно, что при небольшой длине хромосомы L (порядка $10 \div 20$) можно выполнить полный перебор за приемлемое время и найти наилучшие решения. При увеличении L до $50 \div 200$ и выше, полный перебор произвести затруднительно, и необходимы другие механизмы поиска. Здесь как раз и приходит на помощь направленно-случайный поиск, который можно реализовать на основе ПГА.

Таблица 3.3

i	Популяция после оператора репродукции	Случайные пары	Точка ОК	Новая популяция	x	$f_i(x)$
1	0 1 1 0 0	1–4	1	0 0 1 0 1	5	25
2	1 0 0 0 0	2–3	2	1 0 1 1 1	23	529
3	0 0 1 1 1	2–3	2	0 0 0 0 0	0	0
4	1 0 1 0 1	1–4	1	1 1 1 0 0	28	784
$\sum f_i(x)$	1338					
$\bar{f}(x)$	334,5					
$\max f_i(x)$	784					

Применим, например, ОМ к хромосоме p_2 с ЦФ = 23 (столбец 5 табл. 3.3). Выберем позиции 2 и 3 и после ОМ получим: p'_2 : (11011), выполняя далее ОМ получим p''_2 : (11101). Как видно, хромосома p''_2 имеет ЦФ = 29. Снова применим ОМ к хромосоме p''_2 . Тогда при случайном выборе позиций можно выбрать гены 4 и 5 и после ОМ получим: p''_2 : (11101) $\Rightarrow$ p'''_2 : (11110).

Хромосома p'''_2 имеет ЦФ = 30. Итак, найдено квазиоптимальное значение x , равное 30 для решения задачи нахождения максимального значения функции $f(x) = x^2$ на интервале $[0,31]$.

Как отмечалось выше, в ГА можно выделять два основных механизма воспроизводства хромосом:

- потомки являются точными копиями родителей (неполовое воспроизводство без мутации);
- потомки имеют большие отличия от родителей.

В эволюционном программировании и ГА в основном используют комбинации этих механизмов, причем согласно [91], в эволюционном программировании и ЭМ в основном используют механизмы родительского воспроизводства с мутацией, а в ГА — механизм родительского воспроизводства с рекомбинацией и мутацией.

Отметим, что в прикладных задачах начальная популяция может выбираться любым образом, например, бросанием монеты (орел = 1, другая сторона монеты = 0). Репродукция выполняется через моделирование движения колеса рулетки. ОК в задачах вычислительного характера обычно выполняется через двоичное декодирование двух положений монеты. В комбинаторно-логических задачах применяют другие типы ОК и другие технологии его выполнения. Вероятность ОК допускается равной $P(ОК) = 1,0$ и меньше, вероятность ОМ допускается равной $P(ОМ) = 0,01$ и больше. Как видно из табл. 3.2, лучшая хромосома в первой генерации (10101) получает 2 копии, потому что у нее высокая ЦФ.

Приведем вариант реализации алгоритма, описанный в [90]:

1. Инициализация популяции хромосом.
2. Оценка каждой хромосомы в популяции.
3. Создание новых хромосом посредством спаривания текущих хромосом: применение мутации и рекомбинации.
4. Устранение хромосом из популяции, чтобы освободить место для новых хромосом.
5. Оценка новых хромосом и вставка их в популяцию.
6. Если время исчерпано, то останов и возврат к наилучшей хромосоме; если нет, то переход к 3.
7. Конец работы алгоритма.

Сравнивая описание ПГА Д. Гольдберга, Д. Холланда и [90], видно, что в них реализована одна основная идея моделирования эволюции с некоторыми модификациями. Заметим, что эти изменения могут существенно влиять на окончательное качество решения.

Приведем пример одной из модифицированных базовых структур ПГА (см. [17]):

1. Создание начальной популяции решений для оптимизационной задачи.
2. Моделирование популяции (определение ЦФ для каждой хромосомы).
3. Пока оптимум заданного критерия не достигнут или в случае, когда не проведено необходимое число генераций:
 - а) выбор элементов для репродукции;
 - б) применение ОК для создания потомков (итерации);
 - в) применение ОМ (итерации);
 - г) применение оператора инверсии (итерации);
 - д) применение оператора транслокации (итерации);

- е) применение оператора сегрегации (итерации);
 - ж) применение оператора удаления вершин (итерации);
 - з) применение оператора вставки вершин (итерации);
 - и) рекомбинация родителей и потомков для создания новой генерации.
4. Реализация новой генерации.

На рис. 3.3 приведен пример модифицированной структуры ПГА. Новые модификации могут строиться путем объединения, например, пунктов а)–з) или их частичного устранения, или их перестановок, а также на основе применения адаптационных принципов управления генетическим поиском.

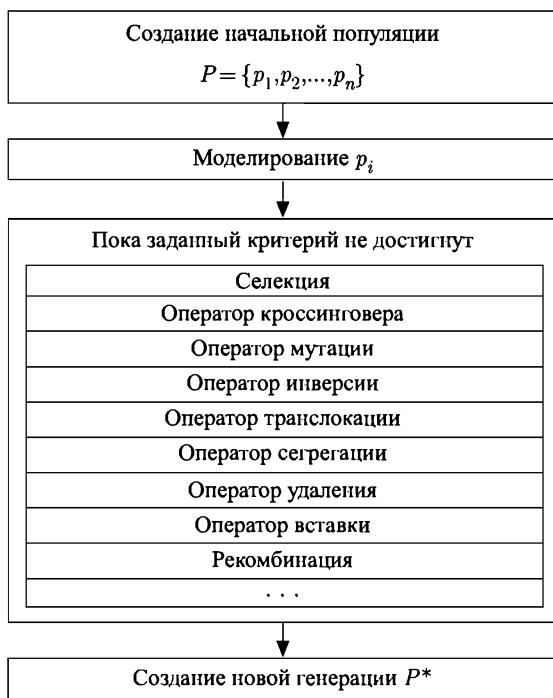


Рис. 3.3. Пример модифицированной структуры ПГА

При разработке архитектуры ГА авторы предлагают использовать некоторые положения и принципы древних учений [51–56, 80]. Основной закон существования — это наличие противоположностей. Это всеобщий принцип ИНЬ–ЯН. Согласно аналогии Гермеса: «Что наверху, то и внизу». Поэтому в генетическом поиске будем использовать основные принципы микро- макро- и метаэволюции. Различают также два вида специальной эволюции:

- эволюция напряжения (ЯН);
- эволюция расслабления (ИНЬ).

Эволюция ЯН происходит скачкообразно, аналогично эволюции де Фриза. Для эволюции ИНЬ характерно плавное развитие, аналогично эволю-

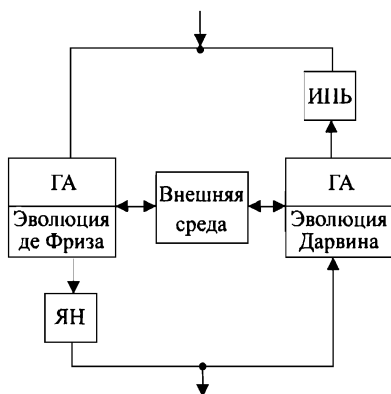


Рис. 3.4. Схема поиска

Структуры оптимизационной задачи позволяет частично избегать предварительной сходимости алгоритма.

3.3. Теоремы эволюционного моделирования

В ЭМ важным понятием является «элитизм». Он определен как операция, в которой хромосома с наилучшей ЦФ сразу копируется в новую популяцию. Это позволяет не терять лучшие решения в процессе эволюции. С точки зрения исследователей, одним из основных вопросов ГА является вопрос о подобии особей в популяции и, главным образом, о связи подобия с эффективностью.

Д. Холланд для решения этих вопросов в ГА предложил использовать алфавит, состоящий из трех символов: $\{0, 1, *\}$. Значок $*$ определяется так: «не имеет значения» и вместо него может быть использован 0 или 1. В [88] введено понятие «шаблон» (схема) в ГА, это шаблон подобия, описывающий подмножество особей (строк) в популяции с совпадением в определенных строковых позициях. Шаблон в ГА — это подмножество хромосом, которые могут быть получены из одной записи.

Например, шаблон $(*0001)$ будет состоять из двух хромосом: (00001) и (10001) . А шаблон $(*0000)$ соответствует двум хромосомам вида $\{(10000), (00000)\}$. Другой пример: шаблон $(*111*)$ описывает множество хромосом с 4 членами $\{(01110), (11110), (01111), (11111)\}$. Шаблон $(0*1**)$, будет уже иметь 8 хромосом длины 5, т.е. в общем случае в хромосоме с $L = 5$ можно иметь $3^5 = 243$ шаблона, включающих изоморфные схемы. Очевидно, что для хромосомы $L = 5$ можно иметь $2^5 = 32$ различных альтернативных шаблонов. Это следует из того, что схема $(**)$ имеет в общем $3^2 = 9$ хромосом, т.е. $\{(**), (*1), (*0), (1*), (0*), (00), (01), (10), (11)\}$. Из этих 9 хромосом выбираются 4 неизоморфные: $\{(00), (01), (10), (11)\}$ [89].

В ПГА основная идея — это объединить две хромосомы с ЦФ выше средней, а именно, найти шаблоны типа $(11***)$ и $(***111)$ при решении

циям Ч. Дарвина, Ж. Ламарка и К. Поппера. Поэтому реализация ГА в общем случае может состоять из бесконечного числа генераций. В этой связи предлагается новая схема параллельного поиска из двух ГА (рис 3.4). В первом реализуется эволюция ЯН, а во втором ИНБ. Лучшие или заданные хромосомы (решения) мигрируют из ГА ЯН в ИНБ и наоборот. Кроме этого, после каждой генерации лучшее решение идет в третий ГА, где собирают все наилучшие хромосомы. Над ними также происходит процесс эволюции. Применение такой струк-

задач максимизации ЦФ. Тогда, применяя ОК к данным шаблонам, можно получить хромосому (11111) с наилучшим значением ЦФ при поиске максимума.

Для вычисления числа шаблонов или их границы в популяции требуются точные данные о каждой хромосоме в популяции. Поэтому сначала вычисляют число шаблонов, содержащихся в индивидуальной хромосоме, а затем находят верхнюю границу общего их числа в популяции.

Рассмотрим, например, хромосому длины $L = 6$, (111111). Она имеет 2^6 шаблонов, потому что каждая позиция может быть 1 или *. В общем случае считают, что хромосома содержит 2^L шаблонов. Тогда популяция размера N_p будет содержать от 2^L до $N_p \cdot 2^L$ шаблонов. Шаблоны определенной короткой длины называют «строительными блоками» [89]. Размер строительных блоков очень важен для скорости и качества нахождения результата. Рассмотрим примеры строительных блоков для разных шаблонов. Например, в шаблоне (***)1 строительным блоком будет элемент 1. В шаблоне (10***) строительным блоком будет составной элемент 10. Заметим, что вид строительного блока должен выбираться, исходя из знаний о решаемой задаче, чтобы далее из них как из «кирпичиков» собирать «здание», т. е. решение с лучшей целевой функцией. Причем при реализации ГА должно выполняться условие, чтобы строительные блоки разрывались только в крайних случаях, указанных ЛПР.

ГА в зависимости от размера популяции разделяют на:

- стационарного состояния;
- поколенческие.

В стационарных ГА размер популяции является входным постоянным параметром, задаваемым ЛПР. В поколенческих ГА разрешается увеличивать или уменьшать размер популяции в каждой последующей генерации. Следует отметить, что речь в основном идет о появлении $N_p + N_1$ потомков ($N_1 \gg 1$), прежде чем начинает реализовываться оператор отбора, устраняющий N_1 хромосом с меньшей ЦФ. Вопросы удаления «лишних» хромосом как в стационарных, так и в поколенческих ГА, основаны на нескольких эвристиках:

- случайное равновероятное удаление хромосом;
- удаление N_1 хромосом, имеющих худшее значение ЦФ;
- удаление хромосом на основе обратно пропорционального значения ЦФ;
- удаление хромосом на основе заданной турнирной стратегии.

Можно предложить еще ряд эвристик удаления, но в [91] отмечено, что удаление худших хромосом может привести к преждевременной утрате разнообразия и, следовательно, к попаданию ЦФ в локальный оптимум. Оставление в популяции большого числа хромосом с «плохой» ЦФ приведет к слепому поиску.

Выделяют три особенности алгоритма эволюции:

- каждая новая популяция состоит только из «жизнеспособных» хромосом;

- каждая новая популяция «лучше» (в смысле ЦФ) предыдущей;
- в процессе эволюции последующая популяция зависит только от предыдущей.

Для количественной оценки шаблонов в ГА введены две характеристики [88, 89]: порядок шаблона — $o(H)$; определенная длина шаблона — $\delta(H)$.

Порядок шаблона — число закрепленных позиций (в двоичном алфавите это число единиц и нулей, представленных в шаблоне). Например, для шаблона $H = (0^{*****})$ порядок шаблона $o(H) = 1$.

Определенная длина шаблона — это расстояние между первой и последней позициями нулей и единиц. Например, для схемы $H = (011^*1^{**})$, получаем $\delta(H) = 5 - 1 = 4$, а для схемы $H = (0^{*****})$ получаем $\delta(H) = 1 - 1 = 0$.

При выполнении оператора рекомбинации хромосомы копируются пропорционально значению их ЦФ или, более точно, хромосома i получает выбор с вероятностью $P_i(\text{OP})$, определяемой выражением:

$$P_i(\text{OP}) = f_i(x) / \sum_{i=1}^{N_p} f_i(x).$$

Пусть имеется некоторый шаблон H , который присутствует в популяции P^t (это популяция P после реализации t генераций алгоритма), тогда обозначим $m(H, t)$ число хромосом популяции P^t , которые являются элементами шаблона H . После получения набора непересекающихся популяций размера N_p с перемещением части хромосом из популяции P^t (здесь t означает номер поколения или условный параметр времени) предполагается получить $m(H, t+1)$ представителей схемы H в генерации популяции P^{t+1} . Тогда выражение $m(H, t+1)$ определится по формуле:

$$m(H, t+1) = m(H, t) \times N_p \times f(H) / \sum_{i=1}^{N_p} f_i(x),$$

где $f(H)$ — средняя ЦФ хромосом, представленных схемой H в генерации t .

Если обозначить среднюю ЦФ всей популяции как

$$\bar{f}(x) = \sum_{i=1}^{N_p} f_i(x) / N_p,$$

то

$$m(H, t+1) = m(H, t) \times f(H) / \bar{f}(x).$$

Из полученного выражения видно, что ЦФ определенного шаблона изменяется как отношение средней ЦФ шаблона к средней ЦФ популяции. Пусть ЦФ шаблона остается выше среднего значения ЦФ шаблона на величину $c \times \bar{f}(x)$, где $c > 0$ — константа. Тогда имеем:

$$m(H, t+1) = m(H, t) \times (\bar{f}(x) + c \times \bar{f}(x)) / \bar{f}(x) = (1 + c) \times m(H, t).$$

Начиная с $t = 0$ и считая $c > 0$ константой получаем уравнение:

$$m(H, t) = m(H, 0) \times (1 + c)^t.$$

Данное выражение показывает, что с течением времени число хороших хромосом популяции экспоненциально растет, а число хромосом, имеющих ЦФ ниже средней, в популяции экспоненциально уменьшается.

Правило репродукции Д. Холланда запишется так: шаблон с ЦФ выше среднего значения существует и копируется в следующую генерацию, а шаблон с ЦФ ниже среднего — устраняется [88]. Если хромосомы из предыдущей популяции копируются в новую популяцию без обмена генами, то поисковое пространство не увеличивается и процесс затухает. Поэтому во всех ГА кроме оператора рекомбинации используются различные генетические операторы, такие как кроссинговер, мутация и другие. Они создают новые хромосомы и увеличивают или уменьшают количество шаблонов в популяции.

Нижняя граница вероятности выживания шаблона после применения ОК может быть приближенно определена для любой популяции. Для кроссинговера внутри строки (хромосомы) доступно $(L - 1)$ точек. Так как шаблон в ГА выживает, когда точка ОК попадает вне «определенной длины», то вероятность выживания для простого ОК записывается так:

$$P_1(s) = 1 - \delta(H) / (L - 1), \quad (3.1)$$

где L — длина хромосомы, т. е. суммарная длина, входящих в нее генов.

Например, рассмотрим хромосому $P = (01111000)$ и два шаблона $H_1 = (*1***0*)$ и $H_2 = (**11***)$. Шаблон H_1 имеет определенную длину $\delta(H_1) = 7 - 3 = 4$, и если точка ОК выбирается равновероятно среди $8 - 1 = 7$ мест, то вероятность разрушения шаблона равна $4/7$ или он выживает с вероятностью $3/7$. Аналогично, шаблон H_2 имеет определенную длину $\delta(H_2) = 5 - 4 = 1$ и разрушается с вероятностью $1/7$, а выживает с вероятностью $6/7$.

Если ОК выполняется посредством случайного выбора, например с вероятностью $P(\text{ОК})$, то вероятность выживания схемы определится таким образом:

$$P_2(s) \geq 1 - \alpha P(\text{ОК}) \delta(H) / (L - 1), \quad (3.2)$$

где α — коэффициент, определяющий применение модифицированных ОК, $\alpha \approx 1-4$. Допуская независимость оператора рекомбинации и ОК, получим объединенную оценку их действия:

$$m(H, t + 1) \geq [m(H, t) f(H) / \bar{f}(x)] \times [1 - \alpha P(\text{ОК}) \delta(H) / (L - 1)].$$

Из данного выражения следует, что шаблоны хромосом с ЦФ выше средней и короткой определенной длиной $\delta(H)$ имеют возможность экспоненциального роста в новой популяции [82, 83].

Рассмотрим влияние модифицированных ОМ на возможность выживания хромосом с лучшей ЦФ на следующих генерациях. Для того, чтобы после ОМ шаблон H хромосомы выжил в следующей генерации, все специфические позиции должны выжить. Выживание одиночной позиции имеет вероятность $(1 - P(\text{ОМ}))$. Шаблон H выживает, когда каждая из $o(H)$ закреплённых его позиций выживает. При умножении вероятности $(1 - P(\text{ОМ}))$ самой на себя $o(H)$ раз мы получаем вероятность выживания шабло-

на $(1 - P(OM))^{o(H)}$. При малых значениях $P(OM) \ll 1$ вероятность выживания шаблона может быть аппроксимирована выражением:

$$P_3(s) = 1 - o(H) \beta P(OM), \quad (3.3)$$

где β — коэффициент, определяющий применение модифицированных ОМ, $\beta \approx 1 \div 3$. Тогда предполагается, что произвольный шаблон H получает ожидаемое число копий в следующей генерации после модифицированных оператора рекомбинации, ОК и ОМ:

$$m(H, t+1) > [m(H, t) f(H) / \bar{f}(x)] \times \\ \times [1 - \alpha P(OK) \delta(H) / (L-1) - o(H) \beta P(OM)]. \quad (3.4)$$

Это выражение по аналогии с [82, 83] назовем модифицированной фундаментальной теоремой ГА.

Вероятность выживания шаблона H в следующих генерациях после применения оператора инверсии запишется в виде [88, 89]:

$$P_4(s) \geq 1 - 2\varphi P(OИ) \cdot \left[\frac{\delta(H)}{L-1} - \frac{\delta(H)^2}{(L-1)^2} \right], \quad (3.5)$$

где $P(OИ)$ — вероятность выбора хромосомы, соответствующей шаблону H из популяции на заданном шаге генерации (вероятность оператора инверсии); φ — коэффициент, определяющий применение модифицированных операторов инверсии, $\varphi \approx 1 \div 3$. Тогда фундаментальная теорема ГА (3.4) с учетом оператора инверсии запишется:

$$m(H, t+1) > \left[m(H, t) \frac{f(H)}{\bar{f}(x)} \right] \times \\ \times \left[1 - \alpha P(OK) \frac{\delta(H)}{(L-1)} - o(H) \beta P(OM) - P_1(s) \right].$$

Данное выражение перепишем в упрощенном виде:

$$m(H, t+1) > \left[m(H, t) \frac{f(H)}{\bar{f}(x)} \right] \times [P_2(s) - P_3(s) - P_4(s)].$$

При использовании оператора сегрегации вероятность выживания шаблона на следующей генерации определяется по формуле

$$P_5(s) \geq [1 - \gamma P(OC) \delta(H) / (L-1)] / N_p, \quad (3.6)$$

где γ — коэффициент, определяющий применение модифицированных операторов сегрегации, $\gamma \approx 1 \div 3$.

Для оператора транслокации вероятность выживания запишется

$$P_6(s) \geq [1 - \alpha P(OK) \delta(H) / (L-1)] \varphi P(OИ).$$

При использовании оператора удаления вероятность выживания шаблона на следующей генерации определяется по формуле:

$$P_7(s) \geq [1 - \delta P(OY) \delta(H) / (L - 1)] / N_p,$$

где δ — коэффициент, определяющий применение оператора удаления, $\delta \approx 1 \div 2$.

При использовании оператора вставки вероятность выживания шаблона на следующей генерации определяется так:

$$P_8(s) \geq [1 - \lambda P(OB) \delta(H) / (L - 1)] / N_p,$$

где λ — коэффициент, определяющий применение оператора вставки, $\lambda \approx 1 \div 2$.

Тогда модифицированная фундаментальная теорема ГА с учетом всех рассмотренных генетических операторов для решения инженерных оптимизационных задач примет вид

$$m(H, t + 1) > [m(H, t) f(H) / \bar{f}(x)] \times \\ \times [|P_2(s) - P_3(s) - P_4(s) - P_5(s) - P_6(s) - P_7(s) - P_8(s)|]. \quad (3.7)$$

Основная теорема ГА, приведенная Д. Холландом, показывает асимптотическое число схем, «выживающих» при реализации ПГА на каждой итерации. Выражение (3.7), в отличие от теоремы Холланда, показывает асимптотическое число хромосом, «выживающих» при реализации модифицированных операторов кроссинговера, мутации, инверсии, сегрегации, транслокации, удаления и вставки на каждой генерации. Очевидно, что это число приблизительное. Оно меняется в зависимости от вероятности применения того или иного ГА. Особенное влияние на число «выживающих» и «умирающих» хромосом при реализации ГА оказывает значение ЦФ отдельной хромосомы и всей популяции и знания о конкретной решаемой задаче. Во многих инженерных задачах имеются специальные знания, позволяющие построить аппроксимационную модель.

Лемма 3.1. Если на некотором шаге генерации ГА $P_1(\text{ГА})$ есть вероятность того, что хромосома p_i определяет потомка на этом шаге и $P_2(\text{ГА})$ есть вероятность, что p_i уничтожается в течение этого шага, то ожидаемое число потомков хромосомы p_i есть $P_1(\text{ГА})/P_2(\text{ГА})$.

Рассмотрим, например, хромосому длины $L = 7$ и две схемы, представляющие ее:

$$\begin{array}{l} p_i = \quad 0 \quad 1 \quad 1 \quad | \quad 1 \quad 0 \quad 0 \quad 0 \\ H_1 = \quad * \quad 1 \quad * \quad | \quad * \quad * \quad * \quad 0 \\ H_2 = \quad * \quad * \quad * \quad | \quad 1 \quad 0 \quad * \quad *, \end{array}$$

где $|$ — символ, обозначающий точку кроссинговера. Схема H_1 после ОК, скорей всего, будет уничтожена потому, что 1 в позиции 2 и 0 в позиции 7 расположатся в различных новых хромосомах после ОК. Ясно, что схема

H_2 будет выживать, так как после ОК число 1 в позиции 4 и 0 в позиции 5 будут в той же самой новой хромосоме. Хотя точка ОК выбрана случайно, ясно, что схема H_1 менее приспособлена к выживанию, чем схема H_2 . Если точка ОК выбирается случайно среди $L-1 = 7-1 = 6$ возможных позиций, то схема H_1 уничтожается с вероятностью $P(d) = \delta(H_1)/(L-1) = 5/6$, она же (схема H_1) выживает с вероятностью $P_1(s) = 1 - P(d) = 1/6$. Аналогичным образом, схема H_2 имеет $\delta(H_2) = 1$ и вероятность ее уничтожения $P(d) = 1/6$, а вероятность выживания $P_1(s) = 5/6$.

Вернемся к примеру Д. Гольдберга вычисления максимума функции $f(x) = x^2$. В добавление к имеющейся информации пусть имеются три частные схемы $H_1 = (1****)$, $H_2 = (*10**)$ и $H_3 = (1***0)$, описанные в табл. 3.4.

Таблица 3.4

Схемы	Перед репродукцией	Номера хромосом	Значение средней ЦФ схемы $f(H)$
H_1	1****	2, 4	348,5
H_2	*01**	3, 4	245
H_3	1***0	2	256

В табл. 3.5 во втором столбце приведем ожидаемое число копий хромосом, которые должны перейти из генерации t в генерацию $t+1$ после оператора рекомбинации, а в третьем столбце реальное число копий хромосом, которые должны перейти из генерации t в генерацию $t+1$ после оператора рекомбинации. В четвертом и пятом столбце приведены аналогичные результаты после выполнения всех генетических операторов. В четвертом и шестом столбцах таблицы 3.5 приведены номера хромосом.

Таблица 3.5

Схемы	После репродукции			После всех операторов		
	Ожидаемое число	Реальное число	Номера хромосом	Ожидаемое число	Реальное число	Номера хромосом
H_1	3,20	2	2, 4	3,13	2	2, 4
H_2	2,18	2	3, 4	2,0	2	1, 2
H_3	1,97	2	2	1,15	1	4

Рассмотрим сначала схему H_1 . В течение репродукции хромосомы копируются с вероятностью, определенной согласно величине их ЦФ. Из первой строки табл. 3.4 следует, что хромосомы номеров 2, 4 являются представителями схемы H_1 (см. табл. 3.3). После оператора рекомбинации получены две копии схемы и хромосомы с номерами 2, 4 вошли в популяцию. Проверим, соответствует ли это число фундаментальной теореме. Из (3.4) при $\alpha, \beta = 1$ ожидается получить $n(H, t)f(H)/\bar{f}(x)$ копий. Вычисляя значение средней ЦФ $f(H_1)$, получим $(256 + 441)/2 = 348,5$. Разделив это число на значение средней ЦФ популяции ($\bar{f}(x) = 222,5$) и умножая на число H_1

схем в момент времени t ($m(H, t) = 2$), получим ожидаемое число H_1 схем в момент времени $(t + 1)$, т. е. $m(H, t + 1) = 2 \cdot 348,5/222,5 = 3,13$. Сравнивая это число с реальным числом копий (схем) — 2, видим, что получили меньшее число копий.

Продолжая определим, что ОК не может дать дальнейший эффект, так как определенная длина $\delta(H_1) = 1$ предотвращает разрыв единственного бита. Далее, если для ОМ взять $P(ОМ) = 0,01$, то ожидается $m(H, t + 1) \times P(ОМ) = 3 \cdot 0,01 = 0,03$, но не существует битового обмена с тремя копиями в трех хромосомах. Как результат, для схемы H_1 получаем ожидаемое экспоненциальное увеличение числа схем, что соответствует (3.4). Рассмотрим теперь схемы H_2 и H_3 с двумя закрепленными позициями. Схема H_2 имеет также две хромосомы в начальной популяции (номера 3, 4) и имеет две копии в следующей репродукции. Вычисляем $m(H_2) = 2 \cdot 245/222,5 = 2$. Здесь 245 — средняя ЦФ схемы, а 222,5 — средняя ЦФ популяции. Для H_3 получим только одну хромосому и $m(H_3) = 1 \cdot 256/222,5 = 1,15$. Здесь 256 — значение средней ЦФ схемы.

Заметим, что для короткой схемы H_2 две копии — это хороший результат и ОК может случиться только один раз за четыре возможности ($L - 1 = 5 - 1 = 4$). В схеме $H_2 = (*|0|1|*|*)$ — это единственная реальная возможность, которая может дать новую копию. Как результат, схема H_2 выживает с высокой вероятностью. Действительное число H_2 схем есть $m(H_2, t + 1) \approx 2$. Для схемы H_3 , так как $\delta(H_3) = 4$, ОК обычно разрушает ее.

Ответа на вопрос, почему необходимо давать выживание схемам с лучшей ЦФ, не существует или он нечеткий, или каждый раз зависит от конкретной задачи. Часто возникает вопрос, а сколько новых схем в новой генерации необходимо, сколько реально, сколько полезно [89]?

Д. Холланд считает, что количество шаблонов, которое может быть получено из популяции размера N_p с длиной хромосом, равной L , лежит между 2^L и $N_p \cdot 2^L$. Д. Гольдберг и Д. Холланд отмечают, что число «эффективных» шаблонов составляет ориентировочно величину N_p^3 [88, 89].

Рассмотрим популяцию N_p двоичных хромосом длины L . Будем считать только шаблоны, которые выживают с вероятностью большей, чем константа $P(s)$. Как результат упрощения, рассмотрим простой ОК и ОМ с малой вероятностью выполнения. Это приведет к рассмотрению только тех шаблонов, где $\varepsilon < 1 - P(s)$. Будем считать только те шаблоны, где длина $L(s) < \varepsilon(L - 1) + 1$. Ясно, что существует $2^{L(s)-1}$ таких шаблонов. Для вычисления общего числа шаблонов возьмем заданный шаблон и будем перемещать его вдоль хромосомы на каждом шаге. Выполним этот шаг общее число $L - L(s) + 1$ раз и тогда можно вычислить общее число шаблонов длины $L(s)$ или меньше следующим образом:

$$2^{L(s)-1} \times (L - L(s) + 1).$$

Данная формула показывает число шаблонов для одной хромосомы. Очевидно, для целой популяции число шаблонов ориентировочно определяется как

$$N_p \times 2^{L(s)-1} \times (L - L(s) + 1).$$

Размер популяции обычно берут равным $N_p = 2^{L(s)/2}$. Считают, что нижняя граница числа шаблонов, «выживающих» после ГА, ориентировочно равна [88, 89]:

$$N(s) \geq N_p \cdot (L - L(s) + 1) \cdot 2^{L(s)-2}.$$

Если учесть, что $N_p = 2^{L(s)/2}$, то получим

$$N(s) = \frac{[(L - L(s) + 1) \cdot N_p^3]}{4}.$$

Следовательно, число схем «выживания» пропорционально кубу размера популяции и имеет порядок $N(s) = \omega N_p^3$, где ω — коэффициент ($\omega = 1 \div 4$).

В большинстве ГА и ЭМ используется механизм или процедура случайного скрещивания. В биологии это имеет место, когда пол индивида не определен. В инженерных задачах обсуждаются различные механизмы и модели этого процесса. Основные из них описаны в [91]:

- М1 — вытеснение (crowding factor). Этот механизм определяет способ и порядок замены старых хромосом из генерации t новыми после генерации $t + 1$. При этом из популяции удаляются «похожие на себя» хромосомы и остаются отличающиеся.
- М2 — разделение (sharing). Этот механизм вводит зависимость ЦФ хромосомы от их распределения в популяции и поисковом пространстве. Это позволяет копиям старых хромосом или близких к ним не появляться в популяциях.
- М3 — введение идентификаторов (tagging). Специальным хромосомам присваиваются метки. Операторы ГА применяются только к помеченным хромосомам.

Сформулируем описание ГА в виде научной теории. Отметим, что способ построения научной теории, в основе которой используются исходные положения, называемые аксиомами, а все остальные предложения теории получаются как логические следствия аксиом, называется аксиоматический метод. Основным в нем является метод интерпретаций.

Пусть каждому исходному понятию и отношению аксиоматической теории ЭМ поставлен в соответствие некоторый конкретный математический объект. Совокупность таких объектов называется полем интерпретации. Всякому утверждению U теории ЭМ ставится в соответствие некоторое высказывание U^* об элементах поля интерпретации, которое может быть истинным или ложным. Тогда можно сказать, что утверждение U теории ЭМ соответственно истинно или ложно в данной интерпретации. Поле интерпретации и его свойства сами обычно являются объектом рассмотрения другой теории ГА, которая, в частности, может быть аксиоматической. Этот метод позволяет доказывать суждения типа: если теория ГА непротиворечива, то непротиворечива и теория ЭМ.

Пусть теория ЭМ проинтерпретирована в теории ГА таким образом, что все аксиомы A_i теории ЭМ интерпретируются истинными суждениями A_i^* теории ГА. Тогда всякая теорема теории ЭМ, т.е. всякое утверждение A ,

логически выведенное из аксиом A_i в ЭМ, интерпретируется в ГА некоторым утверждением A^* , выводимым в ГА из интерпретаций A_i^* аксиом A_i и, следовательно, истинным [109].

Метод интерпретаций позволяет также решать вопрос о независимости систем аксиом: для доказательства того, что аксиома A теории ЭМ не зависит от остальных аксиом этой теории, т.е. не выводима из них, и, следовательно, необходима для получения всего объема данной теории, достаточно построить такую интерпретацию ЭМ, в которой аксиома A была бы ложна, а все остальные аксиомы этой теории истинны. Уточнением понятия аксиоматической теории является понятие формальной системы. Это позволяет представлять математические теории как точные математические объекты и строить общую теорию или метатеорию таких теорий. Всякая формальная система строится как точно очерченный класс выражений—формул, в котором некоторым точным образом выделяется подкласс формул, называемых теоремами данной формальной системы. При этом формулы формальной системы непосредственно не несут в себе содержательного смысла. Их можно строить из произвольных знаков и символов. Общая схема построения произвольной формальной системы ЭМ такова:

1. Язык системы ЭМ: аппарат алгебры логики; теория множеств; теория графов.

1.1. Алфавит — перечень элементарных символов системы: двоичный, десятичный, буквенный, Фибоначчи и др.

1.2. Правила образования (синтаксис): правила, по которым из элементарных символов строятся формулы системы ЭМ. Правила:

- построения модели эволюции;
- конструирования популяций;
- построения ЦФ;
- разработки генетических операторов;
- репродукции популяций;
- рекомбинации популяций;
- редукции.

Последовательность элементарных символов считается формулой тогда и только тогда, когда она может быть построена с помощью правил образования.

2. Аксиомы системы ЭМ. Выделяется некоторое множество конечных формул, которые называются аксиомами системы. В ЭМ существует большое число аксиом. Основные из них следующие:

- «Выживание сильнейших», т.е. переход решений с лучшей целевой функцией в следующую генерацию.
- Размер популяции после каждой генерации остается постоянным.
- Обязательное применение во всех генетических алгоритмах операторов кроссинговера и мутации.
- Число копий (решений), переходящих в следующую генерацию, определяется по формуле (3.7).

3. Правила вывода ЭМ. Фиксируется конечная совокупность предикатов $\Pi_1, \Pi_2, \dots, \Pi_k$ на множестве всех формул системы. Пусть $\Pi(x_1, \dots, \dots, x_{n_i+1})$ — какой-либо из этих предикатов ($n_i > 0$). Если для данных формул $F_1, \dots, F_{n_i+1}$ утверждение $\Pi(F_1, \dots, F_{n_i+1})$ истинно, то говорят, что формула F_{n_i+1} непосредственно следует из формул $F_1, \dots, F_{n_i+1}$ по правилу Π_i .

Заданием 1, 2, 3 исчерпывается задание формальной системы ЭМ как точного математического объекта. При этом степень точности определяется уровнем точности задания алфавита, правил образования и правил вывода. Выводом системы ЭМ называется всякая конечная последовательность формул, в которой каждая формула либо является аксиомой системы ЭМ, либо непосредственно следует из каких-либо предшествующих ей этой последовательности формул по одному из правил вывода Π_i системы.

Всякую конкретную математическую теорию ЭМ можно перевести на язык подходящей формальной системы таким образом, что каждое ложное или истинное предложение теории ЭМ выражается некоторой формулой системы. Метод интерпретаций позволяет устанавливать факт относительной непротиворечивости, т. е. доказывать суждения типа: «если теория ГА непротиворечива, то непротиворечива и теория ЭМ». В общем проблема непротиворечивости не решена и является одной из основных в математике [109].

В заключении отметим, что существуют четыре основных отличия ГА от оптимизационных методов:

- прямое преобразование кодов;
- поиск из популяции, а не из единственной точки;
- поиск через элементы (слепой поиск);
- поиск, использующий стохастические и модифицированные операторы, а не детерминированные правила.

Использование ЭМ и ГА при решении инженерных задач позволяет уменьшить объем и время вычислений и упростить моделирование функций, сократить число ошибок моделирования.

ИНСТРУМЕНТАЛЬНЫЕ СРЕДСТВА ЭВОЛЮЦИОННОГО МОДЕЛИРОВАНИЯ

Каждая отрасль наших знаний последовательно
проходит через три различные теоретические
стадии: теологическую или фиктивную,
метафизическую или абстрактную, научную или
положительную.
О. Конт

4.1. Технологии локального и генетического поиска

Рассмотрим инструментальные средства ЭМ, в которых используется ряд простейших методов одномерного поиска, градиентного поиска, а также статистических методов оптимизации. На рис. 4.1 показана одна из возможных классификаций таких стратегий поиска.

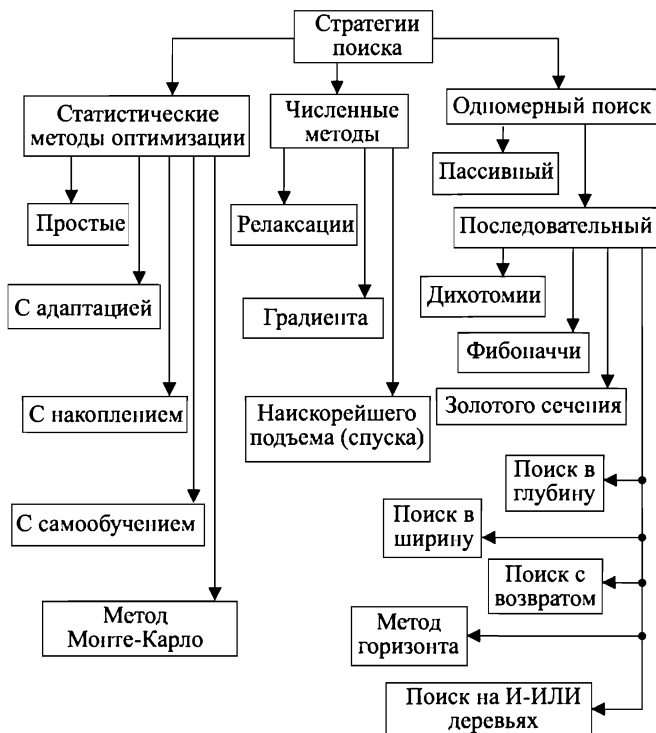


Рис. 4.1. Классификация стратегий поиска

К **одномерному поиску** относят методы: пассивный, последовательный. **Пассивный поиск** заключается в случайном выборе пар на заданном отрезке для нахождения оптимума ЦФ. **Последовательный поиск** выполняется путем перебора значений ЦФ для нахождения оптимального значения. Опишем один из возможных методов одномерного поиска — алгоритм поиска из начальной точки [112].

1. Получить набор или одно альтернативное решение.
2. Построить график зависимости ЦФ от времени (или числа элементов модели оптимизационной задачи).
3. Ввести начальную точку V_0 , начальный шаг ΔV , точность ε , вычислить ЦФ $F(V_0)$.
4. Положить $V_1 = V_0 + \Delta V$, вычислить ЦФ $F(V_1)$.
5. При минимизации ЦФ: если $F(V_1) > F(V_0)$, то $\Delta V := -\Delta V$ и перейти к 4.
6. Присвоить $\Delta V := 2\Delta V$.
7. Считать $V_2 := V_1 + \Delta V$, вычислить $F(V_2)$.
8. Если $F(V_2) < F(V_1)$, то $V_0 = V_1$, $V_1 = V_2$, $F(V_0) = F(V_1)$ и переход к 6, иначе к 9.
9. Положить $a = V_0$, $b = V_2$, вывести $[a, b]$ — интервал неопределенности.
10. Конец работы алгоритма.

К последовательному поиску относятся методы: дихотомии, Фибоначчи, золотого сечения, поиска в глубину, в ширину, с возвратом, горизонта и поиск на И-ИЛИ деревьях.

Метод дихотомии реализуется за счет механизма обычного перебора возможных точек разрыва. Он аналогичен методу деления отрезка пополам для нахождения точки, в которой ЦФ имеет локальный оптимум [110–113].

Например, на отрезке $A : a \overset{1}{|} b \overset{2}{|} c \overset{3}{|} d \overset{4}{|} e \overset{5}{|} f \overset{6}{|} g \overset{7}{|} i \overset{8}{|} h$ возможны максимально 8 точек разрыва. Очевидно, точность (живучесть, эффективность) такого процесса определяется количеством выбранных точек n , причем $n = L - 1$, где L — длина отрезка, т. е. количество выделенных дискретных точек. Дискретный интервал $[1-8]$ обозначим L_0 . Реализованный интервал назовем интервалом неопределенности L_{II} .

Эффективность поиска будет $\frac{2L_0}{n+1}$, что приблизительно равно двум. Такой метод деления отрезка пополам для нахождения точки, в которой ЦФ имеет локальный оптимум, называется дихотомическим методом. Равномерное распределение всех разрывов на интервале $[a, b]$ не является наилучшим. Эффективность поиска можно улучшить, если все разрывы проводить последовательно и попарно, анализируя результаты после каждой пары экспериментов. Наиболее эффективные результаты — такое расположение пары разрывов, при котором текущий интервал неопределенности сокращается практически вдвое. Итак, после первого разбиения интервал

$$L_0 = L_1 + L_2, \quad L_1 = \frac{1}{2}L_0 + \varepsilon, \quad L_2 = \frac{1}{2}L_0 - \varepsilon, \quad L_3 = \frac{1}{4}L_0 + \varepsilon, \\ L_4 = \frac{1}{4}L_0 - \varepsilon, \quad L_5 = \frac{1}{8}L_0 + \varepsilon, \quad L_6 = \frac{1}{8}L_0 - \varepsilon,$$

где $\varepsilon = 0, 1, 2, \dots$ в зависимости от решения ЛПР.

Приведем укрупненный алгоритм метода дихотомии.

1. Получить набор или одно альтернативное решение.
2. Ввести границы интервала, параметр ε .
3. Делить исходный отрезок пополам (при нечетном размере в любую часть берется ближайшее большее число).
4. Вблизи точки деления (по разные стороны с наименьшим интервалом) $\varepsilon = 0, 1, 2, \dots$ находим точки с экстремальным значением ЦФ.
5. Каждую половину отрезка снова делим пополам и процесс расчета продолжаем по исходной схеме до тех пор, пока не будет получена ЦФ с наилучшим значением или ранее не будет закончено деление на основе заданной метки.
6. Конец работы алгоритма.

Эффективность метода дихотомии экспоненциально растет с ростом n .

Оптимальную стратегию последовательного поиска может дать **метод Фибоначчи** [109–113]. Сущность алгоритма состоит в том, что выборки, проводимые последовательно, располагаются так, чтобы для соседних выборок выполнялось условие

$$d_j = d_{j+1} + d_{j+2}, \quad (4.1)$$

здесь d_j — номер дискретной точки на интервале $[a, b]$. Для любой $(n - k)$ -выборки справедливо равенство

$$d_{n-k} = F_{k+1}d_n,$$

где F_k — числа Фибоначчи.

В алгоритме используется то свойство чисел ряда Фибоначчи, что очередной член ряда равен сумме двух предыдущих, кроме первого и второго: $F_k = F_{k-1} + F_{k-2}$, где $F_0 = 0$, $F_1 = 1$, $k = 2, 3, \dots$ (например, числа Фибоначчи 0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, ...).

Вначале, исходя из заданной требуемой точности ε , определяется F_n — наименьшее из чисел Фибоначчи, удовлетворяющее неравенству

$$\Delta = (b - a) / F_n \leq \frac{\varepsilon}{2}.$$

Тогда $(b - a) = F_n \Delta$, т. е. весь интервал можно разделить на количество частей F_n , причем каждая часть будет иметь длину, меньшую ε ($\Delta < \varepsilon$). При этом $F_n = F_{n-1} + F_{n-2} = F_{n-2} + F_{n-3} + F_{n-2} = 2F_{n-2} + F_{n-3}$. Точки деления x_1 и x_2 определяются по формулам:

$$x_1 = a + F_{n-2}\Delta; \quad x_2 = x_1 + F_{n-3}\Delta = a + F_{n-1}\Delta.$$

В найденных точках проверяются значения ЦФ. Если лучшие значения ЦФ достигнуты в точке x_i , то в качестве интервала выбирается $[a, x_i]$, в противном случае $[x_i, b]$. Для продолжения поиска точки следующего замера следует расположить симметрично имеющемуся разбиению. Далее поиск осуществляется аналогично. Согласно [110] точка искомого экстремума определяется с точностью $2\Delta \leq \varepsilon$, и для достижения этой точности требуется $N - 2$ вычислений.

Приведем модифицированный алгоритм поиска на основе чисел Фибоначчи для нахождения экстремума ЦФ:

1. Получить набор или одно альтернативное решение.
2. Ввести границы исследуемого интервала $[a, b]$.
3. По заданной точности δ рассчитать вспомогательное число N :

$$N = \frac{b-a}{\delta}, \text{ где } \delta \text{ — условный коэффициент } (\delta \geq 1).$$
4. Найти такое число Фибоначчи F_n , чтобы выполнялось неравенство $F_{n-1} < N \leq F_n$.
5. Определить минимальный шаг поиска $m = \left\lfloor \frac{b-a}{F_n} \right\rfloor$, где $[x]$ — ближайшее меньшее целое x .
6. Выбрать первую точку на отрезке и рассчитать ЦФ $f(a)$.
7. Перейти к следующей точке, в которой рассчитать $f(x_1)$, где точка определяется как $x_1 = a_n + mF_{n-2}$.
8. Если (шаг удачный) для новой точки $f(x_1) > f(a)$, то следующая точка определяется как $x_2 = x_1 + mF_{n-3}$; при неудачном шаге — $f(x_1) < f(a)$ имеем: $x_2 = x_1 - mF_{n-3}$.
9. Последующие шаги выполняются с уменьшающейся величиной шага, которое для i -го шага будет $\Delta x_i = \pm mF_{n-i-2}$.
10. Конец работы алгоритма.

Если при выполнении шага значение ЦФ в точке $x_{i+1} = x_i + \Delta x_i$ лучше, то следующий $(i+1)$ шаг выполняется из точки x_{i+1} : $x_{i+2} = x_{i+1} + \Delta x_{i+1}$. Если i шаг неудачный, т. е. $f(x_{i+1}) < f(x_i)$, то следующий $(i+1)$ шаг выполняется из точки x_i : $x_i = x_{i+1} - \Delta x_{i+1}$. Процесс продолжается, пока не рассмотрены все числа Фибоначчи в отрезке $[a, b]$ в убывающей последовательности или по заданию ЛПР.

Рассмотрим поиск на основе *метода золотого сечения* [109–113].

Он применяется, когда необходимо найти не собственно экстремум, а то значение аргумента функции, при котором выполняется заданное условие, например, $f(x) \geq y$. Тогда оценить заранее требуемое число измерений невозможно, поэтому нельзя определить и условия первого эксперимента для поиска по методу Фибоначчи. Метод золотого сечения позволяет избавиться от этого недостатка. Здесь также сохраняется условие симметричности последовательных экспериментов, как и в методе Фибоначчи,

т. е. справедливо соотношение (4.1). Выполняется условие

$$\frac{d_{j-1}}{d_j} = \frac{d_j}{d_{j+1}} = c = \text{const} = \frac{1 + \sqrt{5}}{2} \approx 1,618,$$

которое означает постоянство отношения длин последовательных отрезков.

При использовании метода золотого сечения первое измерение делается на расстоянии $d_2 = \frac{d_1}{c}$ от любого края интервала неопределенности ($d_1 = b - a$). Второе измерение делается симметричным первому. Поэтому места расположения точек поиска определяют по формулам:

$$x_1 = a + \left[\frac{b-a}{c} \right], \quad x_2 = b - \left[\frac{b-a}{c} \right].$$

Приведем обобщенный модифицированный нечеткий алгоритм метода Фибоначчи и метода золотого сечения:

1. Получить набор или одно альтернативное решение.
2. Ввести границы интервала $[a, b]$, число обращений к модели φ для метода Фибоначчи или ψ для метода золотого сечения. Здесь $\psi = 0,5(\sqrt{5} - 1) \approx 0,618$, $F_n = (b - a)/(\delta V_i)$, V_i — точка интервала.
3. Вычислить $V_2 = a + \lambda(b - a)$, причем $\lambda = F_{n-1}/F_n$ для метода Фибоначчи или $\lambda = \psi$ для метода золотого сечения.
4. Вычислить $F(V_2)$.
5. Вычислить $V_1 = a + b - V_2$.
6. Вычислить $F(V_1)$.
7. Если $F(V_1) > F(V_2)$, то $a := b$, $b := V_1$. В противном случае $b := V_2$, $V_2 = V_1$, $F(V_2) = F(V_1)$.
8. Если $(b - a) < \delta V_i$, конец поиска, иначе вернуться к 5.
9. Конец работы алгоритма.

Преимущество этих методов в том, что при их использовании каждый новый эксперимент приводит к сокращению интервала неопределенности.

Рассмотрим ряд **численных методов** решения инженерных задач. Сущность всех численных методов оптимизации состоит в построении последовательности векторов $\{X_k\}$, где k — шаг поиска, $k = 0, 1, 2, \dots$, удовлетворяющих условию $f(X_{k+1}) > f(X_k)$ (в случае поиска максимума) и $f(X_{k+1}) < f(X_k)$ (в случае минимума). К ним относят методы, требующие для своей реализации вычисления первых производных ЦФ $f(X)$. Известно, что градиент скалярной функции $f(X)$ в некоторой точке X_k направлен в сторону наискорейшего возрастания функции. Вектор, противоположный градиенту, $\nabla f(X_k)$ — антиградиент и направлен в сторону наискорейшего убывания функции $f(X)$. Наиболее применимы на практике следующие методы:

- релаксации;
- градиента;
- наискорейшего спуска (подъема).

Метод релаксации наиболее прост. Он заключается в определении направления, вдоль которого ЦФ изменяется наиболее быстро. Для этого в начальной точке поиска определяются производные оптимизируемой $f(X)$ по всем направлениям. Затем выбирается наибольшая по модулю производная и соответствующая ей переменная изменяется до достижения локального оптимума. В новой точке определяются производные по всем остальным переменным, и производится поиск нового локального оптимума и т. д. Скорость движения к локальному оптимуму зависит от выбора величины шага изменения независимых переменных. При «малом» шаге процесс сходится медленно, а при «большой» величине шага возможно блуждание, т. е. проскакивание оптимума. В теории поиск заканчивается, когда все частные производные ЦФ равны нулю. На практике в качестве критерия окончания поиска берется условие:

$$\sum_{j=1}^m \left(\frac{\partial f(X)}{\partial x_j} \right)^2 \leq \delta,$$

где δ — заданное заранее малое число; m — максимальный шаг поиска.

В **градиентном методе** движение при поиске точки оптимума осуществляется в направлении наибо́льшего возрастания ЦФ, т. е. в направлении градиента $\text{grad } f(X) = \nabla f(X)$. Основное уравнение градиентного поиска оптимума имеет вид:

$$x_{k+1} = x_k + h_k \text{grad } f(X_k),$$

где h_k — длина шага поиска.

Одним из вариантов градиентного метода является **равномерный поиск** при $h_k = h_0 = \text{const}$. Он гарантирует приближение к экстремуму на расстояние не более h_0 , хотя затем вызывает блуждания в окрестности экстремума. Существует еще пропорциональный градиентный поиск, когда используют определенные значения коэффициента пропорциональности α_k ($k = 0, 1, \dots$). Применяв в начале поиска произвольное или заданное значение $\alpha_0 \leq 1$, на каждом шаге его уменьшают вдвое, пока не получится $f(X_{k+1}) < f(X_k)$ (при поиске минимума). Схема такого поиска следующая [112]:

$$x_{i,k+1} = x_{i,k} + \alpha_k \frac{\Delta f(X_k)}{\Delta x_i}, \quad i = 1, 2, \dots, n \text{ — номер выбранной точки,}$$

$$\alpha_{k+1} = \begin{cases} \alpha_k, & \text{если } f(X_{k+1}) < f(X_k), \\ 0,5 \alpha_k, & \text{если } f(X_{k+1}) \geq f(X_k), \end{cases} \quad k \text{ — шаг поиска, } k = 0, 1, 2, \dots$$

Процесс можно остановить, как только приращение $\Delta_{k+1} \leq \Delta_0$ сравняется с заранее заданным. Отметим, что при использовании градиентного метода на каждом шаге меняется значение не одной, а всех независимых переменных. В каждой точке градиент вычисляется заново, а каждый шаг выполняется в направлении наискорейшего возрастания функции в соответствующей точке.

Опишем модифицированный нечеткий **алгоритм сопряженных градиентов** (Флетчера–Ривса) [110–112]:

1. Получить одно или набор альтернативных решений.
2. Ввести начальный вектор поиска V_0 , размерность n , точность ε .
3. Задать шаг поиска $i = 1$.
4. Принять вектор N_i , пропорциональный вектору изменения внутренних параметров, $N_i = 0$. Вектор $V_i = V_{i-1} - \alpha_i N_i$, где α_i — коэффициент, определяющий длину шага на каждой итерации, подбирается экспериментально.
5. Вычислить вектор $\text{grad } F(V_{i-1})$.
6. Вычислить вектор N_i .
7. Провести одномерный поиск по вектору N_i .
8. Если $i < n$, то задать $i = i + 1$ и перейти к 5, иначе к 9.
9. Если длина вектора N_i меньше ε , конец поиска, иначе $i = 1$ и перейти к 4.
10. Конец работы алгоритма.

В **методе наискорейшего подъема** объединены основные идеи методов релаксации и градиента. После нахождения градиента анализируемой функции в начальной точке осуществляется продвижение, до тех пор пока функция $f(X)$ не перестанет возрастать. При использовании метода наискорейшего подъема объем вычислений уменьшается и поиск ускоряется. Для окончания процесса используем выражение

$$\sqrt{\sum_{i=1}^n (x_{j,A} - x_{j,B})^2} < \delta,$$

где $x_{j,A}$, $x_{j,B}$ — координаты начальной и конечной точек подъема.

Приведем модифицированный нечеткий алгоритм **метода наискорейшего спуска или покоординатного спуска Гаусса–Зейделя** [110–113], когда направление шага на каждой итерации выбирается вдоль координатной оси и выполняется однопараметрическая оптимизация ЦФ поочередно по всем осям:

1. Получить одно или набор альтернативных решений.
2. Ввести V_0 , размерность n , точность ε .
3. Присвоить $i = 1$ (i — номер итерации).
4. Присвоить $j = 1$ (j — номер переменной функции $F(V_j)$).
5. Выполнить одномерный поиск ЦФ $F(V_j)$.
6. Если $j < n$, то $j = j + 1$ и перейти к 5, иначе к 7.
7. Если изменение координат меньше ε , конец работы алгоритма, иначе $i = i + 1$ и перейти к 4.
8. Конец работы алгоритма.

Теперь приведем нечеткий *объединенный алгоритм метода наискорейшего спуска* на основе вектора градиента и антиградиента [110–113]:

1. Получить одну или набор альтернативных решений.
2. Задать V_0 , размерность n , точность ε .
3. Считать $i = 1$.
4. Задать $j = 1$.
5. Вычислить составляющую вектора градиента $\partial F(V)/\partial V_j$ в точке V_{i-1} .
6. Если $j < n$, задать $j = j + 1$ и перейти к 5, в противном случае к 7.
7. Выполнить одномерный поиск по антиградиенту.
8. Если изменение координат меньше ε , конец поиска, иначе $i = i + 1$ и переход к 4.
9. Конец работы алгоритма.

Для решения инженерных задач с большим числом переменных используют *статистические методы оптимизации*. Основным их отличием от детерминированных методов является введение элемента случайности в процесс поиска экстремума. Для характеристики статистических методов оптимизации используются понятия синергетики, накопления, адаптации, самообучения. Накопление — это выборка информации на основе пробных шагов о поведении ЦФ вблизи рассматриваемой точки для выбора направления поиска. Самообучение — это управляемый процесс изменения вероятностных свойств случайности поиска в интеллектуальных ИС. Выделяют следующие основные статистические методы оптимизации [110]: простые, с накоплением, с адаптацией, с самообучением.

Рассмотрим эти статистические методы оптимизации и покажем их взаимосвязь с ЭМ. В *статистических методах оптимизации с накоплением* с помощью пробных шагов собирается информация о поведении ЦФ в некоторой окрестности точки x_k . Затем находится направление для рабочего шага, близкого к антиградиентному, причем степень близости зависит в основном от числа пробных попыток. Основной в этом классе — метод наилучшей пробы. Из точки x делается q случайных независимых попыток с заданным шагом. Для каждой пробы вычисляется ЦФ и шаг совершается в направлении той пробы, что привела к наилучшему значению ЦФ. Очевидно, что при $q \rightarrow \infty$ мы найдем оптимальное значение ЦФ. Важным отличием этого метода является возможность работы при $q < n$, где n — число переменных оптимизируемой функции.

В *статистических методах оптимизации с адаптацией* параметры поиска изменяются в процессе оптимизации. Введение автоматически изменяющегося масштаба поиска улучшает сходимость и точность метода. Масштаб поиска изменяется, чтобы на значительном расстоянии от экстремума шаг поиска увеличивался, а при приближении к нему уменьшался. Автоматическая настройка масштаба шага часто реализуется на использовании результатов последней итерации. Увеличение масштаба шага осуществляется умножением его на коэффициент роста d_1 , а уменьшение —

делением на d_2 , причем $d_1 \geq d_2 \geq 1$. Основное условие, которое необходимо соблюдать во всех случаях, это избыточность масштаба. При большом масштабе поиск становится менее чувствительным к погрешностям вычислений [110].

В *статистических методах оптимизации с самообучением* процесс случайного поиска состоит в перестройке вероятностных характеристик случайного вектора для увеличения числа эффективных итераций и уменьшения неэффективных. В таких алгоритмах единичный случайный вектор ξ перестает быть равновероятным, а в процессе поиска получает определенное преимущество в направлении наилучшего шага. Когда избранное направление не приводит к успеху, алгоритм с самообучением ищет другое. На начальных итерациях поиск эффективного направления начинается в равновероятной зоне, а затем с набором информации о характере ЦФ последовательно приобретает преимущество в выборе наилучшего направления. В общем виде такой алгоритм описывается рекуррентной формулой [110]:

$$P_k(\xi) = F[P_{k-1}(\xi) \cdot \Delta f(X_{k-1})],$$

где $P_k(\xi)$ — распределение случайного вектора ξ на k -м шаге. Оно является некоторой функцией распределения этого вектора на предыдущем шаге (P_{k-1}) вместе с приращением $\Delta f(X_{k-1})$.

Выделяют *статистические методы оптимизации координатного и непрерывного самообучения*. В первом методе вводится постоянный параметр памяти путем задания вероятности выбора удачного шага. Он определяется функцией некоторого параметра памяти W_i , причем:

$$P(W) = \begin{cases} 0, & W < -1, \\ \frac{1}{2} & \text{при } -1 \leq W < 1, \\ 1 & \text{при } W > 1. \end{cases}$$

Алгоритм самообучения реализуется путем изменения параметра памяти:

$$W_{i,k+1} = W_{i,k} - \delta \operatorname{sign}(\Delta x_{i,k} \cdot \Delta f(x_{i,k})),$$

где δ — шаг памяти ($\delta > 0$), отметим, что при $\delta = 0$ самообучения нет. Если шаг окажется неэффективным, то вероятность выбора этого направления при следующем шаге уменьшается, и наоборот. Алгоритмы такого типа, случайно определив удачное направление поиска, стараются его зафиксировать. Во втором статистическом методе оптимизации улучшается поиск при изменении градиентного направления функции. Направление случайного шага представляется в виде векторной функции $f = h\psi(\varepsilon, W)$, где h — размер шага, ξ — единичный случайный вектор, равномерно распределенный в пространстве параметров; ψ — некоторая непрерывная по модулю и направлению единичная функция двух параметров ε и W , удовлетворяющая требованиям [110]:

- $\psi(\varepsilon, 0) = \delta$, т.е. при нулевом значении обучения поиск должен быть равновероятным;

- математическое ожидание случайного шага совпадает с направлением вектора обучения;
- дисперсия случайного шага $D[\psi(\varepsilon, W)] = 1/|W|$.

Отметим, что чем больше модуль вектора памяти $|W|$, тем меньше дисперсия распределения случайного шага. При выполнении указанных условий функция реализует пространственное распределение случайного шага, причем это направление по мере накопления опыта приближается к антиградиентному.

Опишем кратко основной *метод случайного поиска* — *метод Монте–Карло* [110–113]. При этом необходимо найти экстремум $V(F)$ в заданной области допустимых параметров $V_{\min} \leq V \leq V_{\max}$ с точностью интервала неопределенности ε . Пусть допустимая область по одному из параметров $D = V_{\max} - V_{\min}$. В методе Монте–Карло вырабатывается псевдослучайный вектор параметров с равномерным распределением. Тогда вероятность непопадания в область экстремума за один шаг равна $P_1(\text{ММК}) = 1 - \varepsilon/D$. Соответственно за n шагов алгоритма

$$P_n(\text{ММК}) = (1 - \varepsilon/D)^n,$$

а вероятность попадания в область минимума

$$P_n(\text{ММК}) = 1 - (1 - \varepsilon/D)^n.$$

Следовательно, число G генераций случайного вектора, необходимых для уточнения минимума с точностью ε , с заданной вероятностью равно

$$G = \lg(1 - P_n(\text{ММК}))/\lg(1 - \varepsilon/D).$$

В случае k внутренних параметров ЦФ число генераций этого вектора увеличится в $\sqrt{k}$ раз и составит $n\sqrt{k}$. А общее число обращений к модели метода Монте–Карло за G генераций составит

$$n_{\text{ср}} = n\sqrt{k}G$$

и определит условие применимости метода при одинаковом числе итераций. Метод Монте–Карло эффективен при большом числе испытаний $G \approx 1000$. При повышенной точности ε число испытаний резко возрастает и метод Монте–Карло неэкономичен. В практических инженерных задачах необходимо комбинировать ГА, статистические и детерминированные методы поиска.

Рассмотрим *методы поиска в глубину, в ширину, с возвратом, горизонта и поиск на И-ИЛИ деревьях*. Эти методы основаны на анализе движений по дереву решений, пока это возможно. Отметим, что эти методы могут выполняться не только на основе случайного поиска, но и с использованием заданных эвристик.

При неудачном шаге метода поиска с возвратом в пространстве параметров выбирает точку x_0 и задает постоянный шаг h . Из этой точки

в случайном направлении, определенном случайным единичным вектором $\xi = (\varepsilon_1, \varepsilon_2, \dots, \varepsilon_n)$, осуществляется шаг h и вычисляется значение ЦФ в новой точке x_1 . Если при поиске минимума значение ЦФ в x_1 уменьшилось, то новый шаг из x_1 осуществляется в случайном направлении. При неудачном шаге (возрастании ЦФ) происходит возврат в начальную точку и осуществление нового пробного шага в случайном направлении. Метод поиска с возвратом можно представить в виде рекуррентного соотношения [110]

$$\Delta x_{k+1} = \begin{cases} h\xi, & f(X_k) < f(X_{k-1}), \\ -\Delta x, & f(X_k) \geq f(X_{k-1}). \end{cases}$$

Метод поиска с возвратом может выполняться в двух направлениях, определенном случайным вектором и противоположным ему с заданным шагом b . Тогда значения ЦФ определяются в точках $x_{1,2} = x_0 \pm b$ и в сторону уменьшения осуществляется рабочий шаг на величину h . После удачного шага можно продолжать поиск в том же направлении, а после неудачного — обращаться к случайному вектору, т.е. «штрафовать случайность». В этом методе из-за введения случайности движение к экстремуму производится не только вдоль координат, но и вдоль любого направления.

Комбинаторная задача, к которой применим *метод поиска с возвратом*, может быть описана следующим образом.

Даны V линейно упорядоченных блоков $B_1, B_2, \dots, B_q$ и требуется построить вектор $A = (x_1, x_2, \dots, x_n)$, где $(x_1, \dots, x_l \in B_1, x_{l+1}, \dots, x_m \in B_2, \dots, x_c, \dots, x_n \in B_q)$, удовлетворяющий заданному множеству условий и ограничений. Предположим, что найдено распределение новых $k-1$ блоков $X' = \{x_1, x_2, \dots, x_{k-1}, \dots\}$, тогда заданное множество условно ограничивает выбор следующего k -блока некоторым множеством $S_k \subset X$. Если S_k — непустое, то в качестве x_k выбирается наименьший элемент S_k и осуществляется переход к S_{k+1} и т.д. Если $S_k = \emptyset$, то происходит возврат к предыдущему этапу и удаление элемента x_{k-1} — того элемента из S_{k-1} , который непосредственно следует за удаленным.

Приведем один из возможных алгоритмов поиска с возвратом:

1. Начало положить $k = 1, S_1 = B_1$.
2. Если $S_k = \emptyset$, то перейти к 5. Иначе положить x_k равным наименьшему из элементов S_k .
3. Проверка: закончен ли поиск. Если $k < n$, то перейти к 4. Если $k = n$, то записать $B_1, B_2, \dots, B_k$ как решение. Если необходимо найти все решения, то $k = k + 1$ и перейти к 5, иначе — конец работы алгоритма.
4. Увеличить k на 1, вычислить S_k и переход к 2.
5. Если $k = 1$, то дальнейшее передвижение назад невозможно; конец работы алгоритма. При этом найдены определенные решения, либо их нет. Если $k > 1$, то уменьшить k на 1, положить $S_k = S_k - \{x_k\}$ и перейти к 2.
6. Конец работы алгоритма.

Двухнаправленный поиск моделирует процесс, при котором поиск осуществляется одновременно из начальной и конечной точки. Направленность

поиска — это степень, в которой поиск является сфокусированным в направлении цели и не блуждает по бесперспективным направлениям. Она определяется выражением:

$$\text{search} = L/|X'|,$$

где L — длина найденного пути к цели (количество просмотренных объектов), $X' \subseteq X$ — общее число объектов, определенных в ходе поиска.

Идея **метода поиска в глубину** состоит в том, чтобы в каждой исследуемой вершине дерева решений выбирать один из возможных путей

и исследовать его до конца. Другие пути при этом не рассматриваются, пока сохраняется возможность получить локальный оптимум, исследуя выбранное направление. Пример схемы поиска в глубину показан на рис. 4.2. Здесь знак (*) означает, что задача поиска решена; где $0, \dots, 9$ — исследуемые вершины дерева решений.

Основной недостаток метода поиска в глубину заключается в том, что при исследовании дерева решений с большой вероятностью можно пройти мимо той ветви, на которой раньше всего появляется локальный оптимум. Временная сложность алгоритма метода поиска в глубину $\approx O(n) \div O(n^2)$.

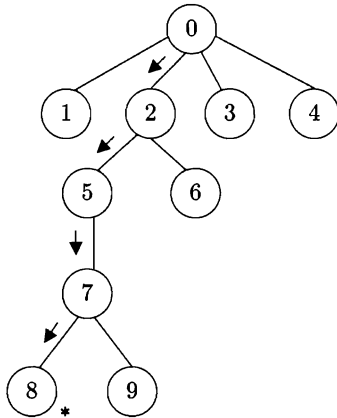


Рис. 4.2. Схема метода поиска в глубину

При использовании **метода поиска в ширину** ветвление происходит от уровня к уровню. Причем если на первом уровне начальная задача разбивается на подзадачи, то каждая из них исследуется раньше, чем задачи 2-го уровня. Задачи первого уровня, которые трудны для разрешения, разбиваются на подзадачи уровня 2, и процесс продолжается аналогично. Примерный вид дерева решений при методе поиска в ширину показан на рис. 4.3. Временная сложность алгоритма метода поиска в ширину $\approx O(n^2) \div O(n^4)$.

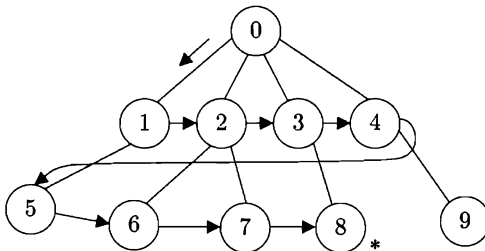


Рис. 4.3. Схема метода поиска в ширину

Рассмотрим **метод горизонта** [114]. Это метод поиска квазиоптимальных решений, заключающийся в ограничении глубины просмотра в каждой

точке дерева решений. Он позволяет достигать промежуточной цели, а затем оценивает дальнейшие возможности и продолжает или усовершенствует процесс поиска. Рассмотрим на примере основную идею реализации метода. Пусть в блоках разбиения расположено несколько вершин графа и требуется разместить еще заданное подмножество вершин. На рис. 4.4 показан граф $G = (X, U)$, $|X| = 9$, который необходимо разбить на три части, причем, частичное разбиение уже выполнено. Если для помещения в блок выбирается претендент, наиболее связанный с ранее размещенными, то нет смысла пробовать размещать его во все свободные блоки. Имеет смысл определять для размещения «перспективные» блоки и анализировать только процесс расположения в них вершин графа. Соответственно определение перспективных блоков зависит от целевой функции. При минимизации суммарного количества внешних связей можно, например, ограничить «горизонт видимости» путями длины два, три или четыре с ранее установленными вершинами.

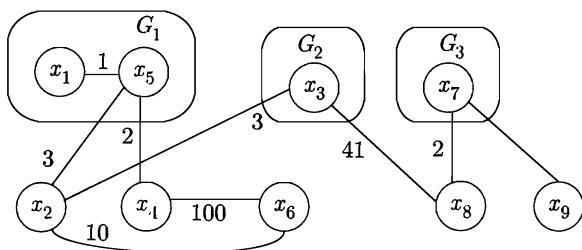


Рис. 4.4. Частичное разбиение графа G на три части

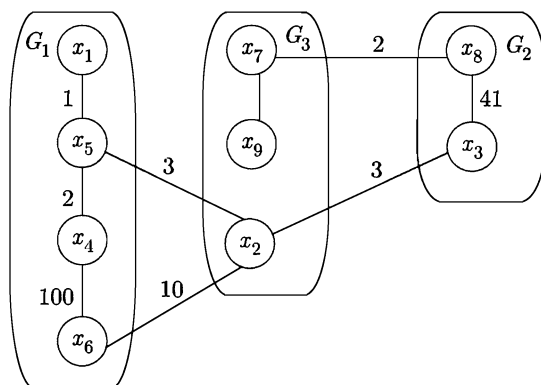


Рис. 4.5. Окончательное разбиение графа (рис. 4.4)

Например, на рис. 4.4 вершины x_1, x_5 размещены в G_1 , x_3 — в G_2 , x_7 — в G_3 . В первом блоке должно быть 4 вершины, во втором — 2, в третьем — 3. Горизонт здесь будет состоять из путей длины 2. Работу алгоритма начинаем с анализа вершин x_2, x_4 , которые связаны с x_5 . Если горизонт состоит из пути длины 1, то более предпочтительна для помещения в блок вершина x_2 .

При увеличении горизонта до двух в блок G_1 помещаем x_4 и x_6 . Продолжая аналогично, получим, что в G_2 попала вершина x_8 , а в G_3 — x_2, x_9 , при этом $K = 18$. Окончательное разбиение графа G на три части показано на рис. 4.5. Временная сложность алгоритма $\approx O(n) \div O(n^4)$. Как видно, временная сложность алгоритма и точность получения результата находятся как бы

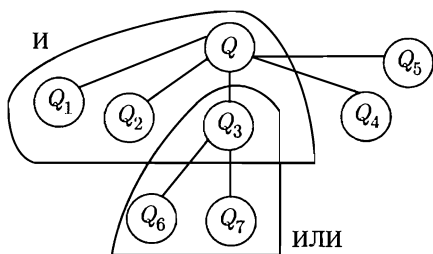


Рис. 4.6. Пример схемы поиска на основе И-ИЛИ деревьев

в противоречии, так как увеличение горизонта просмотра приводит к более качественным результатам за большее время.

При решении инженерных задач большой размерности часто применяется разбиение задачи на составляющие части. В этом случае выделяют три способа ветвления дерева решений: И; ИЛИ; комбинация И-ИЛИ [37]. Пусть у нас имеется задача Q , подлежа-

щая разбиению на подзадачи Q_1, Q_2, Q_3 (рис. 4.6). Если для решения Q необходимо решить все подзадачи Q_1, Q_2, Q_3, Q_4, Q_5 , то это ветвление И. Если решение Q_3 возможно через решения Q_6 или Q_7 , то это — ветвление ИЛИ. Примером ветвления (поиска) И является одновременное решение всех подзадач. Примером поиска ИЛИ является решение одной или другой подзадачи.

В соответствии с вышесказанным можно условно выделить три типа **поисковых методов** [17, 19, 89]:

- вычислительные — подразделяются на два основных класса:
 - А) не прямые методы — они находят локальные экстремумы, решая обычно множество линейных или нелинейных уравнений. Данные методы находят экстремум функции, анализируя ограниченное пространство точек во всех направлениях;
 - В) прямые методы — осуществляют поиск оптимума путем градиентных алгоритмов. Уже для двух локальных оптимумов это большая проблема найти лучшее решение, а для многоэкстремальных схем применение этих методов затруднительно. Сейчас применяются методы линейного, нелинейного и динамического программирования;
- невычислительные;
- случайные:
 - С) чисто случайный поиск;
 - Д) генетические алгоритмы.

Опишем модифицированный алгоритм общей нечеткой стратегии решения инженерных задач на основе **комбинированного поиска**:

1. Выбрать начальное приближение (решение инженерной задачи).
2. Определить динамику ЦФ от времени или числа элементов математической модели.
3. Определить вектор направления движения.

4. Выполнить оптимальный шаг по выбранному направлению — одномерный поиск.
5. Если ЦФ $\rightarrow \min$ (при минимизации), то перейти к 3. Иначе перейти к 6.
6. Если выходные параметры задачи удовлетворяют заданным требованиям, то окончить оптимизацию. В противном случае изменить ЦФ, весовые коэффициенты, параметры, ввести несколько уровней адаптации и перейти к 3.
7. Конец работы алгоритма.

Для определения глобального минимума ЦФ в многоэкстремальных задачах предлагается:

- использовать возможности ЭМ;
- реализовать совместные схемы поиска с ГА;
- учитывать знания о решаемых задачах, позволяющие связать возможности значения ЦФ с известными значениями в точках реализованных испытаний;
- изменять начальную точку спуска методом проб и ошибок, градиентными методами, совместными методами ЭМ и локального поиска;
- использовать различные архитектуры совместной эволюции и статистические методы поиска, позволяющие находить несколько областей локальных экстремумов.

Предлагается ряд основных стратегий взаимодействия методов поиска и эволюционного моделирования:

- стратегия «поиск–эволюция»;
- стратегия «эволюция–поиск»;
- стратегия «поиск–эволюция–поиск»;
- стратегия «эволюция–поиск–эволюция».

Заметим, что иерархически можно строить стратегии такого типа любого уровня сложности. Например, «эволюция–поиск–эволюция–поиск–эволюция–поиск» и т.д. Отметим, что такое иерархическое наращивание зависит от наличия вычислительных ресурсов и времени, заданного на получение окончательного решения.

В первом случае любым из описанных алгоритмов поиска или их комбинаций определяется одно или пара альтернативных решений задачи. На основе этих решений строится популяция, к которой применяется одна из схем эволюции. Далее процесс продолжается итерационно до достижения критерия остановки.

Во втором случае конструируется популяция и реализуется одна из схем эволюции. Лучшее решение анализируется и улучшается (если это возможно) одним из алгоритмов поиска. Далее процесс выполняется, как в первом случае. В остальных случаях процесс поиска результатов выполняется аналогично.

В заключение отметим, что в данные схемы поиска, согласно концепциям, приведенным выше, для повышения качества решений могут быть добавлены блоки адаптации к внешней среде, ЭС, иерархии различных уровней сложности, микро-, макро-, метаэволюции и др.

4.2. Архитектуры и стратегии эволюционного моделирования

Необходимым условием эффективной работы интеллектуальных ИС с использованием ГА является ее автоматическая или автоматизированная адаптивная настройка на объект исследования.

Отметим, что решение инженерных задач осуществляется последовательными, итерационными, точными или комбинированными алгоритмами [116–118]. В основном эти алгоритмы выполняют совершенствование начального варианта решения, полученного одним из случайных методов. Такой подход, кроме известных преимуществ, обладает трудноразрешимыми недостатками:

- все алгоритмы, кроме точных, часто останавливаются на локальных решениях, которые могут быть далеки от оптимальных, и с трудом осуществляют выход из локальных «ям»;
- точные алгоритмы могут работать только с очень ограниченным количеством элементов, что для инженерных задач большой размерности является недостаточным;
- в качестве исходного используется лишь один вариант решения;
- такие методы весьма чувствительны к выбору начальных условий.

Эволюционное моделирование использует в качестве начального не одно, а несколько альтернативных решений, причем в зависимости от сложности перерабатываемой информации исходные решения могут быть получены с использованием стохастических, детерминированных или комбинированных алгоритмов. Далее полученные решения будут обрабатываться адаптированными к решаемой оптимизационной задаче генетическими алгоритмами поиска с учетом синергетических и гомеостатических принципов управления. Тогда базовую структуру ГА можно представить в виде, показанном на рис. 4.7.

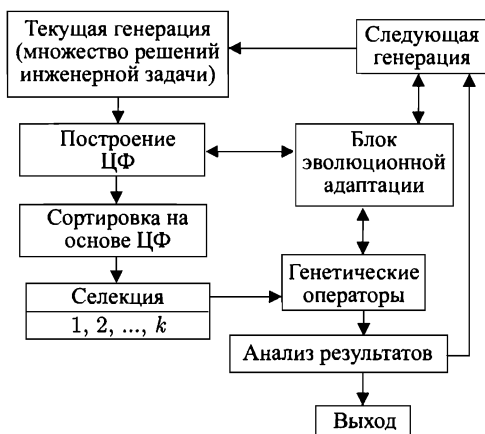


Рис. 4.7. Базовая структура ГА с учетом блока эволюционной адаптации

Здесь блок эволюционной адаптации — это специальный блок, который на основе обратных связей с учетом синергетических и гомеостатических принципов позволяет управлять процессом генетического поиска. Согласно данной схеме на первом этапе случайным, направленным или комбинированным методом получают некоторое подмножество решений P рассматриваемой задачи. Эти решения образуют текущую генерацию или популяцию исследуемых решений на шаге t ($t = 0, 1, \dots, N_G$). Далее вводится или вычисляется ЦФ. Вычисление ЦФ является очень сложной задачей, причем от точности ЦФ зависит качество будущих решений.

Отметим, что для каждой оптимизационной задачи желательно строить свою ЦФ. При построении ЦФ необходимо использовать знания о конкретной задаче. На основе ЦФ производится ранжирование и сортировка популяции решений P . Затем в результате различных методов селекции в P подбираются пары для применения различных генетических операторов. После применения операторов ОК, ОМ, инверсии, сегрегации, транслокации, удаления, вставки и т.д. получается новое подмножество решений P' . Оно объединяется с первоначальным подмножеством решений. Получается новое множество $P_{ГА} = P \cup P'$. Используя ЦФ, производится анализ $P_{ГА}$. Все элементы в $P_{ГА}$ (решения задачи), значения ЦФ которых хуже заданного порога, являются с нашей точки зрения неперспективными решениями и удаляются. Получается новое множество $P'_{ГА}$, причем $|P'_{ГА}| = |P|$. Если данное условие не выполняется, например, $|P'_{ГА}| < |P|$, то в $P'_{ГА}$ включается элемент с лучшими характеристиками из отброшенных. Множество $P'_{ГА}$ объявляется новой текущей популяцией решений и далее процесс может повторяться на основе блока эволюционной адаптации итерационно до получения подмножества или одного оптимального решения.

Заметим, что такая стратегия позволяет быстрее находить локально-оптимальные результаты. Это связано с параллельной обработкой множества альтернативных решений, причем в такой схеме возможно концентрировать поиск на получение более перспективных решений. Отметим, что периодически в каждой итерации ГА можно проводить различные изменения в перспективных, неперспективных и в других решениях. Временная сложность таких алгоритмов в основном совпадает со сложностью быстрых итерационных алгоритмов и лежит в пределах $O(n) \div O(n^3)$, где n — число входов алгоритмов. Такая сложность обещает перспективность использования ГА с блоком эволюционной адаптации при решении инженерных задач.

Отметим, что при реализации в рассмотренных ПГА существует трудно разрешимый недостаток, заключающийся в предварительной сходимости алгоритмов в локальном оптимуме, в общем случае далеком от глобального. Для устранения этого недостатка используется большое количество модифицированных генетических операторов, схем селекции и различных архитектур генетического поиска.

В последнее время появились новые нестандартные архитектуры генетического поиска, позволяющие в большинстве случаев решить проблему предварительной сходимости алгоритмов. Это — методы миграции и ис-

кусственной селекции [16–19, 119], метагенетической параметрической оптимизации, стохастически-итерационные генетические и поисковые [120], методы «прерывистого равновесия» [121], объединения генетического поиска и моделирования отжига [122].

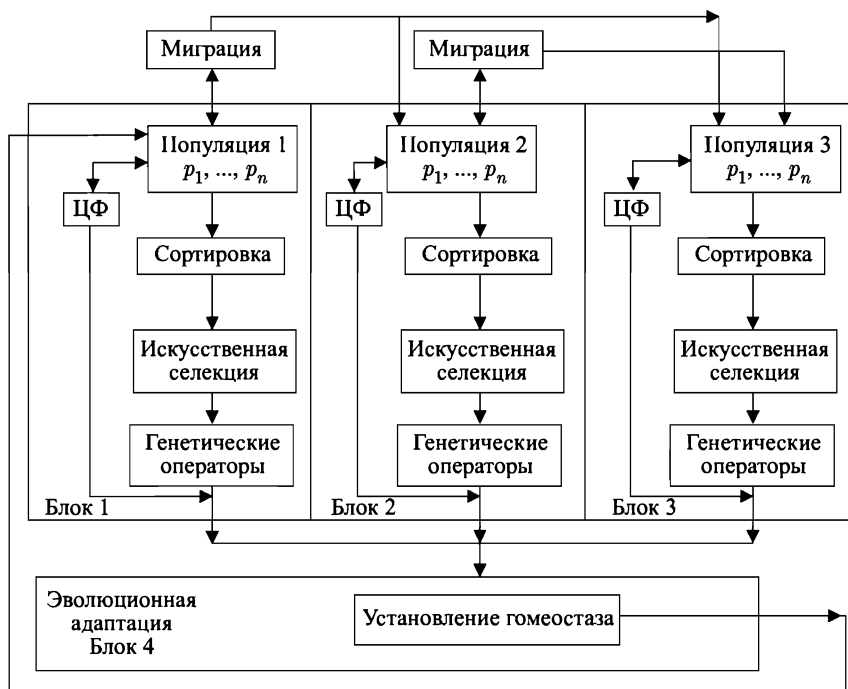


Рис. 4.8. Модифицированная схема миграции и искусственной селекции

Часто в ЭМ сначала выполняется этап макроэволюции, т. е. создается не одна популяция, а некоторое множество. Генетический поиск здесь осуществляется путем объединения хромосом из различных популяций. Опишем модифицированную архитектуру генетического поиска с миграцией и искусственной селекцией (рис. 4.8). Здесь блоки 1–3 реализуют простой ГА. Заметим, что в каждом блоке выполняется своя искусственная селекция. В первом блоке — селекция на основе рулетки. Во втором блоке используется селекция на основе заданной шкалы. В третьем блоке — элитная селекция. В блок миграции каждый раз отправляется лучший представитель из популяции. Связь между блоками 1–3 осуществляется путем последовательной цепочки 1–2, 2–3. В блоке 4 осуществляется установление баланса на основе синергетических и гомеостатических принципов и управления всем процессом генетического поиска.

Отметим, что можно организовать различное количество связей между блоками по принципу полного графа, по принципу звезды и т. д. Такая схема (рис. 4.8) в случае наличия большого количества вычислительных

ресурсов может быть доведена до N блоков, причем $N - 1$ блоков могут параллельно осуществлять эволюционную адаптацию и через блоки миграции обмениваться лучшими представителями решений. Последний блок собирает лучшие решения, он может окончить работу алгоритма или продолжить генетическую оптимизацию. Для таких схем авторы предлагают использовать платоновы графы [123, 124], т. е. правильные многоугольники, которые, как считалось в древних учениях, обладают внутренней красотой и гармонией [51–56].

На рис. 4.9, *а–д* приведены упрощенные схемы организации связей при генетическом поиске на основе платоновых графов. На рис. 4.9, *а* на основе тетраэдра; *б* — гексаэдра (куба); *в* — октаэдра; *г* — икосаэдра; *д* — додекаэдра. На рис. 4.9 использовано отношение вход-выход «1–многие». Отметим, что здесь могут быть использованы и другие отношения вход-выход: «1–1»; «многие–1»; «многие–многие». Эффективность таких отношений проверяется экспериментально.

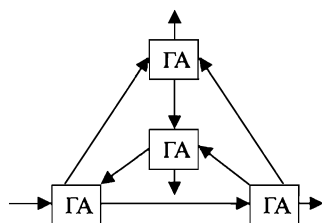


Рис. 4.9, *а*. Схема поиска на основе тетраэдра

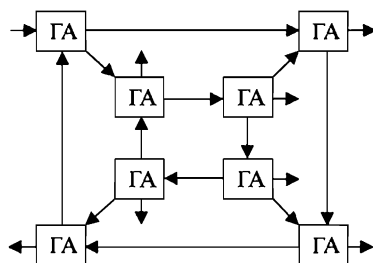


Рис. 4.9, *б*. Схема поиска на основе гексаэдра (куба)

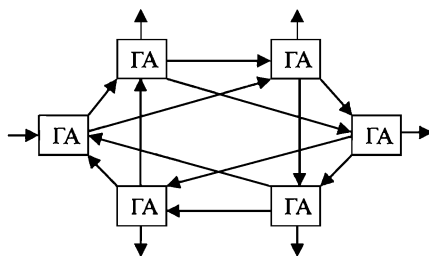


Рис. 4.9, *в*. Схема поиска на основе октаэдра

Использование платоновых графов связано со структурными моделями распространения энергии, материи и информации в пространстве. Схемы взаимосвязей блоков, использующие эти идеи, приведены на рис. 4.10, *а–ж*. На рис. 4.10, *а* приведена схема последовательного поиска. Здесь возможно разбиение популяции на подпопуляции с уменьшением времени реализации генетических операторов. На рис. 4.10, *б* приведена последовательная схема поиска с обратными связями. Введение обратных связей позволяет стабилизировать поиск и осуществлять взаимодействие с внешней средой. На

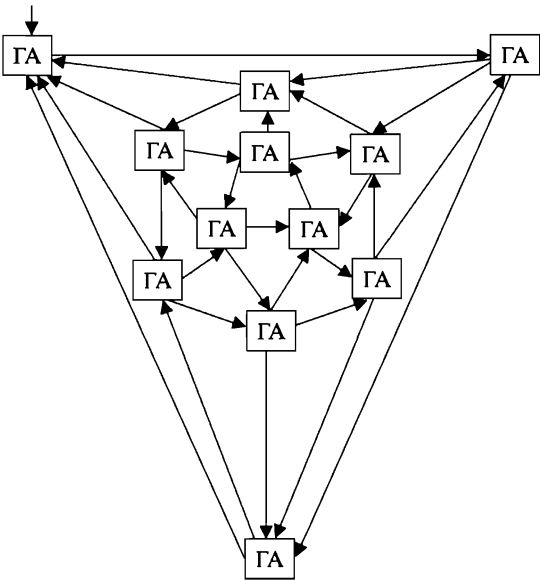


Рис. 4.9. з. Схема поиска на основе икосаэдра

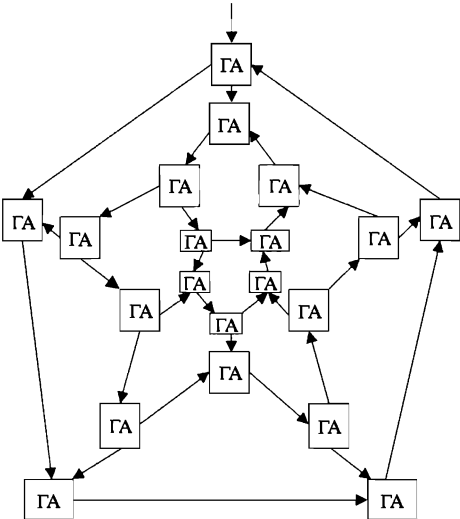


Рис. 4.9. д. Схема поиска на основе додекаэдра

рис. 4.10, в приведена последовательная схема поиска с циклами. Использование такой схемы с одной стороны вводит избыточность, а с другой — позволяет расширить область поиска решений.

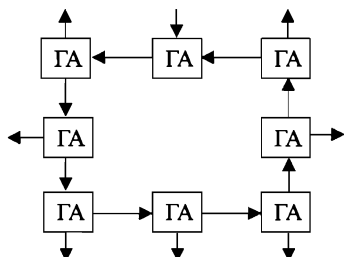


Рис. 4.10, а. Схема последовательного поиска

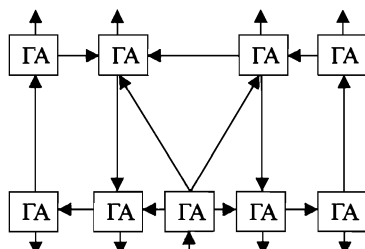


Рис. 4.10, б. Последовательная схема поиска с обратными связями

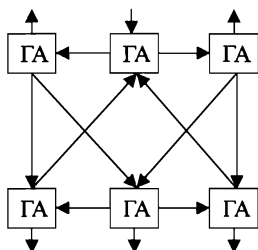


Рис. 4.10, в. Последовательная схема поиска с циклами

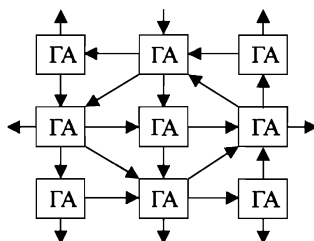


Рис. 4.10, г. Последовательная схема поиска на основе треугольников

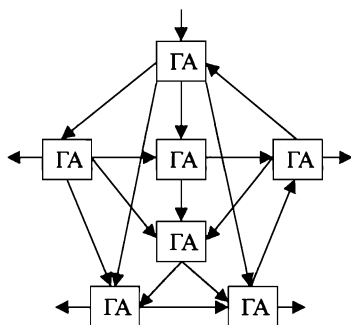


Рис. 4.10, д. Поиск на основе модифицированного полного графа на 5 вершин

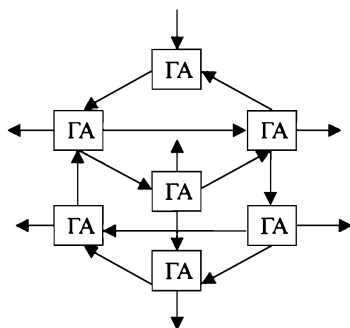


Рис. 4.10, е. Поиск на основе моделирования движения информации

На рис. 4.10, г приведена последовательная схема взаимосвязей ГА при генетическом поиске на основе треугольников. Здесь происходит моделирование идей, реализованных на основе треугольника Фреге, за счет создания устойчивых строительных блоков на высших иерархических уровнях. На рис. 4.10, д приведена последовательная схема поиска на основе модифицированного полного графа на пять вершин. Этот граф является

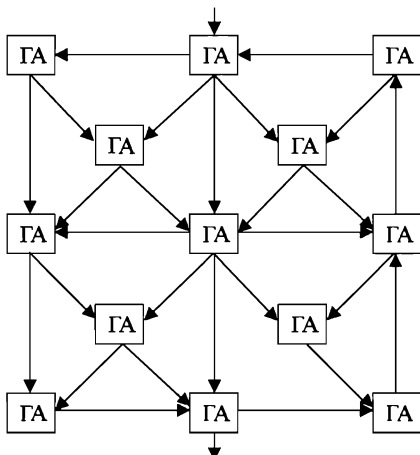


Рис. 4.10, ж. Последовательная схема поиска на основе ячеечной структуры

верхней оценкой минимально непланарного графа [123]. Он также состоит из треугольников. Такая схема генетического поиска применяется для решения комбинаторных оптимизационных задач, связанных с расположением ребер графа на плоскости. Отметим, что можно строить схемы генетического поиска на основе полных графов на любое количество вершин. На рис. 4.10, е приведена последовательная схема поиска на основе моделирования движения информационных потоков в пространстве. Она моделирует упрощенную схему реализации противоположностей ИНЬ–ЯН. На рис. 4.10, ж

приведена последовательная схема поиска на основе ячеечной структуры. Здесь также используются строительные блоки, состоящие из треугольников, а также идеи самоорганизации, построение порядка из хаоса и создание фракталов.

Такие схемы, состоящие из нескольких ГА, выполняемых параллельно, последовательно-параллельно и параллельно-последовательно, в отличие от существующих позволяют во многих случаях выходить из локальных оптимумов.

Процесс метагенетической оптимизации заключается в следующем (рис. 4.11).

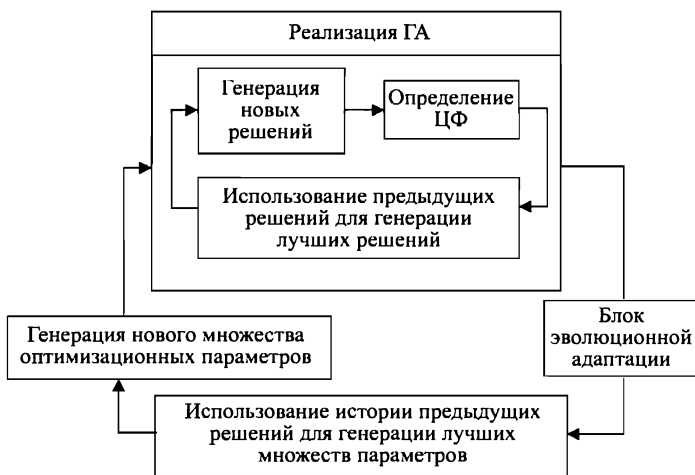


Рис. 4.11. Метагенетический оптимизационный процесс

Основным является первый блок, в котором осуществляется реализация ГА, генерация новых решений, определение ЦФ и использование предыдущих решений для генерации лучших результатов. Второй блок позволяет использовать «историю» предыдущих решений для генерации лучшего множества параметров. В третьем блоке генерируется новое множество оптимизационных параметров. Используя метагенетический оптимизационный процесс, можно случайным, направленным или случайно направленным способом генерировать начальные популяции, моделировать их и выполнять ГА на основе реализации различных генетических операторов. Можно случайно выбирать родителей из популяции с произвольной или заданной вероятностью, причем вероятность выполнения каждого оператора может определяться пропорционально его ЦФ. Окончательное множество параметров выбирается после моделирования из конечной популяции на основе блока эволюционной адаптации. Отметим, что для каждой инженерной задачи желательно строить конкретный метагенетический алгоритм.

Стохастически-итерационный метод заключается в следующем. На основе генетического поиска определяются стартовые точки для направленного поиска, причем направленный поиск осуществляется совместно с генетическими операторами на основе блока эволюционной адаптации. После нахождения стартовых точек можно параллельно использовать такие методы оптимизации, как методы золотого сечения, градиентного спуска, поиска в глубину и ширину, ветвей и границ и др.

Метод прерывистого равновесия [121] основан на палеонтологической теории, которая описывает быструю эволюцию за счет вулканических и других изменений земной коры. Он аналогичен моделям эволюции де Фриза. Для применения данного метода предлагается после каждой генерации случайным образом «перемешивать» элементы в популяции, а затем формировать новые текущие популяции, применяя к ним стандартные и модифицированные генетические операторы. Здесь можно предложить, как аналог из ЕС, бессознательный отбор родительских пар и синтетический отбор «лучших» хромосом. Далее случайным образом смешать результаты обоих отборов и не оставлять размер популяции постоянным, а управлять им на основе блока эволюционной адаптации в зависимости от наличия «лучших» элементов. Такая модификация метода прерывистого равновесия может позволить сократить неперспективные популяции и расширить популяции, в которых находятся «лучшие» элементы. Метод прерывистого равновесия — это мощный стрессовый метод изменения окружающей среды, который используется для эффективного выхода из локальных оптимумов.

В 1953 г. Метрополис предложил вычислительную процедуру, воспроизводящую механизм отжига металлов, для моделирования состояния равновесия сложных систем при заданной конечной температуре. Суть этой процедуры состоит в следующем. На каждом шаге случайным образом осуществляется малое возмущение конфигурации и вычисляется изменение ΔE энергии системы. Новая конфигурация системы принимается с вероятностью 1, если $\Delta E \leq 0$, и с вероятностью, равной $\exp(-\Delta E/kt)$, если

$\Delta E > 0$. Эта процедура легко переносится на решение дискретных оптимизационных задач, при этом состояния физической системы заменяются конфигурацией системы, изменением критерия качества, а произведение kt заменяется обобщенным понятием температуры T , которая может рассматриваться как управляющий параметр оптимизационной процедуры.

Суть переноса механизма отжига металлов на решение оптимизационной задачи состоит в том, что процесс оптимизации связывают с некоторой температурой, причем на начальном этапе температуру принимают высокой, а затем ее ступенчато снижают. При каждой температуре выполняют серию пробных перестановок элементов, и после каждой перестановки подсчитывается критерий качества. Лучшие решения принимаются с вероятностью 1, а «плохие», для которых критерий качества увеличился, принимаются с некоторой вероятностью. Такой вероятностный механизм принятия решений дает возможность, принимая в качестве исходных некоторые «плохие» решения, проскакивать через локальные оптимумы и находить глобальные.

Реализация вероятностного механизма в оптимизационном алгоритме на основе моделирования отжига осуществляется следующим образом. Если в результате перестановки элементов на дискретном рабочем пространстве произошло увеличение целевой функции F , то генерируется случайное равномерно распределенное число R в диапазоне $[0, 1]$ и проверяется условие

$$R < \exp(-\Delta F/T),$$

где ΔF — приращение целевой функции; T — температура на заданном шаге работы алгоритма. Если условие выполняется, то полученное решение принимается за новое исходное, в противном случае исходным остается старое решение.

Объединение ГА и моделирования отжига позволяют получать более качественные результаты за счет усложнения процедуры оптимизации [107, 122]. Например, на основе ПГА можно получить некоторое подмножество родителей с лучшими характеристиками и для одного из них (наилучшего) или некоторого подмножества применить оптимизационную процедуру моделирования отжига. Такое объединение можно делать различными способами. К сожалению, процедуры моделирования отжига требуют больших вычислительных затрат. Для повышения скорости такого поиска можно выбрать одну ветвь дерева решений и провести моделирование отжига, как усложненный поиск в глубину.

Перейдем к рассмотрению основных стратегий ЭМ. Комбинации теорий эволюций Ч. Дарвина и Ж. Ламарка были использованы в генетическом поиске [18, 23–25]. Было показано, что эволюция Ж. Ламарка оказывается наиболее эффективной, когда популяция имеет сходимость в область локального минимума, в отличие от стандартного ГА. Опишем схему поиска, основанную на моделях указанных эволюций. Здесь эволюция Ч. Дарвина реализуется в виде ГА, в который встроен алгоритм эволюции Ж. Ламарка. Он содержит элемент (шкала, N_G), который используется для контроля эволюции Ж. Ламарка. Шкала — это действительное число от 0 до 1, опреде-

ляющее процентное соотношение хромосом в популяции, к которым будет применена эволюция Ж. Ламарка, а N_G — число генераций ГА.

Модель эволюции Ж. Ламарка содержит пять предикатов, определяющих изменение ГА:

- селекция. Этот предикат выбирает тип селекции;
- генетические операторы. Этот предикат выбирает соответствующие генетические операторы согласно управляющим воздействиям из блока эволюционной адаптации;
- целевая функция. Предикат применяет ЦФ для выбора подходящих хромосом;
- адаптация. Этот предикат осуществляет подбор параметров генетического поиска на основе обратных связей и воздействия внешней среды;
- цикл.

На рис. 4.12 показана упрощенная модель генетического поиска при совместной эволюции Ч. Дарвина и Ж. Ламарка. Первый элемент эволюции Ж. Ламарка предусматривает случай, когда ЛПР применяет ее ко всей популяции (шкала = 1). Этот элемент собирает полное множество номеров хромосом в популяции и для них вызывает алгоритм Ж. Ламарка. Второй элемент используется, если процентное соотношение в популяции больше 0. Тогда число хромосом для процесса вычисляется по параметру процентного соотношения и размеру популяции и соответственно обрабатывается набор номеров хромосом в популяции.

Второй элемент эволюции Ж. Ламарка вызывает предикат для выбора множества номеров хромосом, лежащих в пределах от 1 до размера популяции. Для выполнения этого предикат многократно повторяется до тех пор, пока не будет набран требуемый размер. Селекция по стандартному ГА используется для выбора хромосом. Заметим, что если большой процент популяции подвергается эволюции Ж. Ламарка, то этот процесс может быть медленным, так как выбор хромосом для данной эволюции требует дополнительного времени. Если ЛПР не желает использовать эволюцию Ж. Ламарка, то выбираем шкалу равную 0.

Кроме основных введен предикат цикла, который контролирует эволюцию Ж. Ламарка перечнем выбранных хромосом. Для каждой хромосомы определяются ЦФ. Затем выполняется локализованный поиск для заданного числа итераций, определяемых параметром N_G . Результатом этого локализованного поиска является, возможно, новая хромосома с новой ЦФ. Если новая ЦФ лучше, чем у исходной хромосомы, то она заменяется оптимизированным выражением. В противном случае замены не происходит. ЛПР на основе блока эволюционной адаптации может утвердить или отменить эволюцию Ж. Ламарка. При использовании поискового алгоритма исследуется фенотип и при достижении лучших результатов соответствующий генотип (который идентичен фенотипу в этой задаче) исправляется.

Шаблон локализованного поиска является метод горизонта, «первый наилучший» или поиска в глубину. Они анализируют хромосому с лучшей ЦФ для поиска лучшего локального оптимума. Отметим, что оператор мутации разрешен для всего поискового пространства. Блок поиска по-

сле окончания итерации возвращает хромосому с лучшей ЦФ. Ее значение определяется оператором оценки ЦФ и наилучшая хромосома из старой и новой популяций используется для последующих генераций. Если получено решение с заданной ЦФ, то локальный поиск не используется.

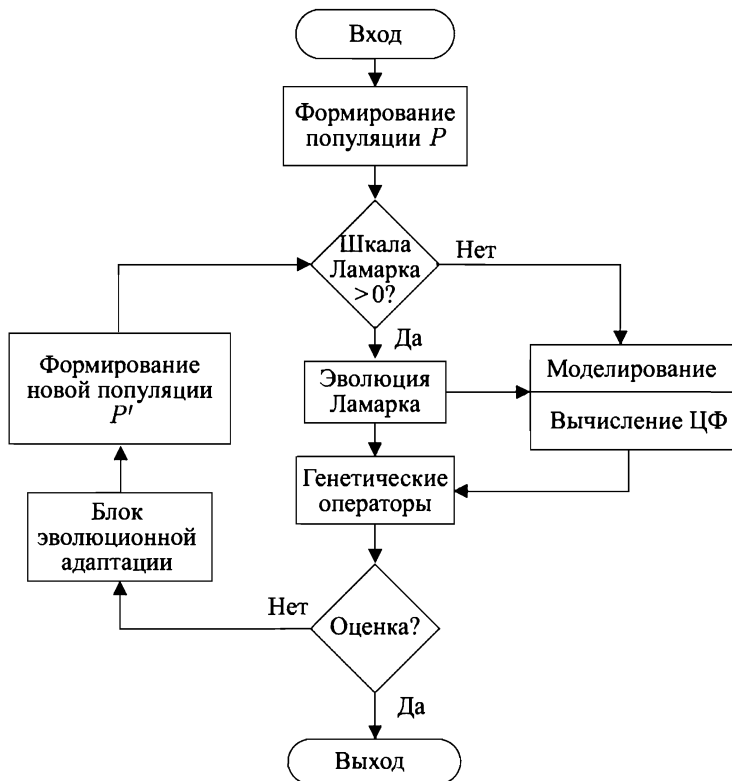


Рис. 4.12. Структурная схема совместного алгоритма эволюций

В схему совместного алгоритма эволюций можно включить любой набор поисковых алгоритмов. ГА использует гибридный устоявшийся подход, в котором генерации определяются для исследования эволюции Ж. Ламарка и накопления статистики. Это позволяет частично решать проблему предварительной сходимости алгоритмов. Экспериментальные исследования показали, что максимальное время при использовании эволюции Ж. Ламарка не пропорционально числу исследованных хромосом, и это используется при исследованиях на следующих поколениях ГА. Эффективность использования эволюции Ж. Ламарка с эволюцией Ч. Дарвина зависит от исследуемой проблемы. Локализованная оптимизация с использованием ламаркизма будет повышать среднюю ЦФ популяции и, следовательно, ускорять выполнение поиска в исходном ГА. Выполнение эволюции Ж. Ламарка может быть изменено в процессе поиска.

Рассмотрим модифицированные архитектуры поиска. Если следовать учениям древности, то теория систем макрокосм–микрокосм может быть в информационном смысле аналогична интеллектуальным ИС. Такая модель представляет две взаимодействующие противоположности. Все последующие иерархические уровни увеличивают число противоречий. Один из возможных строительных блоков, на основе которого может быть построена многоуровневая интеллектуальная ИС для решения инженерных задач, показан на рис. 4.13. Здесь P — начальная популяция альтернативных решений. На создание новой популяции P' оказывают влияние не только ГА и блок эволюционной адаптации, но и внешняя среда. Из таких строительных блоков, как из «кирпичиков», может быть построена интеллектуальная ИС любой сложности.

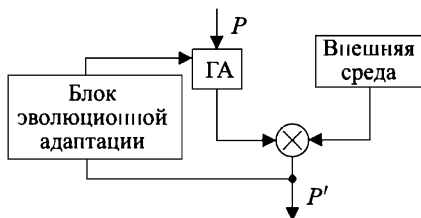


Рис. 4.13. Схема строительного блока

Отметим, что принятие решений в неопределенных, расплывчатых условиях при решении инженерных задач — это генерация возможных альтернативных решений, их оценка и выбор лучшей альтернативы [125–127]. В последнее время интерес представляют компьютерные системы поддержки принятия решения, основанные на формализации методов входных решений, вычислении ЦФ и алгоритмизации процесса выработки решений. Важное назначение систем поддержки принятия решения состоит в том, чтобы находить скрытый порядок в хаосе возможных альтернативных решений, который окружает ЛПР. Поэтому выделение некоторого подмножества решений задач относится к проблемам выбора и принятия решений.

Задачей принятия решений называют кортеж $Z = \langle N, \Theta \rangle$ (где N — множество вариантов решений задачи; Θ — принцип оптимальности, дающий представление о качестве вариантов, в простейшем случае правил предпочтения вариантов). Решением задачи α называют множество $N_{\text{оп}} \subseteq N$, полученное на основе принципа оптимальности. Задачу Z решают следующим образом. Составляют множество N , если это возможно, т. е. определяют варианты, а затем решают задачу выбора. Стандартная структура системы поддержки принятия решения состоит из следующих основных блоков [126]:

- генерация возможных альтернатив решений (сценариев);
- оценка решений (построения ЦФ));
- согласование решений, анализ динамики развития ситуации;
- выбор решения (группы решений), сценария;
- оценка соответствия принятых решений заданным целям.

Сравнивая эту структуру со структурой генетических алгоритмов можно заметить много общего. В этой связи предложенные концепции, принципы, архитектуры и ГА могут быть эффективно использованы в системах поддержки принятия решения. Предлагается комбинированная схема принятия решения с обратной связью, имеющая вид рис. 4.14. На этом рисунке

ПГА — простой генетический алгоритм, реализующий модель эволюции Ч. Дарвина. Блок эволюции Ч. Дарвина и эволюции Ж. Ламарка — это модель совместной их реализации. Эффектор объединяет свойства экспертной системы, ЛПР, блоков установления аналогий и повторителей. Компенсатор регулирует динамически изменяемый размер популяции решений, а именно, расширяя, сужая или оставляя ее постоянной. После реализации эволюции Ч. Дарвина и эволюции Ж. Ламарка компенсатор при взаимодействии с внешней средой реализует синергетические принципы, а эффектор поддерживает гомеостаз, при этом лучшие хромосомы отправляются для смешивания популяций и выхода из локальных оптимумов.

На рис. 4.15 приведена модифицированная схема принятия решения, приведенного на рис. 4.14. Здесь блок-редуктор уменьшает размер популяции, устраняя хромосомы со значением ЦФ ниже средней. Автомат адаптации должен приспосабливать свои действия, чтобы суммарный штраф при вычислении ЦФ решения был меньше заданной величины.

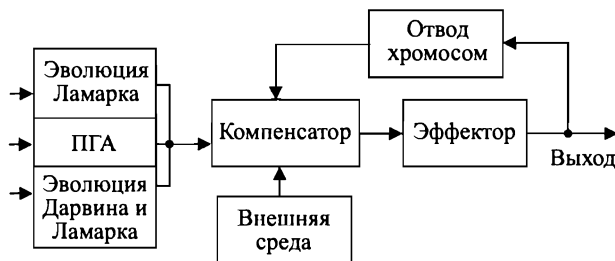


Рис. 4.14. Комбинированная схема принятия решений

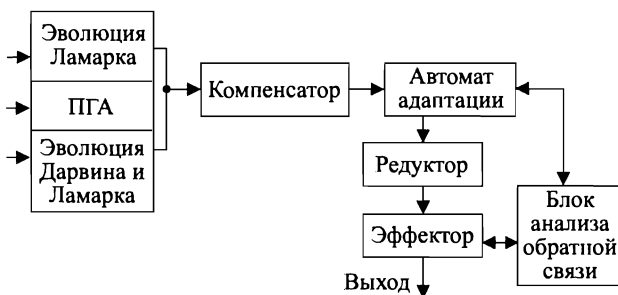


Рис. 4.15. Модифицированная схема принятия решения

На рис. 4.16 приведена архитектура принятия решения генетического поиска с использованием блока эволюционной адаптации. Блоки сумматор, редуктор и фильтр позволяют повысить эффективность реализации эволюции и скорость принятия решения. Следует отметить, что в инженерных задачах большой размерности процесс решения резко усложняется, но параллельное выполнение ГА на порядок снижает время реализации алгоритма. Схемы с подобными обратными связями присутствуют в ЕС и эффективно функционируют.

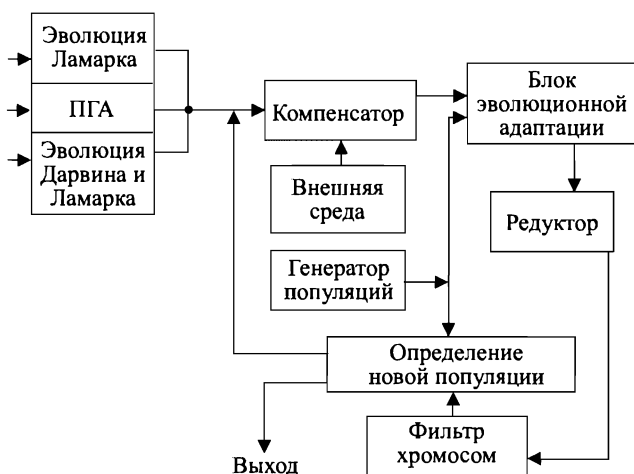


Рис. 4.16. Комбинированная схема принятия решения на основе блока эволюционной адаптации

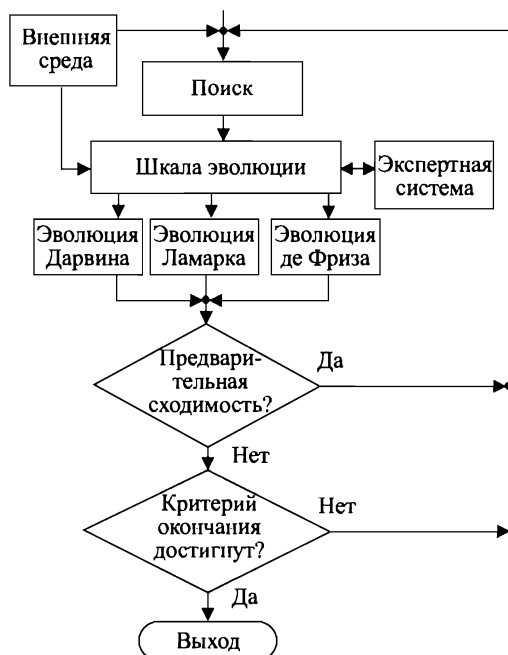


Рис. 4.17. Базовая структура генетического поиска на основе использования эволюций Ч. Дарвина, Ж. Ламарка и де Фриза

Отметим, что внешняя среда в конечном итоге определяет эволюцию, но не в простой форме связи между наследственной изменчивостью организмов и средой как в эволюции Ж. Ламарка, а в более сложной форме.

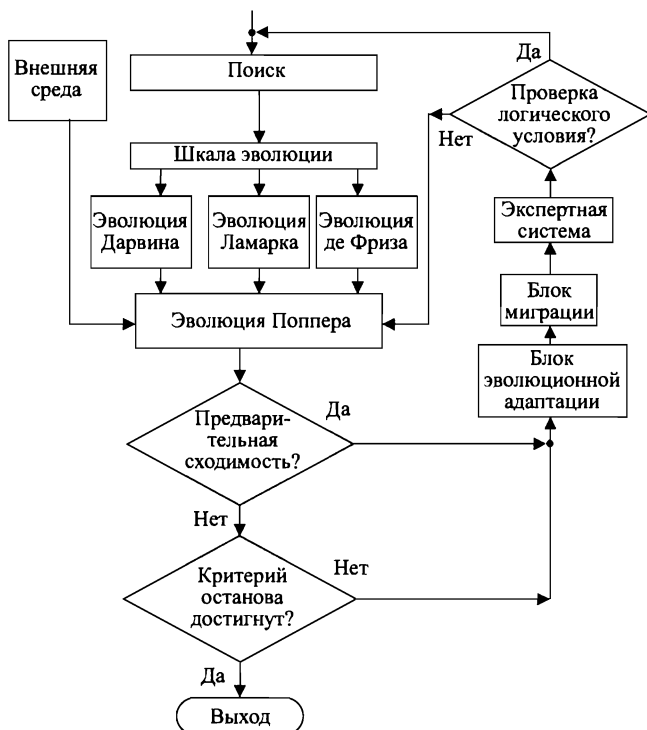


Рис. 4.18. Модификация базовой структуры

В эволюции ЕС и ИС важное значение имеют объединения всех видов и форм эволюции. На рисунке 4.17 показана базовая структура генетического поиска на основе использования моделей эволюций Ч. Дарвина, Ж. Ламарка, де Фриза. Здесь на основе шкалы эволюции при взаимодействии с внешней средой вырабатываются сигналы 0 — на выполнение эволюции Ч. Дарвина; 1 — на реализацию эволюции Ж. Ламарка; 0,5 — на реализацию эволюции де Фриза. Повторение генетического поиска возможно при предварительной сходимости алгоритма или при достижении заданного значения ЦФ. Особенностью данной схемы является использование поисковых стратегий, описанных выше.

На рис. 4.18 приведена модификация базовой структуры генетического поиска на основе использования моделей эволюций Ч. Дарвина, Ж. Ламарка, де Фриза и К. Поппера. Здесь, в отличие от базисной структуры, шкала эволюции, взаимодействуя только с внешней средой, вырабатывает сигналы на выбор эволюции Ч. Дарвина (0), Ж. Ламарка (1), де Фриза (0,5). После этого выполняется модель эволюции К. Поппера, реализую-

шая один из видов эвристического поиска в виде метода проб и ошибок. В цепи обратной связи добавлены блоки адаптации и миграции. Они позволяют производить построение порядка из хаоса, установление баланса в системе, выбор параметров для управления генетическим поиском с целью получения оптимальных и квазиоптимальных решений инженерных задач.

Отметим, что модель поиска в виде гомеостатической системы с повторением позволяет описать механизм взаимодействия интеллектуальной ИС с внешней средой. Структурные уровни интеллектуальной ИС представимы в виде параллельно-последовательно соединенных диалектических пар, образующих совокупность в виде балансной системы. Каждая пара противоположностей, являясь элементом смежных диалектических пар, тем не менее функционирует абсолютно независимо от них.

Взаимодействие противоположностей на одном структурном уровне и между смежными уровнями, как правило, подчинено методу золотого сечения. Модель интеллектуальной ИС в целом состоит из множества взаимодействующих диалектических пар (ИНЬ-ЯН). Нормальное функционирование ЕС и ИС (их возможности и свойства) связано с равновесием в них полярных элементов ИНЬ-ЯН. На рис. 4.19 показана модель взаимодействия указанных полярных элементов в процессе генетического поиска.

Данные модели позволяют сбалансировать применение моделей эволюций во взаимосвязи с полярными элементами ИНЬ-ЯН.

Для повышения качества принятия решений используют, кроме описанных, технологии генетического поиска, связанные с иерархическим распараллеливанием ГА, структурированием популяции, миграцией хромосом и популяций и т.п. Обычно используется два подхода структурирования популяции [91]:

- после конструирования популяции она случайным или заданным образом разбивается на несколько подпопуляций. Эволюция Ч. Дарвина, Ж. Ламарка, де Фриза и К. Поппера или любая их модификация применяется внутри каждой подпопуляции. При этом возможно случайное или направленное перемещение хромосом между любыми или заданными популяциями;
- после инициализации популяции для каждой хромосомы определяется ее пространственное местоположение в популяции. Далее, если необходимо, выполняется разбиение популяции. Выбор пар для использования генетических операторов зависит от «близости» решений в пространстве (значение ЦФ).

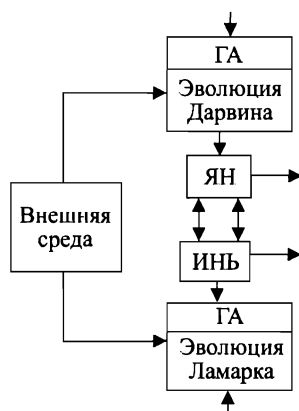


Рис. 4.19. Модель генетического поиска при взаимодействии элементов ИНЬ-ЯН

высоких значений ЦФ. В кластерно-ориентированном ГА установлен режим мутации переменных ГА, введен адаптивный фильтр, отсекающий решения с низким значением ЦФ, причем нижняя граница ЦФ адаптивна в смысле зависимости от решений на каждой генерации. В качестве модификации кластерно-ориентированного ГА предлагается начальное структурирование популяции на основе ЦФ, реализация эволюции Ч. Дарвина, Ж. Ламарка, де Фриза и К. Поппера в каждом поисковом пространстве, использование эволюции ИНЬ-ЯН и адаптации. Выбор генетических операторов здесь осуществляется на основе динамической ЭС, с использованием статистических методов оптимизации.

Дж. Шаффер и Л. Эшельман предложили проблемно-ориентированные ГА, вырабатывающие решения (фенотипы) комбинаторных задач, представленных в виде задач параметрической оптимизации, состоящих из строк параметров (генотипов) [91]. В данных ГА создается проблемно-ориентированный генератор решений (хромосом). Кроме этого, строится эвристический генератор планов, состоящий из набора шаблонов решений. Далее на основе «жадной» эвристики и ЦФ строится возможный план решения задачи. Изменяя параметры, ГА заставляет «жадные» эвристики вырабатывать большое число разнообразных хромосом. Взаимодействие ГА и эвристического генератора планов позволяет устанавливать время генерации, производить адаптацию наборов параметров, фокусировать наилучшие решения.

К. Де Йонг [91, 128] предложил в популяциях, имеющих общие хромосомы, выбирать для дальнейшего поиска $N \cdot N_G$ элементов. Здесь N_G — число генераций, а N — мощность начальной популяции. Потомки размещаются в новые популяции. Хромосомы из популяции выбираются случайно. Следовательно, при выборе размера популяции нет точных рекомендаций — большая популяция дает потери хромосом; малая популяция дает большую возможность попасть в плохой локальный оптимум.

Отношение прироста численности dN_p некоторой популяции к ее общей численности N_p называется коэффициентом прироста популяции r на заданном промежутке времени t . При постоянном значении этой величины в течение всего периода времени закон роста является линейным и приводит к экспоненциальной зависимости:

$$\frac{dN_p}{N_p} = r dt,$$

где dt — отрезок времени. При коэффициенте $r = 0,05$ популяция удваивает свою численность каждые 14 генераций. Для роста всегда существуют пределы, и приведенная зависимость справедлива на ограниченных промежутках времени.

В [11] сформулированы ограничения на рост популяции: любая внешняя среда способна обеспечить существование популяции только определенного размера, коэффициент прироста должен снижаться при приближении размера популяции к N_p . Следовательно, переменный коэффициент r сделал процесс нелинейным. Приведем пример модели роста произвольной

популяции. Пусть N_p^0 — начальная численность популяции, N_p^t — ее численность через t генераций ($t \in N_G$), коэффициент прироста по определению есть относительное изменение численности за одну генерацию ($t = 1$):

$$\alpha = \frac{N_p^{t+1} - N_p^t}{N_p^t}. \quad (4.2)$$

Пусть α — константа. Следуя [11, 64, 65, 88], сформулируем вытекающий из (4.2) закон динамики роста популяций:

$$N_p^{t+1} = f(N_p^t) = (1 + \alpha) N_p^t. \quad (4.3)$$

Через t генераций численность популяции будет

$$N_p^t \approx (1 + \alpha)^t N_p^0.$$

При $N_p^0 = 0$ и $N_p^0 = 1$ роста популяции нет. Отметим, что параметр α влияет на детерминированность и хаотичность процесса изменения популяции. Использование выражений (4.2) и (4.3) помогает планировать генетический поиск и определять критерии окончания процесса поиска.

К. де Йонг предложил 5 вариантов планирования процесса генетического поиска [91, 128]:

- элитная модель. Пусть $p'(t)$ — лучшая хромосома, сгенерированная за время t . Если после генерации $(t + 1)$ хромосома $p'(t)$ не попала в $(t + 1)$, то $p'(t)$ включается в эту генерацию как $N_p + 1$ член;
- модель с ожидаемой величиной. Здесь для селекции выбирается хромосома и вычисляется ожидаемое число потомков для каждой популяции;
- элитная модель с ожидаемой величиной. Это — объединение первой и второй моделей;
- crowding factor (CF). Когда хромосома потомок появилась после генерации ГА, она анализируется на предмет выживания. Хромосомы с ЦФ ниже средней выбираются из подмножества CF и удаляются. Обычно $CF = 2, 3, 4$;
- модель генерации генетических операторов.

Возможны новые варианты планирования генетического поиска, основанные на триедином подходе (синергетика – гомеостатика – эволюция), комбинированной эволюции, адаптации к внешней среде, использовании статистических методов оптимизации и различных поисковых стратегий.

Одной из наиболее простых технологий генетического поиска в отличие от ППА является так называемая «устойчивая репродукция» (stady-state reproduction) [90]. Она соответствует следующему алгоритму:

1. Создать k потомков (хромосом) из начальной популяции через репродукцию.
2. Убрать k членов из начальной популяции, чтобы освободить место для получаемых потомков.

3. Оценить и вставить полученные хромосомы (потомки) на освободившиеся места в популяции.
4. Конец работы алгоритма.

При оценке хромосом используются два основных подхода: линейная нормализация, определение «окна». В первом случае производится упорядочивание хромосом по убыванию ЦФ. Создается ЦФ с постоянной величиной и шкала линейного уменьшения ЦФ. Во втором случае определяется хромосома с наименьшей ЦФ. Хромосомы, у которых значение ЦФ меньше определенного, не участвуют в репродукции. Такой метод показал эффективные результаты на детерминированных проблемах [89].

Дж. Шапиро, М. Реттрей, А. Прюгель-Беннетта [91] разработали теорию, которая на основе принципов статистической механики анализирует динамику ГА при решении комбинаторных задач. Статистическая механика используется в двух эвристиках: в первой — для надежного расчета ожидаемых результатов применения генетических операторов и для расчета отклонений от ожидаемых значений: во второй — для анализа информации, полученной из макропараметров методом максимума энтропии. В [91] введено понятие корреляции хромосом, т. е. оценки, насколько «похожи» между собой хромосомы. Очевидно, что для любой популяции на основе математического ожидания и распределения ЦФ можно вычислить корреляцию. Каждому значению корреляции ставится в соответствие энтропия, равная минус логарифму от числа популяций, на которых корреляция принимает данное значение. Тогда максимум энтропии, согласно [91], соответствует наиболее часто встречающемуся значению корреляции среди всех популяций фиксированного размера. Корреляция (сходство) между хромосомами p_i и p_j популяции P , взвешенной в соответствии с относительной важностью элементов, определяется соотношением

$$q^{i,j} = \frac{1}{L} \sum_t p_i^t p_j^t,$$

где L — длина хромосом в популяции, $P = \{p_1, p_2, \dots, p_i, p_j, \dots\}$, $|P| = N_p$, t — номер генерации ГА, i, j — метки хромосомы в популяции. Тогда средняя корреляция

$$q = \frac{1}{N_p(N_p - 1)} \sum_i \sum_{j \neq i} q^{i,j}.$$

В работе [91] предлагается понятиям ЦФ и функции полезности (пригодности) (ФП) придать разный смысл. ЦФ — это средство измерения качества отдельно взятой хромосомы. ФП — описывает способ выбора хромосом из популяции. Тогда вероятность выбора хромосомы p_i определяется ее ФП F^i в соответствии с выражением:

$$P(p_i) = \frac{F^i}{\sum_{j=1, j \neq i}^{N_p} F^j}.$$

Конкретная форма отбора хромосом определяется функциональной зависимостью между ЦФ и ФП.

В [129] предложен групповой ГА с направленной мутацией. Он состоит из двух уровней. На верхнем уровне ведется групповой, а на нижнем — индивидуальный поиск. На первом этапе берется популяция размером в несколько раз больше, чем в ПГА, с целью большего охвата всего пространства поиска. Элементы в популяции оцениваются, затем хромосомы с ЦФ меньше средней отбрасываются, а из оставшихся хромосом составляются подпопуляции. Далее поиск ведется внутри отдельных групп. В отличие от [129] далее предлагается групповой ГА с направленным набором генетических операторов (ОК, ОМ, инверсии, сегрегации, транслокации, удаления, вставки и различными их модификациями) и блоком эволюционной адаптации. Приведем структуру модифицированного нечеткого алгоритма:

1. Сконструировать начальную популяцию P размера $|P| = N_p$, $p_i \in P$, $i = \overline{1, N_p}$.
2. Определить ЦФ для всех хромосом в популяции и вычислить среднюю ЦФ по формуле: $f_{\text{ср}} = \frac{1}{N_p} \sum_{i=1}^{N_p} f(p_i)$.
3. На этапе ОР (селекции) оставить в популяции хромосомы с ЦФ больше средней по всей популяции ($j = \overline{1, N_p}$), $f(p_j) > f_{\text{ср}}$.
4. С помощью шкалы эволюции на основе взаимодействия с внешней средой, блоками адаптации и ЭС выбрать модели эволюции Ч. Дарвина, Ж. Ламарка, де Фриза или К. Поппера.
5. Образовать группы хромосом по принципу их пространственной близости $|p_i - p_j| \leq \delta$, где δ — параметр соседства: $\delta = 2L_k/N_p$, где L_k — длина отрезка, на котором задана k -компонента хромосом p_i и p_j . При этом хромосомы с «малоприспособной» ЦФ удаляются, и каждая группа как бы исследует отдельный локальный оптимум.
6. Провести новую селекцию, оставив по одной хромосоме в каждой группе разбienia.
7. Применить генетические операторы ко всем хромосомам следующим образом:

$$P_i^{t+1} = \begin{cases} p_i^t + w, & \text{если } f(p_i^t + w) > f(p_i^t), \\ p_i^t - w, & \text{если } f(p_i^t - w) < f(p_i^t), \\ p_i^t & \text{в противном случае.} \end{cases} \quad (4.4)$$

Здесь w — случайное число из интервала $\left[0, \delta \left(1 - \frac{t}{N_G}\right)\right]$, где t — номер генерации ГА, $t \in \overline{1, N_G}$; N_G — заданное число генераций; δ — параметр соседства, не позволяющий хромосомам выходить из заданной группы после проведения генетических операторов. Генетические операторы реализуются, пока не будут выполнены условия остановки ГА.

8. Конец работы алгоритма.

Формула (4.4) показывает, чем больше генераций ГА, тем меньше изменений в хромосоме, т. е. шаг генетического поиска становится все меньше. При другом подходе к созданию начальной популяции на отрезке, равном длине хромосомы, задается неравномерная сетка со случайным шагом Δ , удовлетворяющим условию $\frac{0,5L}{N_p} \leq \Delta \leq \frac{2L}{N_p}$. Успех предложенного нечеткого ГА зависит от правильного выбора размера популяции N_p . Во многих оптимизационных задачах число N_p неизвестно заранее и его можно только прогнозировать, используя знания о характере функции процесса поиска или любых других знаний.

На рис. 4.21 показана укрупненная схема параллельного генетического поиска при разбиении популяции на две подпопуляции. Здесь в блоках генетических операторов выполняются операторы ОК, ОМ, инверсии, сегрегации, транслокации, удаления и вставки с учетом выражения (4.4). В блоке редукции производится удаление хромосом с ЦФ ниже средней.

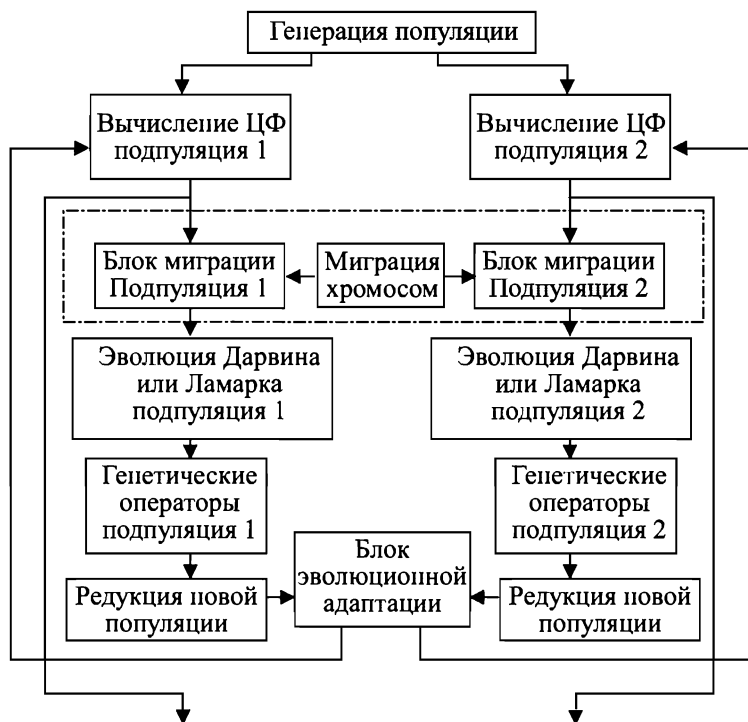


Рис. 4.21. Укрупненная схема генетического поиска

Схемы генетического поиска для решения инженерных задач — сложные системы, и по отношению к ним не может быть использована какая-то одна модель, схема, архитектура, представляющая сложную сеть взаимосвязанных элементов, свойств и функций и находящаяся в определенном

взаимодействии с окружающей средой. Такие модели не позволяют без использования моделей эволюций и систем противоречий осуществлять эволюцию по аналогии, вычленив те стороны и связи, которые дают новые знания об исследуемом объекте.

В заключение раздела отметим, что основные задачи повышения качества решений оптимизационных задач с применением ГА — это выход из локальных оптимумов. Для этого используют различные методы селекции, заданное число генераций поиска, изменение размера популяции и др. ГА в отличие от классических методов поиска, например, градиентных, не требует дифференцируемости или непрерывности функции и другой информации, кроме значения самой функции в данной точке. Кроме того, ГА эффективен при неровных пиках функций. Он с использованием приведенных методик рассчитан на поиск всех экстремумов и более эффективен, если функция имеет пики одинаковой высоты. В этой связи исследователи разрабатывают различные схемы поиска и строительные блоки, на основе которых можно получать интеллектуальные ИС различных уровней сложности. В оптимизационных задачах принятия решения находят применение нестандартные архитектуры генетического поиска связанные с метагенетической оптимизацией, а также с миграцией и искусственной селекцией. В некоторых схемах поиска для нахождения квазиоптимальных решений за приемлемое время можно использовать модели совместной эволюции Ч. Дарвина, Ж. Ламарка, де Фриза и К. Поппера.

4.3. Модифицированные генетические операторы

Рассмотрим новые технологии построения генетических операторов в общей структуре ЭМ для решения инженерных задач. С 70-х годов в эволюционных стратегиях известна глобальная рекомбинация [23–25]. Операторы рекомбинации со многими родителями до последнего времени сравнительно мало использовались для решения технических задач. Имеются три различных механизма оператора рекомбинации со многими родителями, называемые оператором рекомбинации большинства, оператором рекомбинации скрещивания и оператором рекомбинации скрещивания в среднем. Все эти операторы могут быть использованы при любом числе родителей. В таких технологиях генетического поиска используется более чем два оператора рекомбинации, причем хромосомы для реализации этого оператора выбираются случайно. Это позволяет с помощью данного оператора анализировать всю популяцию. Такой многохромосомный механизм глобальной рекомбинации является следствием перебора хромосом. Следовательно, более двух хромосом родителей могут принимать участие в построении хромосом потомков. Отметим, что заметное ускорение поиска наблюдается при переходе от половой модели размножения к неполовой модели, т. е. при использовании глобальной рекомбинации вместо варианта с двумя родителями.

Для обобщения и расширения стандартного ОК был предложен сканирующий алгоритм. Он базируется на следующей процедуре просмотра

популяции, когда хромосома-потомок строится слева направо. Модифицированная процедура сканирования имеет вид:

BEGIN

Индивидуализация (маркер на первой позиции каждой хромосомы)

For $i = 1$ **to** $i = L$

CHOOSE Выбор j из $1, \dots, r$

Определение i -й позиции родителя j

UPDATE (Модификация маркера)

END

Эта процедура определяет общие положения рекомбинации нескольких родителей и дает общее определение оператора рекомбинации, частные случаи которого определяются механизмами выбора **CHOOSE** и модификации **UPDATE**. В простейшем случае модификация **UPDATE** может сводиться к перемещению маркера на одну позицию вправо. Сканирование может быть адаптировано к упорядоченному представлению, когда каждый ген хромосомы может быть переставлен. Это гарантирует, что «лучшие» гены будут выбраны из родителей в популяции для получения потомков с оптимальной ЦФ. В зависимости от механизма выбора родителей **CHOOSE** возможны различные версии сканирования: единое сканирование (все родители имеют одинаковую вероятность быть выбранными); сканирование по ЦФ, когда выбор выполняется пропорционально целевой функции (целевое сканирование).

Можно предложить оператор рекомбинации с настраиваемым уровнем в эволюционных стратегиях на основе блока эволюционной адаптации. Параметры (N_p, L, r) эволюционных стратегий обеспечивают свободное согласование числа родителей в популяции. Параметр r определяет число хромосом и оператор рекомбинации применяется для любого данного множества r родителей. Дискретная версия выбирает случайно одну родительскую хромосому. Средняя версия дает усреднение всех родительских хромосом как хромосому потомка. Число r является независимым параметром рекомбинации. Для выполнения теоретического анализа необходимо, чтобы все родители являлись различными случайно выбранными из популяции размерности N_p . Новый оператор применяется после селекции r родителей из популяции. Выбор двух хромосом для каждого i -го случая выполняется только из r индивидов. В этом случае оригинальный механизм сохраняется неизменным, насколько это возможно, хотя вариации между двумя экстремумами $r = 2$ и $r = N_p$ возможны.

Существует ряд инженерных задач, когда при генетическом поиске большее число родителей (альтернативных решений) обеспечивает лучшие результаты. Заметим, что использование оператора рекомбинации с несколькими родителями позволяет увеличить быстродействие генетического поиска. Однако это увеличение не всегда может произойти. Поэтому эффективность поиска изучают на тестовых задачах с контролируемыми параметрами.

Качество конкретного ГА можно оценить расстоянием $D = (f_1 - f_2)/f_1$, где f_1 — практический глобальный экстремум, а f_2 — лучшее значение,

найденное этим алгоритмом. Другими словами, ГА с операторами высокого уровня имеют большие размеры популяции, что и объясняет их достоинства. Сформулируем следующие гипотезы [17].

Гипотеза 1. Использование меньшего числа точек разрыва в генетических операторах приводит к повышению быстродействия генетического поиска.

Гипотеза 2. Большие размеры популяции повышают качество ГА до определенного предела.

Гипотеза 3. Использование большего числа родителей приводит к увеличению пространства поиска и повышает вероятность получения оптимального или квазиоптимального результата.

На основании этой гипотезы приведем два следствия:

Следствие 3.1. Увеличение числа родителей от одного до двух в генетических операторах приводит к повышению качества ГА.

Следствие 3.2. Увеличение числа родителей от 2 до N_p приводит к повышению качества ГА, хотя увеличивает время поиска.

Приведем новые нечеткие алгоритмы построения генетических операторов, используя фрактальные множества, методы одномерного поиска (пассивный, последовательный, дихотомии, Фибоначчи и золотого сечения) и другие методы, описанные выше.

Введем модифицированный диагональный ОК, который является обобщением многоточечного ОК. Диагональный ОК создает r потомков от r родителей с выбранными $(r - 1)$ точками ОК и потомок формируется из r частей хромосом родителей, взятых по диагонали. Например, в популяции $P = \{p_1, p_2, p_3\}$ при реализации диагонального ОК с двумя точками разрыва будут построены три хромосомы потомка, используя диагональный принцип:

p_1 :	1	2		3	4		5	6
p_2 :	a	b		c	d		e	f
p_3 :	x	y		z	v		s	t
p_4 :	1	2		c	d		s	t
p_5 :	a	b		z	v		3	4
p_6 :	x	y		5	6		e	f

Здесь p_1, p_2, p_3 — родительские хромосомы популяции P , а p_4, p_5, p_6 — хромосомы-потомки после диагонального ОК. Такой диагональный ОК является частным случаем оператора сегрегации. Легко убедиться, что для $r = 2$ диагональный кроссинговер совпадает с одноточечным кроссинговером и в некоторых случаях обобщает многоточечный ОК с двумя родителями. Диагональный ОК иногда называют кроссинговером триады.

Опишем ОК на основе множества Кантора. Пусть начальная популяция P состоит из двух хромосом p_1, p_2 . Согласно построению множества

Кантора в p_1 , p_2 вырежем отрезки между $(1/3, 2/3)$ и произведем перестановку соответствующих генов:

p_1 :	1	2	3		4	5	6		7	8	9
p_2 :	2	4	6		8	1	3		5	7	9

p'_1 :	—	—	—	8	1	3	—	—	—
p'_2 :	—	—	—	4	5	6	—	—	—

Дальнейшее заполнение p'_1 выполняется из p_1 , слева направо, исключая повторяющиеся и вырезанные гены. Пустые позиции заполняются генами из p_2 . Аналогичная процедура выполняется и для построения p'_2 . Тогда получим:

p'_1 :	2	7	9	8	1	3	4	6	5
p'_2 :	2	5	7	4	5	6	9	1	3

Продолжая процесс итерационно, в хромосомах p'_1 , p'_2 вырежем по два отрезка, расположенных между $(1/9, 2/9)$ и $(7/9, 8/9)$, и произведем перестановку соответствующих генов. При этом получим:

p'_1 :	2	7	9	8	1	3	4	6	5
p'_2 :	2	5	7	4	5	6	9	1	3

p''_1 :	—	5	—	8	1	3	—	1	—
p''_2 :	—	7	—	4	5	6	—	6	—

p''_1 :	2	5	9	8	1	3	4	~~1~~	6
p''_2 :	2	7	9	4	5	6	3	~~6~~	1

преобразование

p''_1 :	2	5	9	8	1	3	4	7	6
p''_2 :	2	7	9	4	5	6	3	8	1

В результате использования ОК множества Кантора получили четыре хромосомы потомка p'_1 , p'_2 , p''_1 , p''_2 . Отметим, что на основе множества Кантора и ковра Серпинского можно строить любой генетический оператор.

Самыми простыми генетическими операторами являются ОК, ОМ, инверсии, сегрегации, транслокации, удаления, вставки и их модификации на основе пассивного поиска, когда случайным образом производится выбор пар и точек разрыва на исследуемых хромосомах. При использовании последовательного поиска выполняется перебор точек разрыва для нахождения хромосомы с оптимальной ЦФ. Построение генетических операторов с использованием метода дихотомии реализуется за счет механизма перебора точек разрыва.

Приведем укрупненный нечеткий алгоритм построения ОК метода дихотомии:

1. Пусть заданы две родительские хромосомы длины L .
2. Делим хромосому (отрезок L) пополам (при нечетном размере в любую часть берется ближнее большее).

3. Точка разрыва определяет точку ОК метода дихотомии.
4. По правилам построения одноточечного ОК получаем две новых хромосомы потомка.
5. Каждую половину хромосомы потомка снова делим пополам и процесс расчета продолжаем по исходной схеме: $1СБ_1 \cup 2СБ_3 \cup 1СБ_2 \cup 2СБ_4$; а для второй хромосомы потомка: $2СБ_1 \cup 1СБ_3 \cup 2СБ_2 \cup 1СБ_4$. Процесс продолжается до тех пор, пока не будет получено заданное количество хромосом потомков или метод дихотомии завершится. При получении нелегальных хромосом с повторяющимися генами последние меняются на отсутствующие гены из хромосомы родителей.
6. Конец работы алгоритма.

Рассмотрим пример. Пусть для ОК метода дихотомии выбраны две родительские хромосомы p_1, p_2 , блок эволюционной адаптации дал задание получить четыре хромосомы потомков:

$p_1: 1\ 2\ 3\ 4\ \ 5\ 6\ 7\ 8$	$p_3: 1\ 2\ \ 3\ 4\ \ 8\ 6\ \ 7\ 5$
$p_2: 4\ 2\ 1\ 3\ \ 8\ 6\ 7\ 5$	$p_4: 4\ 2\ \ 1\ 3\ \ 5\ 6\ \ 7\ 8$
$p_3: 1\ 2\ 3\ 4\ \ 8\ 6\ 7\ 5$	$p_5: 1\ 2\ 5\ 6\ 3\ 4\ 7\ 8$
$p_4: 4\ 2\ 1\ 3\ \ 5\ 6\ 7\ 8$	$p_6: 4\ 2\ 8\ 6\ 1\ 3\ 7\ 5$

Получены четыре хромосомы потомков p_3, p_4, p_5, p_6 . Здесь на втором шаге метода дихотомии для хромосомы p_3 имеем $1СБ_1 = (1, 2)$, $1СБ_2 = (3, 4)$, $1СБ_3 = (8, 6)$, $1СБ_4 = (7, 5)$, а для хромосомы p_4 $2СБ_1 = (4, 2)$, $2СБ_2 = (1, 3)$, $2СБ_3 = (5, 6)$, $2СБ_4 = (7, 8)$. Эффективность метода дихотомии экспоненциально растет с ростом числа хромосом. Количество разбиений в ОК метода дихотомии назовем глубиной дихотомии. Она обычно задается ЛПР или определяется в интеллектуальной ИС на основе адаптации. Отметим, что существует большое количество способов конкретной реализации ОК метода дихотомии. Целесообразность ОК метода дихотомии определяется на основе вероятности его выживания, которая определяется по формулам (3.3), (3.4) или их модификациям.

Рассмотрим модифицированные ОМ. В инженерных задачах, где необходимо выполнять анализ моделей с большим числом элементов ($n > 10000$) используют ОМ Монте–Карло [17, 19]. Этот ОМ заключается в единственном обмене двух случайно выбранных генов или ансамблей из нескольких генов. Кроме этого, при разбиении моделей больших размерностей на части целесообразно использовать ОМ, сохраняющий симметрию. Это единственный, сохраняющий симметрию, обмен случайно выбранного ансамбля генов с ее симметричным партнером.

Оператор мутации на основе множества Кантора заключается в перестановке генов, находящихся за точками разреза. Например:

$p_1: 1\ 2\ 3\ \ 4\ 5\ 6\ \ 7\ 8\ 9$
$p'_1: 1\ 2\ 7\ \ 4\ 5\ 6\ \ 3\ 8\ 9$

Здесь p_1 — родительская хромосома, а p'_1 — хромосома-потомок. Модификацией ОМ множества Кантора является следующая процедура: выбирается набор генов, расположенных слева от первой точки ОМ и производится его перестановка с аналогичным набором, лежащим справа от второй точки ОМ. Например:

$$p_1: 1\ 2\ 3\ |\ 4\ 5\ 6\ |\ 7\ 8\ 9$$

$$p'_1: 7\ 8\ 9\ |\ 4\ 5\ 6\ |\ 1\ 2\ 3$$

Приведем нечеткий алгоритм реализации оператора мутации на основе метода дихотомии:

1. Пусть задана родительская хромосома длины L .
2. Делим хромосому (отрезок L) пополам (при нечетном размере в любую часть берется ближнее большее).
3. Точка разрыва определяет точку ОМ метода дихотомии.
4. По правилам построения одноточечного ОМ получаем новую хромосому потомка.
5. Каждую половину хромосомы потомка снова делим пополам и процесс расчета продолжаем по исходной схеме до тех пор, пока не будет получено заданное количество хромосом потомков или ОМ метода дихотомии завершится.
6. Конец работы алгоритма.

Рассмотрим пример:

$$p_1: 1\ 2\ 3\ 4\ |\ 5\ 6\ 7\ 8 \quad p_2: 1\ 2\ |\ 3\ 5\ 4\ 6\ |\ 7\ 8 \quad p_3: 1\ |\ 3\ 2\ 5\ 4\ 7\ 6\ |\ 8$$

$$p_2: 1\ 2\ 3\ 5\ |\ 4\ 6\ 7\ 8 \quad p_3: 1\ 3\ |\ 2\ 5\ 4\ 7\ |\ 6\ 8 \quad p_4: 3\ |\ 1\ 2\ 5\ 4\ 7\ 8\ |\ 6$$

Здесь p_1 — родительская хромосома, а p_2, p_3, p_4 — хромосомы потомки. Целесообразность ОМ метода дихотомии определяется на основе вероятности его выживания, которая определяется по формуле (3.3) или ее модификации.

Построение ОМ с использованием метода Фибоначчи реализуется за счет аналогичного методу дихотомии механизма перебора точек разрыва. Приведем модифицированный нечеткий алгоритм построения ОМ с использованием метода Фибоначчи. В алгоритме используется свойство чисел ряда Фибоначчи, что очередной член ряда равен сумме двух предыдущих, кроме первого и второго.

1. В заданной популяции хромосом на основе селекции выбираем родительскую хромосому длины L с наименьшим значением ЦФ.
2. В данной хромосоме определяем точку разрыва для реализации ОМ метода Фибоначчи. Она соответствует третьему числу ряда Фибоначчи.
3. По правилам построения стандартного ОМ производим реализацию указанного ОМ и получаем новую хромосому потомка.
4. Вычисляем значение ЦФ хромосомы потомка. Если найден глобальный оптимум, то — конец работы алгоритма.

5. Далее в качестве точки ОМ метода Фибоначчи выбираем 4, 5, ... чисел ряда Фибоначчи и переходим к пункту 3. Алгоритм оканчивает работу по установке ЛПР на основе блока адаптации или когда номер числа ряда Фибоначчи $\geq L$.
6. Конец работы алгоритма.

Рассмотрим пример:

p_1 : 1 2 3 4 5 6 7 8 9	p_3 : 2 3 1 4 5 6 7 8 9
p_2 : 2 1 3 4 5 6 7 8 9	p_4 : 2 3 4 1 5 6 7 8 9
p_2 : 2 1 3 4 5 6 7 8 9	p_4 : 2 3 4 1 5 6 7 8 9
p_3 : 2 3 1 4 5 6 7 8 9	p_5 : 2 3 4 1 6 5 7 8 9

Здесь p_1 — родительская хромосома, а p_2, p_3, p_4, p_5 — хромосомы-потомки. Целесообразность ОМ метода Фибоначчи определяется на основе знания о решаемой задаче и вероятности выживания лучших решений после его применения.

Рассмотрим построение модифицированных операторов инверсии. Приведем укрупненный нечеткий алгоритм построения оператора инверсии с использованием метода Фибоначчи:

1. Пусть задана родительская хромосома длины L .
2. Точка разрыва оператора инверсии с использованием метода Фибоначчи соответствует третьему числу ряда Фибоначчи.
3. По правилам построения оператора инверсии, инвертируя правую часть от точки оператора инверсии с использованием метода Фибоначчи, получаем новую хромосому потомка.
4. Далее в качестве точки оператора инверсии с использованием метода Фибоначчи выбираем 4, 5, ... чисел ряда Фибоначчи и переходим к пункту 3. Алгоритм оканчивает работу по установке ЛПР на основе блоков адаптации и гомеостатических принципов управления или когда номер числа ряда Фибоначчи $\geq L$.
5. Конец работы алгоритма.

Рассмотрим пример:

p_1 : 1 2 3 4 5 6 7 8 9	p_2 : 1 9 8 7 6 5 4 3 2	p_3 : 1 9 8 2 3 4 5 6 7
p_2 : 1 9 8 7 6 5 4 3 2	p_3 : 1 9 8 2 3 4 5 6 7	p_4 : 1 9 8 2 3 7 6 5 4

Здесь p_1 — родительская хромосома, а p_2, p_3, p_4 — хромосомы-потомки. Целесообразность оператора инверсии с использованием метода Фибоначчи определяется на основе вероятности его выживания, которая определяется по формуле (3.5) или ее модификации.

Рассмотрим построение оператора инверсии на основе метода золотого сечения. Они аналогичны построению оператора инверсии на основе метода Фибоначчи. Для оператора инверсии метода золотого сечения предложим следующий механизм реализации. Первая точка разрыва в операторе инверсии определяется на расстоянии целой части $0,618L$ от любого края

хромосомы. Затем элементы части хромосомы, расположенные между точкой разрыва и правым концом хромосомы, инвертируются. Вторая точка разрыва в новой хромосоме определяется как ближайшее целое из выражения $(L - 0.618L) \cdot 0.618$. Далее процесс продолжается аналогично до окончания возможности разбиения хромосомы, по установке ЛПР или на основе блока адаптации. Например, пусть задана популяция родительских хромосом (альтернативных решений задачи). В этой популяции выберем хромосому p_1 с экстремальным значением ЦФ и для нее применим оператор инверсии метода золотого сечения:

$$\begin{array}{ll} p_1: A B C D E | F G H & p_3: A B | C F G H E D \\ p_2: A B C D E H G F & p_4: A B D E H G F C \\ p_2: A B C | D E H G F & p_4: A | B D E H G F C \\ p_3: A B C F G H E D & p_5: A C F G H E D B \end{array}$$

Здесь p_1 — родительская хромосома, а p_2, p_3, p_4, p_5 — хромосомы-потомки.

Рассмотрим модифицированные операторы сегрегации. Приведем укрупненный нечеткий алгоритм построения оператора сегрегации метода золотого сечения:

1. Пусть задана популяция родительских хромосом длины L .
2. Точки разрыва оператора сегрегации метода золотого сечения определяются как ближайшее целое $0.618L$ с обоих концов хромосом.
3. По правилам построения оператора сегрегации в каждой хромосоме случайным или направленным образом выбирается один из трех строительных блоков.
4. Выбранные строительные блоки соединяются в хромосому с удалением повторяющихся генов. Далее процесс повторяется аналогично.
5. Алгоритм оканчивает работу по установке ЛПР на основе блоков адаптации или когда проанализированы все строительные блоки.

Рассмотрим пример:

$$\begin{array}{ll} p_1: 1 & 2 & 3 & | & 4 & 5 & | & 6 & 7 & 8 \\ p_2: 2 & 4 & 6 & | & 8 & 1 & | & 3 & 5 & 7 \\ p_3: 1 & 3 & 5 & | & 7 & 8 & | & 6 & 4 & 2 \\ p_4: 8 & 6 & 4 & | & 1 & 3 & | & 2 & 5 & 7 \\ \hline p_5: 1 & 2 & 3 & 8 & 6 & 4 & 5 & 7 \end{array}$$

Здесь p_1, p_2, p_3, p_4 — родительские хромосомы популяции P , а p_5 — хромосома-потомок. Точки разрыва определялись как $0.618L = 0.618 \cdot 8 \approx 5$. Хромосома p_5 построена из четырех строительных блоков с удалением повторяющихся генов: $СБ_1(p_1) = (1, 2, 3)$; $СБ_2(p_2) = (8, 1)$; $СБ_3(p_3) = (6, 4, 2)$; $СБ_4(p_4) = (2, 5, 7)$. Целесообразность оператора сегрегации метода золотого сечения определяется на основе вероятности его выжива-

ния, которая определяется по формуле (3.6) или ее модификации. Отметим, что остальные модифицированные генетические операторы на основе рассмотренных методов строятся аналогично.

Важным вопросом при реализации генетического поиска является построение ЦФ. Приведем 5 основных методов построения ЦФ. В первом исходная ЦФ $f(p_i)$ для хромосомы $p_i \in P$ определяется в естественных терминах оптимизационной задачи принятия решения на основе десятичного кодирования. Например, в десятичной системе оценивается стоимость хромосомы. Во втором исходная ЦФ $f(p_i)$ для хромосомы $p_i \in P$ определяется на основе двоичного кодирования. Здесь могут быть использованы коды Грея и Хемминга [17]. В третьем случае используется стандартная ЦФ: $f_c(p_i) = f_{\max} - f(p_i)$ для случая максимизации и $f_c(p_i) = f(p_i) - f_{\min}$ для случая минимизации. Здесь $f_{\max}$ — наибольшее значение исходной ЦФ; $f_{\min}$ — наименьшее значение исходной ЦФ. Чем меньше величина $f_c(p_i)$, тем более пригодна хромосома p_i при решении оптимизационной задачи максимизации. В четвертом случае строится модифицированная ЦФ

$$f_m(p_i) = \frac{1}{1 + f_c(p_i)}.$$

Значение этой ЦФ лежит в интервале $[0, 1]$. В пятом вводится нормализованная ЦФ

$$f_n(p_i) = \frac{f_m(p_i)}{\sum_{i=1}^{N_p} f_m(p_i)},$$

где N_p — размер исследуемой популяции. Построение ЦФ производится на основе модификаций и различных комбинаций приведенных методов.

Сделаем вывод, что существуют три типа механизма создания многохромосомных операторов поиска:

- генетические операторы, базирующиеся на повторении хромосом родителей;
- генетические операторы, основанные на сегментировании и рекомбинации генетической информации родителей;
- генетические операторы, базирующиеся на численном анализе ЦФ.

В общем случае невозможно ожидать, что три различных алгоритма покажут одинаковые результаты при увеличении уровня оператора. В ЕС нет биологических аналогов рекомбинации, когда генотипы двух родителей участвуют в одном акте воспроизводства, ЭМ позволяет исследовать этот механизм.

В заключение подраздела отметим следующее. Описаны генетические операторы, использующие идеи построения фракталов, методов поиска на основе дихотомического разбиения, чисел Фибоначчи и золотого сечения и др. Преимущество всех этих методов в том, что при их использовании каждая новая генерация ГА приводит к сокращению интервала неопределенности, т. е. области поиска.

4.4. Генетическое программирование

Л. Фогель, А. Оуэнс и М. Уолш [13], исследуя возможности проектирования интеллектуальных автоматов, пришли к выводу о необходимости моделирования с использованием в качестве основных операторов поиска мутации и селекции (evolutionary programming). Проблемы компьютерного синтеза программ стали одним из направлений искусственного интеллекта примерно в конце 50-х годов.

Основные исследования в области генетического программирования проведены Д. Коза [97, 98]. Генетическое программирование представляет собой одно из направлений в ЭМ и ориентировано в основном на решение задач автоматического синтеза программ на основе обучающих данных путем индуктивного вывода. Хромосомы или структуры, которые автоматически генерируются с помощью генетических операторов, являются компьютерными программами различной величины и сложности.

Программы состоят из функций, переменных и констант. Исходная популяция P хромосом в генетическом программировании образуется стохастически и состоит из программ, которые включают в себя элементы множества проблемно-ориентированных элементарных функций, а также проблемно-ориентированные переменные и константы. Эти множества являются основой для эволюционного синтеза программы, способной наилучшим образом решать поставленную задачу. Одновременно устанавливаются правила выбора элементов из указанных множеств в пространстве всех потенциально синтезируемых программ. Понятно, что эти множества, а также правила их обработки, оказывают серьезное влияние на размерность пространства поиска наилучшего решения и на качество результатов, получаемых методами генетического программирования. Структуры генетического программирования, как правило, имеют древовидную форму.

Д. Коза в своих исследованиях по генетическому программированию применяет язык LISP, обладающий всеми необходимыми для синтеза структур генетического программирования свойствами:

- LISP является синтаксически простым функциональным языком, программа на котором представляет собой рекурсивную функцию символьных выражений, состоящих из элементарных функций, условных операторов и операторов суперпозиции;
- обработка данных в LISP-программе сводится к объединению, делению и перегруппировке информации;
- LISP-выражения представляются древовидной структурой (рис. 4.22), форма и величина которой может динамически изменяться.

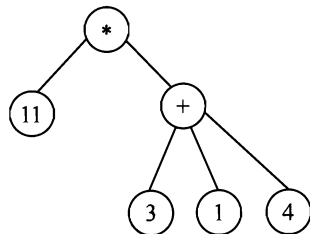


Рис. 4.22. Дерево LISP — выражения $(*11(+314))$

Язык LISP имеет свои недостатки, и применение генетического программирования нельзя связывать лишь с LISP-программированием. Следуя

Д. Коза, сформулируем следующие стартовые условия для генетического программирования [98]:

- установка множества проблемно-ориентированных переменных и констант (terminal set);
- установка множества проблемно-ориентированных элементарных функций (function set);
- определение экстремальных ЦФ;
- установка параметров моделей эволюций;
- определение критерия остановки моделирования эволюции и правил декодирования результатов эволюции.

Поскольку основой моделирования эволюции в генетическом программировании являются элементы множеств terminal set и function set, то, в этом смысле, выбор пользователем языка программирования будет в дальнейшем определять вид получаемых решений. Что касается определения ЦФ, параметров эволюции и критериев остановки процесса моделирования, то они совпадают с аналогичными этапами в других типах ГА.

В качестве элементов function sets могут фигурировать следующие [130]:

- арифметические операции (например, +, −, *);
- математические функции (например, sin, cos);
- булевы операции (например, if-then-else);
- циклы (например, for, do-until);
- некоторые специальные функции для быстрого поиска хороших решений.

Элементами множества terminal set являются константы и переменные, среди которых особое значение имеют так называемые случайные константы, с коротким временем «жизни». Речь идет о булевых константах, принимающих значения из множества $\{true, false\}$, а также вещественных константах, принимающих значение на отрезке $[-1.000; 1.000]$ с шагом 0,001. Множества function set и terminal set должны быть достаточными для нахождения решения задачи, а любая функция должна быть корректно выполнимой при любых допустимых аргументах.

Для эффективности генетического программирования форма ЦФ имеет большое значение. Общеизвестным способом оценки качества ЦФ является такой показатель, как среднеквадратичная ошибка (чем она меньше, тем лучше программа). Иногда используется критерий «выигрыша», согласно которому выигрыш определяется в зависимости от степени близости к корректному значению ЦФ. ЦФ в генетическом программировании обозначают через F_{ro} . На практике используется стандартное значение F_{st} . Обозначим через a_i некоторую программу из популяции размером N_p . Тогда стандартное значение ЦФ определяется как

$$F_{st}(a_i) = \begin{cases} F_{ro}(a_i), & \text{если наименьшее значение } F_{ro}(a_i) \text{ является} \\ & \text{подходящим,} \\ F_{\max} - F_{ro}(a_i), & \text{если наибольшее значение } F_{ro}(a_i) \\ & \text{является подходящим,} \end{cases}$$

а юстируемое значение ЦФ равно $F_{ju}(a_i) = 1/(1 + F_{st}(a_i))$.

Интервал изменения $F_{ju}(a_i)$ равен $(0, 1)$. Размер популяции N_p в генетическом программировании обычно составляет несколько тысяч программ. Для максимального числа генераций $t_{\max}$ рекомендации отсутствуют. Д. Коца в своих экспериментах использует значение $t_{\max} = 51$ [98].

Рассмотрим подробнее процедуру генетического программирования.

1. **Инициализация.** На этом этапе стохастически генерируется популяция P , состоящая из N_p древовидных программ, причем корневой вершиной дерева всегда является функция, аргументы которой выбираются случайно из множеств *function set* или *terminal set*. Концевыми вершинами дерева должны быть переменные или константы, в противном случае процесс генерации необходимо рекурсивно продолжить. Если структура дерева становится сложной, то заранее устанавливается максимальная высота дерева, равная числу ребер дерева, которое содержит самый длинный путь от корневой вершины до некоторой концевой вершины. В экспериментах Д. Коца максимальная высота дерева колеблется от шести для популяции P^0 до 17 в более поздних популяциях P^i . Для обеспечения многообразия популяции P^0 в ходе инициализации Д. Коца предлагает применить способ, согласно которому деревья разной высоты генерируются с одинаковой частотой. Правда, способ не лишен недостатка, связанного с не совсем случайным характером генерируемых деревьев [98].

2. **Оценка решений.** На втором этапе оценивается ЦФ каждой программы в P^0 указанным выше способом. Поскольку все программы выбраны случайно, то большинство из них будут иметь ЦФ далекую от лучшего решения, поэтому в качестве оценки можно взять разницу между лучшим и худшим значением ЦФ в популяции P^0 .

3. **Генерация новой популяции.** Этот этап принято разделять на следующие подэтапы.

3.1. **Выбор операторов генетического программирования.** Основными операторами здесь являются репродукция и кроссинговер, применяемые с вероятностью $P(OP)$ и $P(OK)$ соответственно, причем $P(OP) + P(OK) = 1$ (чаще всего $P(OP) = 0,1$, $P(OK) = 0,9$).

3.2. **Селекция и рекомбинация.** Данный этап моделирования выполняется по схемам, аналогичным ГА [88–92].

3.3. **Образование новой популяции.** Если к некоторой программе применяют оператор репродукции, то эта программа копируется в новую популяцию. Для проведения ОК выбираются две родительские хромосомы (программы), случайным образом определяются точки кроссинговера и путем обмена образуются два потомка (рис. 4.23). Здесь слева расположены два родителя и справа — два потомка. При программной реализации на языке LISP ОК сводится к обмену списками между двумя программами при сохранении синтаксической корректности вновь получаемых программ.

4. **Проверка критерия остановки.** Процедура генетического программирования является итерационной, и критерии ее остановки аналогичны критериям для обычных ГА.

В качестве примера, иллюстрирующего рассмотренную процедуру, возьмем проблему двух параллелепипедов. Пусть два параллелепипеда задаются шестью независимыми переменными $L_1, B_1, H_1, L_2, B_2, H_2$ и одной зависимой переменной D (рис. 4.24).

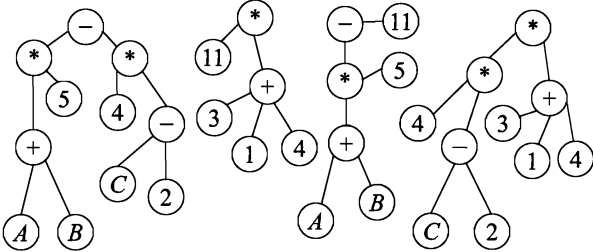


Рис. 4.23. Реализация ОК

В табл. 4.1 представлены 10 различных вариантов исходных значений для шести независимых переменных. а также разность объемов $D = L_1 * B_1 * H_1 - L_2 * B_2 * H_2$. Пусть для получения корректных значений величины D установлены следующие

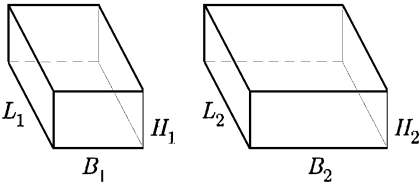


Рис. 4.24. Два параллелепипеда

стартовые условия: terminal set $T = \{L_1, B_1, H_1, L_2, B_2, H_2\}$; function set $F = \{+, -, *, /\}$; F указывает абсолютную ошибку, определяемую разностью между действительным значением D и тем значением, которое получается программно: $F_{st} = F_{ro}$; выигрыш

определяется числом случаев, когда сравниваемые величины D различаются менее, чем на 0,01; $N_p = 4000$; программа останавливается, если число выигрышей равно 10, либо $t_{max} = 51$.

Таблица 4.1

	L_1	B_1	H_1	L_2	B_2	H_2	D
1	3	4	7	2	5	3	54
2	7	10	9	10	3	1	600
3	10	9	4	8	1	6	312
4	3	9	5	1	6	4	111
5	4	3	2	7	6	1	-18
6	3	3	1	9	5	4	-171
7	5	9	9	1	7	6	363
8	1	2	9	3	9	2	-36
9	2	6	8	2	6	10	-24
10	1	10	7	5	1	45	-155

В результате моделирования эволюции для популяции P^0 лучшая ЦФ имела значение 783, что соответствует следующей форме:

$$(*(-(-B_1 L_2)(-B_2 H_1)))(+(-H_1 H_1)(* (H_1 L_1)))$$

или

$$H_1 L_1 (B_1 + H_1 - B_2 - L_2).$$

Видно, что в данном выражении отсутствует переменная H_2 и оно не соответствует корректному решению. Далее, лучшие ЦФ в популяциях $P_1 - P_6$ имели следующие значения: 778, 510, 138, 117, 53, 51. Лучшая программа на восьмом этапе эволюции дала значение ЦФ, равное 4,44, что соответствовало LISP-выражению следующего вида:

$$(-(-(* (B_1 H_1) L_1) (* (L_2 H_2) B_2))) \\ (\% (+ B_1 L_1) (- (- L_1 B_2) (+ (+ B_2 L_2) (* L_2 B_2))))$$

или, в упрощенном виде,

$$B_1 H_1 L_1 - B_2 H_2 L_2 - (B_1 + L_1) / (L_1 - 2B_2 - L_2 - L_2 B_2).$$

Видно, что, за исключением последнего ошибочного термина, данное выражение соответствует корректному решению. Наконец, на 11-м шаге эволюции была получена корректная программа вида

$$(-(* (B_1 H_1) L_1) (* (L_2 H_2) B_2))$$

или

$$L_1 B_1 H_1 - L_2 B_2 H_2.$$

Рассмотрим перспективные направления исследований в области генетического программирования [130]. К ним прежде всего следует отнести работы по так называемым автоматически определяемым функциям (ADF), идея которых состоит в повышении эффективности генетического программирования за счет модульного построения программ, состоящих из главной программы и ADF-модулей, генерируемых в ходе моделирования эволюции. При этом до начала эволюции ориентировочно определяется архитектура программы, число ADF-модулей и параметры (аргументы) ADF. На рис. 4.25 представлена общая структура LISP-программы, состоящей из главной программы и двух модулей ADF0 и ADF1.

Все типы вершин данной структуры имеют свой номер, причем вершины с номерами от 1 до 6 являются инвариантами по отношению к генетическому программированию в отличие от вершин с номерами 7, 8, 9. Общая LISP-функция `progn` определяет последовательно (слева направо) каждый свой аргумент. Каждая подпрограмма декларируется как функция `defun`. Список аргументов включает отдельные локальные переменные, которые определяются при вызове ADF. Наконец, задание `values`-функции главной программы завершает установку общей программы `progn`. Настройка ADF (их число, аргументы и т.п.) зависит от решаемой задачи, имеющихся вычислительных ресурсов и предварительного опыта. Отметим также необходимость типизации вершин дерева, иначе при выполнении оператора кроссинговера могут быть синтезированы синтаксически некорректные решения. Это связано с применением в генетическом программировании

с ADF модифицированной формы ОК, исключающего появление некорректных решений.

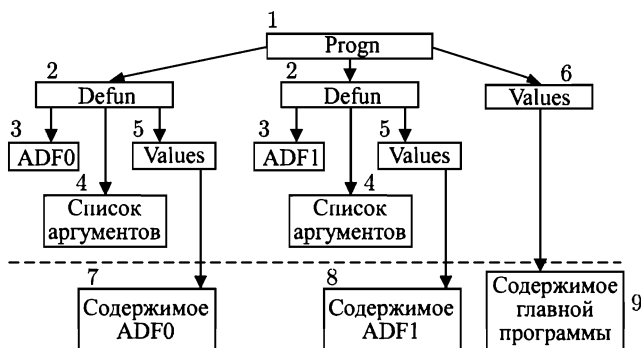


Рис. 4.25. Структура LISP-программы

Вернемся к ранее рассмотренной проблеме двух параллелепипедов и приведем результаты моделирования эволюции с использованием ADF. Определим, что число ADF равно единице (ADF0), а число аргументов равно трем (ARG0, ARG1, ARG2). Тогда множества terminal set и function set для главной программы, соответственно равны $\{L_1, B_1, H_1, L_2, B_2, H_2\}$ и $\{+, -, *, \%, ADF0\}$, а множества terminal set и function set для ADF0 имеют, соответственно, вид $\{ARG0, ARG1, ARG2\}$ и $\{+, -, *, \%\}$. Лучшая из 4000 случайно сгенерированных программ популяции P^0 имела значение ЦФ, равное 1142, что соответствовало следующему LISP-выражению:

```
(progn(defun ADF0(ARG0 ARG1 ARG2)
  (values(%(*(%(-ARG2 ARG0)(%ARG2 ARG2))ARG0)
    (*(*(-ARG1 ARG0)(+ARG2 ARG1))(ARG2))))
  (values(-(*B2B2)(*B2(*(-H2H1)L2))))))
```

или

$$B_2^2 - B_2L_2(H_2 - H_1),$$

которое не соответствует корректному решению.

После 6 генераций значение ЦФ улучшилось с 1101 до 96, причем лучшая программа имела вид

```
(progn (defun ADF0(ARG0 ARG1 ARG2)
  (values (-(-ARG0 ARG0)(*ARG0 ARG1)
    (%ARG2(%ARG2 ARG2))))(values (-(+(- (ADF0 L2B2H2)
      (ADF0B1H1L1))L1)(+(-H2H2)(-(%L1L1H2))))))
```

или

$$B_1 H_1 L_1 - L_2 B_2 H_2 + L_1 - 1 + H_2.$$

В следующей популяции P^7 лучшая программа имела ЦФ, также равную 96, однако решение было другим. Наконец, на 13-й генерации была получена корректная программа

```
(progn (defun ADF0(ARG0 ARG1 ARG2)
  (values (-( *ARH2 ARG0)(*(+ARG0 (*ARG0 ARG1))
    (%ARG2(%ARG2 ARG2)))))(values (-(ADF0 L2B2H2)
  (ADF0 B1H1L1))))).
```

Согласно закону Оккама, если получается несколько различных корректных решений, то лучшее из них — решение, обладающее наименьшей структурной сложностью. Другой важный показатель — вычислительная сложность программы. Сравнительный анализ генетического программирования с ADF и без ADF показывает, что для задачи двух параллелепипедов с точки зрения вычислительной трудоемкости и структурной сложности преимущество — за генетическим программированием без ADF. Однако имеется целый ряд примеров, свидетельствующих об обратном. Рассмотрим один из них.

Проблема проверки на четность пяти имеет следующую формулировку. Пусть булева функция, зависящая от пяти аргументов (D_0, D_1, D_2, D_3, D_4) принимает значение *true*, если четное число аргументов (включая нуль) принимают значение *true*, в противном случае булева функция принимает значение *false* (константа, обозначающая список, в котором нет ни одного элемента); 32 возможных комбинации аргументов служат основой для оценки генетического программирования с ADF и без ADF.

Для генетического программирования без ADF устанавливаются следующие начальные условия и параметры эволюции: terminal set $T = \{D_0, D_1, D_2, D_3, D_4\}$; function set $F = \{\text{AND}, \text{OR}, \text{NAND}, \text{NOR}\}$; ЦФ определяется числом совпадений значений исходной функции со значениями, выдаваемыми генетическим программированием; $F_{st} = 32 - F_{ro}$; $N = 16000$; программа останавливается, если число совпадений равно 32.

Аналогичные условия и параметры устанавливаются также для генетического программирования с ADF. Отличие генетического программирования с ADF от генетического программирования без ADF состоит в следующем: архитектура программы включает главную программу и две ADF (ADF0 и ADF1), каждая из которых содержит 4 параметра, причем ADF1 может включать функцию ADF0; terminal set главной программы совпадает с множеством T в генетическом программировании без ADF; function set главной программы равно $\{\text{ADF1}, \text{ADF0}, \text{AND}, \text{OR}, \text{NAND}, \text{NOR}\}$; terminal set для ADF1 равно $\{\text{ARG0}, \text{ARG1}, \text{ARG2}, \text{ARG3}\}$; function set для ADF1 равно $\{\text{ADF0}, \text{AND}, \text{OR}, \text{NAND}, \text{NOR}\}$; terminal set для ADF0 равно $\{\text{ARG0}, \text{ARG1}, \text{ARG2}, \text{ARG3}\}$; function set для ADF0 равно $\{\text{AND}, \text{OR}, \text{NAND}, \text{NOR}\}$.

Результаты моделирования свидетельствуют о преимуществе генетического программирования с ADF как по структурной сложности решений, так и по вычислительной трудоемкости. Другим перспективным направлением в генетическом программировании считается вопрос о применении в качестве оператора поиска не только ОК, но и ОМ, а также реализацию генетического программирования на транспьютерных вычислительных системах.

В [131], исследуя решения различной длины, получаемые с помощью генетического программирования, обратили внимание на так называемый эффект «компрессионного давления», которым обладают решения с малой структурной сложностью. Программы, генерируемые по методу генетического программирования, могут содержать «лишние» блоки, не влияющие на функциональные возможности программы и на значение ее ЦФ. В генетике это соответствует понятию интрона, который является нечувствительным к кроссинговеру [6,7]. С учетом этого в [131] вводится понятие эффективной сложности, величина которой равна величине структурной (абсолютной) сложности программы за вычетом величины интронов, содержащихся в программе. Принято считать, что применение ОК носит деструктивный характер, если ЦФ ухудшается.

Следовательно, для программы с относительно большим значением структурной сложности, но содержащей много интронов, опасность деструктивного воздействия кроссинговера уменьшается. С учетом этого, а также принимая во внимание фундаментальную теорему ГА [88], приведем следующие рассуждения. Обозначим через $C_e(a_i)$ эффективную сложность программы $a_i = (i = 1, 2, \dots, N_p)$, через $C_a(a_i)$ абсолютную сложность программы a_i , через $P_s(\text{ОК})$ — вероятность выживания решения после применения ОК, а через $P_d(\text{ОК})$ — вероятность того, что применение ОК приведет к деструктивному эффекту, причем $P_d(\text{ОК}) = 0$, если ОК проводится с интроном. Пусть $F(a_i)$ — значение ЦФ программы a_i , $\bar{F}(t)$ — среднее значение ЦФ всех программ в популяции P^t . Если применяется пропорциональная селекция, то нижняя оценка $A_i(t+1)$ «доли» программы a_i в популяции $P^{(t+1)}$ примерно равна

$$A_i(t+1) \cong A_i(t) \cdot F(a_i) / \bar{F}(t) \cdot (1 - P_s(\text{ОК}) \cdot C_e(a_i) / (C_a(a_i) \cdot P_d(\text{ОК}))) = \\ = (F(a_i) - P_s(\text{ОК}) \cdot F(a_i) \cdot C_e(a_i) / C_a(a_i) \cdot P_d(\text{ОК})) A_i(t) / \bar{F}(t).$$

Выражение в скобках представляет собой определение эффективной ЦФ:

$$F_e(a_i) = F(a_i) - P_s(\text{ОК}) \cdot F(a_i) \cdot C_e(a_i) / C_a(a_i) \cdot P_d(\text{ОК}),$$

которая соответствует вкладу программы a_i в следующую популяцию $P^{(t+1)}$. Отсюда следует, что репродуктивные шансы некоторой программы a_i тем выше, чем меньше отношение между эффективной и абсолютной сложностью программы. Достигнуть этого можно двумя путями.

Первый заключается в увеличении абсолютной сложности путем добавления интронов, второй — в поиске простых решений. Эмпирические данные, полученные в [131], подтверждают приведенные выше соображения:

- на ранних этапах эволюции среднее значение ЦФ от популяции к популяции изменяется сильно, в то время как F_e изменяется относительно медленно, а репродуктивные свойства некоторой программы зависят прежде всего от ее ЦФ;
- затем темп изменения ЦФ уменьшается, однако начинает расти соотношение между эффективной и абсолютной сложностью, «компрессионное давление» возрастает, и F_e уменьшается;
- на заключительных этапах эволюции экспоненциально растет абсолютная сложность программы за счет интронов, F_e остается на относительно низком уровне, среднее значение продолжает улучшаться, а деструктивное влияние ОК уменьшается.

Теоретически обоснованное и эмпирически наблюдаемое явление «компрессии» ведет к преждевременной сходимости генетических программ к субоптимальным решениям. В [131] предлагается ввести внешнее управление этим процессом с помощью некоторого коэффициента D , пропорционального абсолютной сложности программы. С учетом этого выражение для F_e принимает вид

$$F_e(a_i) = F(a_i) - D \cdot C_a(a_i) - P_s(\text{ОК}) \cdot F(a_i) \cdot C_e(a_i) / C_a(a_i) \cdot P_d(\text{ОК}).$$

Применение фундаментальной теоремы ГА для описания динамики эволюционного процесса является упрощенным подходом, поэтому исследуются альтернативные подходы, которые базируются на анализе статистических и динамических особенностей различных форм представления хромосом в популяции.

Таким образом генетические алгоритмы, генетическое программирование, эволюционные стратегии и эволюционное программирование являются основными формами эволюционного моделирования. Каждая из этих форм имеет свои отличительные черты и особенности. С практической точки зрения было бы неплохо установить, для каких конкретно инженерных задач лучше подходит та или иная форма моделирования эволюции.

ГЛАВА 5

ГИБРИДНЫЕ СИСТЕМЫ

Истина — это то, что выдерживает проверку опытом.
А. Эйнштейн

5.1. Генетические алгоритмы и имитационное моделирование

Сложные системы, управление сложными системами, системность, модели — эти термины в настоящее время все чаще встречаются и используются практически в любой сфере деятельности человека. Это связано, прежде всего, с обобщением накопленного опыта и результатов в различных сферах человеческой деятельности и естественным желанием найти и использовать некоторые общесистемные принципы и методы. Именно системность решаемых задач в науке и практике может стать той базой, которая позволяет работать исследователю с любой сложной системой, независимо от ее физической сущности и ограниченности рамками определенной науки или ряда наук [29–132].

Весь цикл разработки и эксплуатации любой сложной системы носит итеративный характер (рис. 5.1). Выполнение любой итерации, как показано на рисунке, проводится с использованием моделей сложной системы. Наиболее продвинутым и мощным аппаратом построения соответствующих моделей для рассматриваемых систем является имитационное моделирование. Оно обеспечивает глубокое представление моделируемого объекта, дает возможность анализа процессов на любом временном интервале, позволяет учитывать случайные и неопределенные факторы, оценивать как технические, так и экономические показатели функционирования системы [133–134].

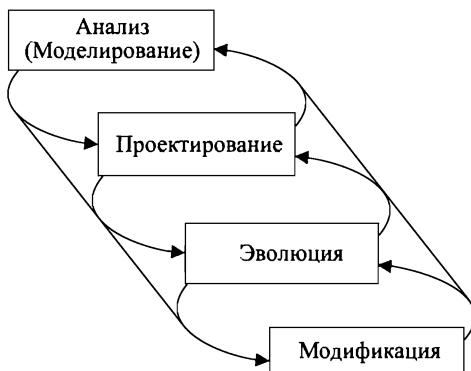


Рис. 5.1. Цикл разработки сложной системы

Сложная система, как было показано в первой главе, представляет собой систему с эволюцией. Любая действительно сложная система характеризуется большим числом гетерогенных подсистем с высокой степенью неопределенности. Следовательно, решение задач анализа, управления и других в таких системах не может быть осуществлено в рамках использования какого-либо единого подхода для всех подсистем. Для принятия решений обычно используют сложное сочетание математических, статистических, вычислительных, эвристических, экспериментальных методов и методов инженерии знаний (чаще всего экспертных систем). Комплексное использование указанных методов и средств обеспечивает пользователя поддержкой при принятии решений. При этом имеет место приоритет решаемой задачи над используемыми методами.

Существование подобной ситуации, когда необходимо совместно использовать имитацию и различные методы принятия решений, привело к появлению так называемых гибридных систем. Под гибридной системой будем понимать систему, состоящую из нескольких систем различного типа, функционирование которых объединено единой целью [135–136].

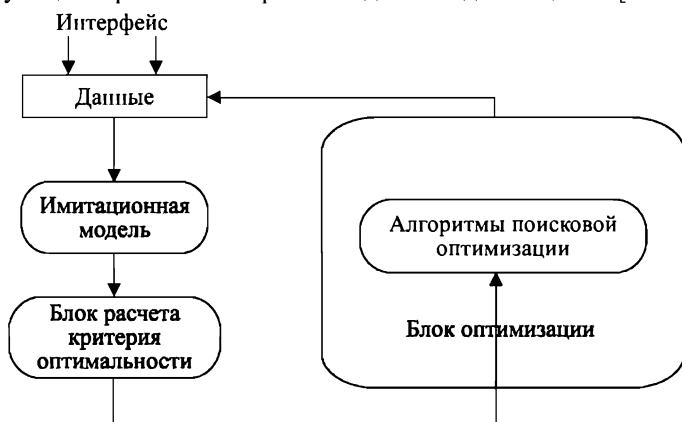


Рис. 5.2. Простейшая гибридная система

Простейшей гибридной системой является система, объединяющая в себе имитационную модель и блок оптимизации. Блок оптимизации реализует один из алгоритмов поисковой оптимизации (например, простейший генетический алгоритм, рассмотренный в третьей главе), а имитационная модель служит для вычисления значений критерия оптимизации (функции пригодности) для выбираемых вариантов решения (рис. 5.2). Прогон имитационной модели обеспечивает, в лучшем случае, получение результатов в одной точке пространства поиска решений. Поэтому требуется реализация серии экспериментов на имитационной модели в большой области поиска, целенаправленность которых обеспечивается в традиционных системах моделирования специалистом-разработчиком.

Использование генетических алгоритмов для решения оптимизационных задач при анализе, управлении или синтезе действительно сложных

систем (а именно для таких систем их применение оказывается наиболее эффективным) возможно лишь в том случае, если имеется способ определения функции пригодности особи с достаточно хорошей точностью. То есть, необходимо иметь возможность разрабатывать модели сложных систем с высокой степенью адекватности объектам и процессам реального мира. Когда речь идет о сложных системах, то можно утверждать, что в настоящее время существует единственный способ для построения ее модели или модели ее части — имитационное моделирование. Все другие способы имеют достаточно узкие рамки применения и часто настолько упрощают описание сложной системы, что о получении с их помощью эффективной оценки функции пригодности решения речь уже не идет. Поэтому далее рассмотрим гибридные системы, использующие совместно ГА и имитацию при решении задач различного типа (рис. 5.3). Это прежде всего относится к задачам организационного управления, принятия решений в реальном масштабе времени, оценки стратегий управления, прогнозирования.



Рис. 5.3. Простейшая гибридная система с генетическим алгоритмом и ИМ

Существующие методы и языки имитационного моделирования часто оказываются неэффективными по причине своей низкой гибкости и сложности моделирования систем принятия решений (даже если речь идет не об оптимальных, а просто эффективных решениях) и управления, особенно если система управления включает в себя человека-оператора, принимающего решения. Использование появившихся на рынке программных продуктов интеллектуальных систем имитационного моделирования снимает часть этих трудностей и предоставляет новые возможности при использовании имитации в гибридных системах для решения прикладных системных задач.

Для реализации имитационных моделей, включаемых в гибридную систему, используем интеллектуальную имитационную среду РДО (Ресурсы—Действия—Операции) [134]. Интеллектуальное имитационное моделирование характеризуется возможностью использования методов искусственного интеллекта и, прежде всего, знаний при принятии решений в процессе

имитации, при управлении имитационным экспериментом, при реализации интерфейса пользователя, создании информационных банков моделей, использовании нечетких данных и др.

Формализмы Ресурсов, Действий и Операций составляют основу РДО-метода. Его основные положения можно сформулировать следующим образом.

- Все элементы моделируемой сложной системы представлены как ресурсы, описываемые некоторыми параметрами. Ресурсы могут быть разбиты на несколько типов; каждый ресурс определенного типа описывается одними и теми же параметрами, состав которых он наследует от типа.
- Состояние ресурса определяется вектором значений всех его параметров. Часть значений параметров или все значения отдельный ресурс наследует из типа, к которому он принадлежит. Состояние моделируемой системы характеризуется значением всех параметров всех ее ресурсов.
- Процесс, протекающий в моделируемой системе, описывается как последовательность целенаправленных действий и нерегулярных событий, изменяющих определенным образом состояния ресурсов; действия ограничены во времени двумя событиями: событиями начала и конца.
- Нерегулярные события описывают изменения состояния моделируемой системы, непредсказуемые в рамках продукционной модели (влияние внешних по отношению к моделируемой системе факторов либо факторов, внутренних по отношению к ресурсам). Моменты наступления нерегулярных событий случайны.
- Действия описываются операциями, которые представляют собой модифицированные продукционные правила, учитывающие временные связи. Операция описывает предусловия, которым должно удовлетворять состояние участвующих в операции ресурсов, и правила изменения состояния ресурсов в начале и конце соответствующего действия. Модифицированная продукция представляет собой конструкцию: ЕСЛИ (условие) ТО1 (событие 1) ЖДАТЬ (выражение для вычисления временного интервала) ТО2 (событие 2). Традиционное продукционное правило ЕСЛИ (условие) ТО (событие), которое является частным случаем модифицированной продукции, так же может использоваться (рис. 5.4).

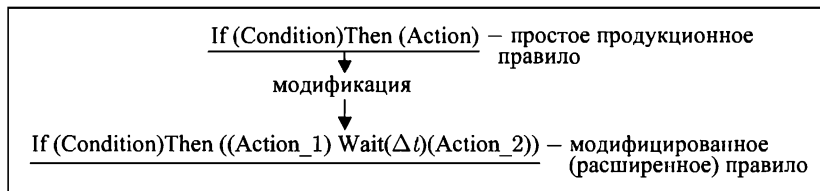


Рис. 5.4. Модифицированное продукционное правило

- Множество ресурсов R и множество операций O образуют модель сложной системы.

Введенная в РДО-методе модификация продукций позволяет устранить недостатки продукционных систем, связанные с их статичностью, сохраняя в то же время известные преимущества продукционных систем: универсальность (возможность описания широкого класса сложных систем применительно к различным задачам исследования); гибкость (простота настройки); независимость формата продукционного правила и механизма поиска решений от физического смысла представляемых знаний; модульность (продукционные правила независимы друг от друга, что позволяет вводить или удалять правило из базы знаний, не затрагивая остальные); соответствие процедурного характера описания знаний в продукционных системах дискретным процессам, имеющим место в сложной системе, что позволяет естественно использовать их для построения последовательности некоторых действий.

Таким образом, имитационная модель в РДО-методе представляет собой динамическую продукционную систему. Базой данных (БД) этой продукционной системы является множество ресурсов R , базой знаний — множество операций O . Создание имитационной модели заключается в формализованном описании ресурсов и операций на некотором языке и введении их в БД и БЗ.



Рис. 5.5. Представление сложной системы в РДО-методе

В РДО-методе можно указать взаимно однозначное отображение реальной сложной системы в ее информационное представление в интеллектуальном обеспечении. Основным составляющим сложной системы, каковыми являются ее элементы, производственный процесс, законы функционирования, соответствуют следующие информационные объекты: ресурсы, действия и нерегулярные события, операции (рис. 5.5). Из указанных элементов

интеллектуального обеспечения. множества ресурсов и операций образуют модель сложной системы. Процесс — временная последовательность действий A и нерегулярных событий $\tilde{E}$.

Основными элементами являются динамическая производственная система и аппарат событий. Действия иницируются системой вывода, а нерегулярные события имитируются специальным блоком. При имитации состояние системы изменяется в соответствии с описанием нерегулярного события либо действия, которое началось или завершилось. После любого изменения состояния, т. е. при каждом событии, вызывается система вывода. Она просматривает в БЗ все операции и проверяет по предусловиям, могут ли они начаться. При нахождении таких операций иницируются события начала соответствующих действий. Итак, производственная система (БД, БЗ и система вывода), система имитации нерегулярных событий и аппарат ведения событий совместно осуществляют построение модели процесса. Система трассировки выводит подробную информацию о событиях в специальный файл, который затем обрабатывается для детального анализа процесса и представления информации в удобном виде. Система анимации позволяет отображать на экране во время моделирования поведение моделируемой системы.

Поскольку традиционные производственные правила являются частным случаем модифицированных, они также могут быть записаны на РДО-языке и использованы в процессе вывода. Это означает, что на РДО-имитаторе реализуются также и гибридные системы, включающие экспертные системы, имитационные модели и алгоритмы оптимизации.

5.2. РДО-модель простого генетического алгоритма

Рассмотрим реализацию имитационной модели простого генетического алгоритма (ПГА) на языке РДО. Иными словами, выполним описание на языке РДО основных генетических законов, отдельных особей, их популяции, а затем проведем имитацию процесса эволюции.

Следует отметить, что генетический алгоритм может быть реализован на любом универсальном языке, например, С++, Паскаль и др., или на специальном языке для программирования генетических алгоритмов проще, чем на РДО в силу универсальности последнего. Однако гибридная система, построенная на едином программном обеспечении, по многим причинам предпочтительнее, чем система, объединяющая блоки, написанные на разных программных средствах.

5.2.1. Задача оптимизации

Рассмотрим простую задачу оптимизации: найти значения переменных X_1 и X_2 , при которых достигается максимум значения функции $f = (X_1^2 + X_2^2)/2$. Область определения переменных: $X_1 \in [0, 1]$, $X_2 \in [0, 1]$, т. е. пространство, в котором осуществляется поиск оптимума, представляет собой квадрат со стороной 1, расположенный у начала координат в первой четверти (рис. 5.6). Как следует из вида функции, максимальное значение

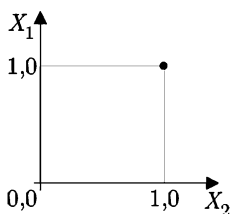


Рис. 5.6. Область оптимизации

достигается в точке с координатами (1, 1) и оно равно 1 (точка на рисунке).

Для решения задачи с помощью ПГА введем следующее представление переменных X_1 и X_2 . Эти переменные кодируются в виде битовой строки длиной 16 бит каждая. Значения переменных определяются следующим образом: битовая строка рассматривается как двоичное целое без знака в диапазоне $[0; 2^{16}-1]$, это число делится на 2^{16} . Следовательно, получаемое при этом значение лежит в диапазоне $[0; 1-2^{-16}]$, а пространство оптимизации становится дискретным (что неизбежно при использовании ПГА) с шагом сетки 2^{-16} . Максимальное значение оптимизируемой функции f равно:

$$f_{\max} = \frac{(1 - 2^{-16})^2 + (1 - 2^{-16})^2}{2} = (1 - 2^{-16})^2 \approx 0,99997.$$

Битовую строку-хромосому представим четырьмя байтами (всего 32 бита). Значения переменных вычисляются по формулам:

$$X_1 = \frac{B_1 \cdot 2^8 + B_2}{2^{16}}, \quad X_2 = \frac{B_3 \cdot 2^8 + B_4}{2^{16}},$$

где B_i — значение i -го байта, рассматриваемое как двоичное целое без знака.

5.2.2. Типы ресурсов и ресурсы

Для реализации ПГА используются ресурсы трех типов:

- тип_Система — всего имеется один такой ресурс, использующийся для хранения общей и вспомогательной информации;
- Поколения — используемые для хранения общей для одного поколения информации;
- Особи — это ресурсы, представляющие собой отдельные особи рассматриваемой популяции.

Рассмотрим параметры ресурсов всех типов.

тип_Система

- Номер_поколения — хранит номер поколения, обрабатываемого в данный момент;
- Размер_поколения — хранит размер поколения (количество особей);
- Вероятность_скрещивания — параметр ПГА;
- Вероятность_мутации — параметр ПГА, задает вероятность, с которой происходит мутация в одной особи;
- Режим — хранит текущий режим (этап) работы ПГА;
- Счетчик — вспомогательный параметр, использующийся при работе с особями в различных режимах; содержит номер обрабатываемой

в данный момент особи при генерации первого поколения, вычислении значения функции пригодности, воспроизведении и мутации, либо номер обрабатываемой в данный момент пары при скрещивании;

- Число — вспомогательный параметр, используемый при воспроизведении, скрещивании и мутации, с помощью которого определяется номер родителя, факт скрещивания в данной паре или мутации в данной особи;
- Родитель_1, Родитель_2 — хранят номера пары особей, подобранных для скрещивания;
- Скрещивание_байт, Скрещивание_бит — определяют место разрыва при скрещивании (номер байта и номер бита в этом байте);
- Мутация_байт, Мутация_бит — определяют место мутации (номер байта и номер бита в этом байте);
- Счетчик_бит — вспомогательный параметр, с помощью которого для текущего поколения подсчитывается количество единичных битов, стоящих на определенном месте в битовой строке.

Поклоения

- Номер — хранит номер поколения, обрабатываемого в данный момент;
- Размер — количество особей в данном поколении;
- Сумма_значений_ФП — накапливает сумму значений функции пригодности для всех особей данного поколения;
- Среднее_значение_ФП — содержит среднее значение функции пригодности для данного поколения;
- Минимальное_значение_ФП — содержит минимальное значение функции пригодности для данного поколения;
- Максимальное_значение_ФП — содержит максимальное значение функции пригодности для данного поколения;
- Количество_скрещиваний — содержит количество скрещиваний в данном поколении;
- Количество_мутаций — содержит количество мутаций в данном поколении.

Особи

- Параметр_1, Параметр_2, Параметр_3, Параметр_4 — эти параметры (в общем случае их количество может быть другим) хранят собственно битовую строку-хромосому; Параметр_1 хранит значение B_1 и т.д.;
- Номер_поколения — номер поколения, к которому принадлежит данная особь;
- Номер_особи — номер особи, служит для идентификации особи;
- Родитель_1, Родитель_2 — номера особей из предыдущего поколения, в результате скрещивания которых получена данная особь; если особь получена в результате просто воспроизведения без скрещивания, то Родитель_1 содержит номер особи, в результате воспроизведения которой получена данная особь, а Родитель_2 имеет значение ноль;
- Количество_потомков — количество потомков в следующем поколении, полученных в результате воспроизведения данной особи;

- Скрещивание_байт, Скрещивание_бит — определяют место разрыва при скрещивании родителей (номер байта и номер бита в этом байте); если особь получена в результате просто воспроизведения без скрещивания, то эти параметры равны нулю;
- Мутация_байт, Мутация_бит — определяют место мутации (номер байта и номер бита в этом байте); если мутация не имела место в данной особи, то эти параметры равны нулю;
- Статус — вспомогательный параметр, использующийся при скрещивании; он хранит информацию о том, рассматривалась ли уже данная особь в качестве кандидата на скрещивание;
- Значение_ФП — значение функции пригодности для данной особи;
- Сумма_до, Сумма_после — эти два параметра используются для реализации взвешенного выбора особей при воспроизведении: Сумма_до содержит сумму значений функции пригодности для всех особей, рассмотренных до данной. Сумма_после равна Сумма_до плюс значение функции пригодности для данной особи.

Первоначально в модели имеется один ресурс Система, хранящий общую и вспомогательную информацию. Для тестовых прогонов выбраны следующие значения параметров ПГА: размер поколения — 20 особей, вероятность скрещивания — 0,6, вероятность мутации на одну особь — 0,3.

5.2.3. Основные параметры

Основными параметрами описываемого ПГА являются следующие:

- длина битовой строки-хромосомы — задается двумя константами: Количество_байтов_в_строке и Количество_битов_в_последнем_байте, поскольку для хранения строки использованы байтовые параметры ресурсов;
- количество поколений, которые необходимо сгенерировать — задается константой Количество_поколений;
- размер популяции (размер поколения) = количество особей в популяции — задается параметром Размер_поколения ресурса Система (в данной реализации ПГА размер популяции не изменяется в процессе ее эволюции, т. е. он одинаков для всех поколений);
- вероятность скрещивания — задается параметром Вероятность_скрещивания ресурса Система;
- вероятность мутации — задается параметром Вероятность_мутации ресурса Система;
- количество последних поколений, для которых модель хранит все особи для возможного анализа (увеличение этого количества увеличивает время вычислений) — задается константой Сколько_поколений_хранить.

5.2.4. Схема работы

Модель ПГА выполняет итерации, включающие расшифровку битовой строки и расчет значения функций пригодности для всех особей, воспроизведение, скрещивание и мутацию. Отдельный блок предназначен для гене-

рации первого поколения. В рассматриваемом примере программа заканчивает работу, сгенерировав заданное число поколений (задается константой `Количество_поколений`).

Для описания ПГА использовано 12 продукционных правил. Каждый блок, кроме блока скрещивания, реализован двумя правилами, блок скрещивания — тремя. Еще одно правило добавлено для уничтожения старых поколений по мере генерации новых, что позволяет уменьшить время вычислений. Это последнее правило записано так, что всегда сохраняются несколько последних поколений для возможного детального анализа эволюции популяции (количество сохраняемых поколений задается константой `Сколько_поколений_хранить`).

В каждом блоке, описанном двумя правилами, одно правило реализует собственно действия данного блока над очередной особью, а другое переключает режим работы ПГА, если действия данного блока произведены над всеми особями. В блоке скрещивания используется та же схема, но основное действие блока реализуется двумя правилами.

Для реализации общего цикла ПГА и циклов по особям внутри каждого блока используются два параметра ресурса Система, а именно — Режим и Счетчик. Рассмотрим эту схему подробнее на примере блока генерации первого поколения. В начальный момент значение параметра Счетчик равно нулю, параметра Режим — Инициализация. Правило Генерация_первой_популяции создает одну за другой особи первого поколения. Чтобы это правило применялось, необходимо одновременное выполнение трех условий:

- текущий номер поколения должен быть меньше или равен заданному количеству поколений (это условие присутствует почти во всех правилах и введено для остановки ПГА, когда сгенерировано требуемое количество поколений; остановка происходит потому, что ни одно правило не может быть применено, когда текущий номер поколения превысил требуемое количество поколений);
- текущий режим работы ПГА должен быть равен Инициализация;
- значение параметра Счетчик должно быть меньше заданного размера поколения, это говорит о том, что еще не все особи первого поколения сгенерированы:

Choice from Общий.Номер_поколения $\leq$ Количество_поколений *and*
Общий.Режим = Инициализация *and*
Общий.Счетчик $<$ Общий.Размер_поколения

Здесь Общий — это имя релевантного ресурса для ресурса Система (это имя используется во всех правилах).

В конверторе правила значение параметра Счетчик увеличивается на 1.

Это правило применяется последовательно до тех пор, пока не перестанет выполняться третья часть предусловия, т. е. пока значение параметра Система.Счетчик не станет больше или равно значению параметра Система.Размер_поколения. Таким образом будет создано нужное количество особей первого поколения.

Предусловие второго правила блока генерации первого поколения *Конец_генерации_первого_поколения* отличается только тем, что значение параметра *Счетчик* должно быть равно заданному размеру поколения, это говорит о том, что все особи первого поколения сгенерированы:

Choice from *Общий.Номер_поколения* <= *Количество_поколений* and
Общий.Режим = *Инициализация* and
Общий.Счетчик = *Общий.Размер_поколения*

В конверторе этого правила параметр *Система.Счетчик* обнуляется, и значение параметра *Система.Режим* устанавливается равным *Расшифровка_и_расчет_ФП* (ПГА переходит к блоку расшифровки битовой строки-хромосомы и расчета функции пригодности). Заметим, что здесь рассмотрены не все действия, выполняемые правилами блока генерации первого поколения, а лишь часть, связанная с организацией вычислений.



Рис. 5.7. Структурная схема ПГА в РДО

В общем случае правила, выполняющие основные функции блоков, увеличивают значение счетчика особей и применяются до тех пор, пока значение счетчика не станет равным заданному количеству особей в поколении. Правила, заканчивающие текущий блок, обнуляют параметр *Счетчик* для его корректного использования в следующем блоке и изменяют режим, после чего начинают выполняться условия применимости

правил другого блока. Параметр Система.Режим может принимать пять значений, каждое из которых соответствует блоку ПГА: «Инициализация», «Расшифровка_и_расчет_ФП», «Воспроизведение», «Скрещивание», «Мутация». Структурная схема ПГА с указанием имен правил (образцов) и режимов приведена на рисунке (рис. 5.7).

5.2.5. Реализация блоков ПГА

Генерация первого поколения.

Генерацию первого поколения особей описывают два образца: Генерация_первой_популяции и Конец_генерации_первого_поколения. Первое правило создает особей первого поколения, увеличивая счетчик особей. Когда поколение создано, второе правило устанавливает текущий номер поколения равным 1, переключает режим, обнуляет счетчик битов, который используется в блоке расчета функции пригодности и создает ресурс типа Поколения для первого поколения.

Значения байтовых параметров, представляющих собой битовую строку-хромосому, определяются с помощью генератора псевдослучайных чисел, равномерно распределенных на интервале $[0; 255]$. Для этого используется последовательность Равномерно. описанная в объекте констант функций и последовательностей. Значения остальных параметров создаваемых ресурсов-особей равны нулю, за исключением следующих: номер поколения устанавливается равным 1, номер особи устанавливается равным текущему значению счетчика особей, параметр Статус приобретает значение «Не_рассмотрен» (для первого поколения это не имеет значения, поскольку этот параметр используется при подборе пар для скрещивания, поэтому можно было бы присвоить любое значение из возможных), значение функции пригодности устанавливается равным $-1,0$, это является признаком того, что функция пригодности для данной особи еще не вычислялась. Значение параметра Количество_потомков будет определено при воспроизведении, параметров Значение_ФП, Сумма_до и Сумма_после — в блоке расчета функции пригодности. Параметры Родитель_1, Родитель_2, Скрещивание_байт, Скрещивание_бит, Мутация_байт и Мутация_бит, описывающие обстоятельства появления данной особи, для первого поколения не имеют смысла.

Значения параметров создаваемого ресурса типа Поколения устанавливаются равными нулю, кроме параметров Номер, который приобретает значение 1 (первое поколение), Размер, который становится равным значению параметра Система.Размер_поколения (в общем случае это значение может быть разным для разных поколений, но в рассматриваемом примере оно не изменяется), Минимальное_значение_ФП, которому присваивается «очень большое» значение (десять в двадцатой степени), заведомо большее реального минимального значения функции пригодности для поколения, Максимальное_значение_ФП, которому присваивается «очень маленькое» значение (минус десять в двадцатой степени), заведомо меньшее реального максимального значения функции пригодности для поколения. Эти действия являются подготовительными для вычисления значений

параметров Среднее_значение_ФП, Минимальное_значение_ФП, Максимальное_значение_ФП в блоке расчета функции пригодности. параметра Количество_скрещиваний — в блоке скрещивания, параметра Количество_мутаций — в блоке мутаций.

Расшифровка битов и расчет функции пригодности.

Этот блок описывают два образца-правила Расшифровка_битов_и_расчет_ФП и Образец_Конец_расшифровки. Первое правило вычисляет значение функции пригодности и выполняет ряд других действий для каждой особи поколения, увеличивая счетчик особей. Когда поколение обработано, второе правило переключает режим (следующим блоком является воспроизведение), обнуляет счетчик особей, определяет значение вспомогательного параметра Система.Число, необходимое при воспроизведении для розыгрыша родителя (см. далее), вычисляет среднее значение функции пригодности для поколения, которое является одной из важнейших характеристик поколения и отражает эффективность работы ПГА. Среднее значение функции пригодности вычисляется как сумма значений функции пригодности (накапливается первым правилом), деленная на количество особей в поколении.

Рассмотрим подробнее образец Расшифровка_битов_и_расчет_ФП. Он имеет четыре релевантных ресурса:

- Общий, представляющий в образце ресурс Система;
- Особь, в качестве которого подбирается ресурс типа Особи, относящийся к текущему поколению и имеющий номер, равный значению счетчика особей;
- Поколение, в качестве которого подбирается ресурс типа Поколения, хранящий общую информацию по текущему поколению;
- Особь_для_показа, это ресурс типа Особи_для_показа, имеющий номер, равный значению счетчика особей, т. е. предназначенный для отображения обрабатываемой образцом особи.

Основные функции, выполняемые образцом, следующие:

- вычисление количества бит, стоящих на определенном месте в битовой строке-хромосоме в текущем поколении (эта возможность может быть полезной при детальном анализе эволюции популяции, поэтому она введена в обсуждаемый пример);
- вычисление функции пригодности особи, минимального и максимального значения функции пригодности в данном поколении, накопление суммы значений функции пригодности;
- подготовка данных для взвешенного розыгрыша родителя в блоке воспроизведения;
- расшифровка битовой строки для определения значений X_1 и X_2 , вычисление координат точки, изображающей особь в пространстве поиска.

Для вычисления количества бит, стоящих на определенном месте в битовой строке, используется параметр Система.Счетчик_бит, который обнуляется правилом, завершающим предыдущий блок ПГА. В рассматриваемом образце к значению этого параметра добавляется единица для

тех особей, которые имеют единичное значение бита, стоящего в заданном месте. Для этого используется алгоритмическая функция целого типа `I_lfEQZero`, имеющая три параметра. Эта функция выдает как свое значение второго параметра, если первый параметр равен нулю, и значение третьего параметра — в противном случае.

Для определения значения бита используется алгоритмическая функция `Извлечь`. Эта функция используется для расшифровки битовой строки и имеет три параметра. Первый параметр — это байт, представляющий собой часть битовой строки, из которой необходимо извлечь битовую подстроку, второй параметр — номер бита в байте, с которого начинается подстрока, третий — длина подстроки. Эта подстрока затем рассматривается как целое без знака, которое выдается как значение функции. Так как в языке РДО не определены логические операции над байтами и операции сдвига, для выделения и преобразования подстроки использованы арифметические операции умножения и деления на 2^k . Значение функции вычисляется по формуле:

$$\left(B_i - \frac{B_i}{2^{9-n}} \cdot 2^{9-n} \right) / 2^{9-(n+l)}$$

где B_i — i -й байт строки-хромосомы (первый параметр); n — номер бита, с которого начинается подстрока; l — длина подстроки.

Значение выдает функция `Биты_строки(m)`. Выражение в числителе в скобках производит обнуление битов, стоящих в байте слева от подстроки, поскольку при делении целых чисел в РДО происходит округление результата до целого с отбрасыванием дробной части (желающие могут убедиться в этом сами), деление на знаменатель сдвигает подстроку к правому краю байта.

Вызов функции `I_lfEQZero` для подсчета, например, количества битов, стоящих в третьей позиции первого байта строки-хромосомы, имеет следующий вид:

`I_lfEQZero(Извлечь(Особь.Параметр_1, 3, 1), 0, 1).`

Функцией пригодности является сама оптимизируемая функция f , для вычисления ее значения используется алгоритмическая функция `Вычисление_ФП`, параметрами которой являются расшифрованные значения X_1 и X_2 .

Для вычисления минимального и максимального значений функции пригодности используются функции `RMin` и `RMax`, значениями которых являются соответственно минимальное и максимальное из значений двух передаваемых им параметров.

Розыгрыш родителя в блоке воспроизведения производится на основе значений двух параметров особи-родителя: `Сумма_до` и `Сумма_после`. Параметр `Сумма_до` принимает значение накопленной до обработки данной особи суммы значений функции пригодности (прибавление к этой сумме значения функции пригодности для текущей особи происходит позже, поскольку релевантный ресурс `Поклоление` описан в образце после релевант-

ного ресурса Особь). Параметр Сумма_после вычисляется прибавлением значения функции пригодности данной особи к накопленной ранее сумме.

Воспроизведение

Воспроизведение является первым этапом генерации нового поколения. Этот блок также реализован двумя правилами: Образец_Воспроизведение и Образец_Конец_воспроизведения. Каждое применение первого правила приводит к созданию одного потомка (одной особи следующего поколения). Второе правило завершает этот процесс, переключает режим и создает ресурс типа Поколения для следующего поколения (создание производится так же, как и в блоке генерации первого поколения). Кроме этого, второе правило устанавливает начальное значение параметра Система.Число, которое используется в блоке скрещивания (исходя из особенностей реализации блока скрещивания это значение может быть любым, но большим 1).

Центральной проблемой при воспроизведении является выбор особей для включения в следующее поколение с вероятностью, пропорциональной значению функции пригодности. Это реализовано следующим образом. Значения всех функций пригодности суммируются в блоке вычисления функции пригодности. Последовательностью Розыгрыш_родителя генерируется псевдослучайное число, равномерно распределенное на интервале от нуля до суммы значений функции пригодности для всех особей. Каждой особи на этом интервале соответствует подинтервал, границы которого задаются параметрами Сумма_до и Сумма_после, которые вычисляются таким образом, что подынтервалы для разных особей примыкают друг к другу, но не пересекаются, а длина интервала равна значению функции пригодности данной особи. Потомка получает особь, которой соответствует подынтервал, на который попало псевдослучайное число. Поскольку при равномерном распределении вероятность попадания случайного числа на некоторый интервал пропорциональна длине этого интервала, то вероятность выбора особи для воспроизведения пропорциональна значению функции пригодности этой особи.

Предусловие выбора родителя записывается в образце так:

Choice from Родитель.Номер_поколения=Общий.Номер_поколения *and*
Родитель.Сумма_до $\leq$ Общий.Число *and*
Родитель.Сумма_после $\geq$ Общий.Число

Битовая строка-хромосома переносится в особь-потомок из особи-родителя. Параметры, описывающие происхождение особи-потомка, обновляются за исключением параметра Родитель_1, которому присваивается номер особи-родителя. Такое сочетание значений этих параметров обозначает, что данная особь получена в результате простого воспроизведения из родителя с номером Потомок.Родитель_1 (при выполнении скрещивания и мутации значения этих параметров могут быть изменены).

Кроме этого, в первом образце блока воспроизведения накапливается количество потомков для особи-родителя и генерируется следующее псевдослучайное число для создания следующей особи-потомка (это число будет использовано в предусловии при следующем применении правила,

поскольку конверторы в РДО выполняются после проверки предусловий для всех релевантных ресурсов).

Скрещивание (кроссинговер).

Блок скрещивания имеет ряд особенностей в реализации общей схемы вычислений, связанных с тем, что не каждая пара особей подвергается скрещиванию и количество пар-кандидатов на скрещивание заранее неизвестно. Первая особенность состоит в том, что для реализации этого блока используются три образца-правила, а не два: Образец_Скрещивание, выполняющий собственно скрещивание, т. е. создание двух гибридов из пары отобранных особей в случае, если скрещивание данной пары происходит; Образец_Подбор_пар, подбирающий пары особей и определяющий, произойдет ли скрещивание данной пары, и Образец_Конец_скрещивания, завершающий блок и переключающий режим. Вторая особенность — это то, что параметр Система.Счетчик считает в этом блоке не особи, а пары. Третья особенность заключается в модификации предусловия образца, заканчивающего скрещивание. Оно не содержит проверки значения счетчика, поскольку количество пар-кандидатов на скрещивание заранее не известно. Скрещивание завершается тогда, когда образец Образец_Подбор_пар не смог подобрать очередной пары.

Взаимодействие образцов происходит с использованием параметра Система.Число. При завершении блока воспроизведения ему присваивается значение 2,0. В предусловии образца Образец_Скрещивание значение этого параметра сравнивается с вероятностью скрещивания, хранящейся в параметре Система.Вероятность_скрещивания. Правило применяется лишь в том случае, если значение параметра Система.Число меньше вероятности скрещивания (т. е. если оно попало на интервал от нуля до вероятности скрещивания), а такое значение ему может быть присвоено только в образце Образец_Подбор_пар. Таким образом, сначала подбирается пара особей и параметру Система.Число присваивается псевдослучайное значение, равномерно распределенное на интервале $[0, 1]$. Затем, если предусловие применения образца Образец_Скрещивание выполняется, производится скрещивание данной пары особей и параметру Система.Число присваивается значение 2,0, что предотвращает повторное применение образца. Если же сгенерированное случайное число оказалось больше вероятности скрещивания, то скрещивания данной пары особей не происходит. Такое предусловие в сочетании с тем, что параметру Система.Число присваиваются равномерно распределенные на интервале $[0, 1]$ псевдослучайные значения, обеспечивает заданную вероятность скрещивания.

Процесс подбора пар и возможного скрещивания продолжается до тех пор, пока не будут рассмотрены все пары (значение счетчика особей не станет больше размера поколения, деленного на два, поскольку счетчик считает пары особей) либо окажется невозможным подобрать следующую пару. Это может произойти по следующей причине. Как видно из предусловий образца подбора пар, особи, входящие в пару, должны иметь разных родителей. Иначе скрещивание не имеет смысла, так как особи, воспроизведенные от

одного и того же родителя, имеют одинаковые битовые строки-хромосомы. Если все оставшиеся не рассмотренными как кандидаты на скрещивание особи имеют одного и того же родителя, подобрать пару невозможно. В этом случае согласно механизму языка РДО происходит проверка предусловия следующего образца (попытка выполнения образцов происходит в порядке следования операций в объекте операций), оно выполняется и блок скрещивания завершается. До тех пор, пока происходит подбор пар, образец Образец_Конец_скрещивания не выполняется, поскольку соответствующие образцам Образец_Подбор_пар и Образец_Скрещивание операции записаны в объекте операций перед операцией, соответствующей образцу Образец_Конец_скрещивания.

Подбор пар происходит следующим образом.

Из особей генерируемого поколения, еще не рассмотренных как кандидаты на скрещивание (параметр Особь.Статус равен «не_рассмотрен»), случайно выбирается первая кандидатура. Для реализации случайного выбора использован следующий прием: выбирается особь с минимальным значением приоритетной функции, которая представляет собой псевдослучайное число. Вторая особь выбирается из числа оставшихся аналогично с дополнительным условием, что номер ее родителя не должен совпадать с номером родителя первой кандидатуры. Для особей подобранной пары устанавливается признак, что они были рассмотрены как кандидаты на скрещивание.

В этом же образце разыгрывается, произойдет ли скрещивание, номера особей подобранной пары заносятся в параметры ресурса Система для возможного использования в образце Образец_Скрещивание и определяется потенциальное место скрещивания — номер байта и номер бита, которые также заносятся в параметры ресурса Система. Номер байта вычисляется как равномерно распределенное на интервале от одного до количества байтов в строке-хромосоме псевдослучайное целое число, номер бита — как псевдослучайное целое число, равномерно распределенное на интервале от одного до восьми, если байт не последний в строке-хромосоме и на интервале от одного до количества битов в последнем байте минус 1 для последнего байта. Разрыв битовой строки при скрещивании происходит за битом, номер которого сгенерирован.

Образец Образец_Скрещивание создает две особи-гибрида вместо двух особей-родителей, которых он уничтожает. Пайдя родителей, номера которых равны значениям параметров Система.Родитель_1 и Система.Родитель_2, этот образец побайтно вычисляет значения битовых строк для гибридов с помощью алгоритмической функции Параметр.гибрида:

```
$Function Параметр_гибрида : integer [0..255]
```

```
$Type = algorithmic
```

```
$Parameters
```

```
Номер : integer
```

```
Параметр_1 : integer [0..255]
```

```
Параметр_2 : integer [0..255]
```

\$Body

```

Calculate_if Номер < Система.Скрещивание_байт
    Параметр_гибрида = Параметр_1
Calculate_if Номер > Система.Скрещивание_байт
    Параметр_гибрида = Параметр_2
Calculate_if Номер = Система.Скрещивание_байт and
    Система.Скрещивание_бит = 8
    Параметр_гибрида = Параметр_1
Calculate_if Номер = Система.Скрещивание_байт and
    Система.Скрещивание_бит < 8
    Параметр_гибрида = Извлечь(Параметр_1, 1,
        Система.Скрещивание_бит) *
        Биты_строки(Система.Скрещивание_бит + 1) +
        Извлечь(Параметр_2, Система.Скрещивание_бит +
            1, 8 - Система.Скрещивание_бит)

```

\$End

Функция имеет три параметра: номер байта в строке-хромосоме и два байта, представляющие собой соответствующие байты в битовых строках двух родителей. Результатом функции является байт, содержащий биты второго параметра функции для битов, находящихся левее места скрещивания, и биты третьего параметра для битов, находящихся правее места скрещивания. Значение функции вычисляется следующим образом. Если первый параметр меньше номера байта, в котором происходит скрещивание (байт находится левее), то результатом функции является значение второго параметра, если больше — то значение третьего параметра. Если первый параметр равен номеру байта, в котором происходит скрещивание, а номер бита m , определяющий место скрещивания, равен 8, то скрещивание происходит на границе байтов за данным байтом и значение функции равно значению второго параметра. Наконец, если скрещивание происходит в текущем байте, а m меньше 8, то для формирования результата необходимо взять m первых бит из второго параметра функции и остальные $8 - m$ бит — из второго параметра. Для этого используется уже обсуждавшаяся функция Извлечь.

Параметрам гибридов Родитель_1 и Родитель_2 присваиваются номера особей-родителей, параметрам Скрещивание_байт и Скрещивание_бит присваиваются номер байта и номер бита, определяющие место скрещивания. Значения этих параметров позволяют проследить происхождение особи. Наконец, для статистики накапливается количество скрещиваний в данном поколении (параметр Поколение.Количество_скрещиваний).

Мутация.

Блок мутации завершает генерацию поколения, он реализован двумя образцами-правилами: Образец_мутация и Образец_Конец_мутации. Первый образец последовательно рассматривает все особи, определяя, произойдет ли мутация в данной особи, и генерируя потенциальное место мутации. Для реализации мутаций с заданной вероятностью использована

та же схема, что и при реализации скрещивания для определения факта скрещивания. Мутация в данной особи происходит, если сгенерированное псевдослучайное число, равномерно распределенное на интервале от нуля до единицы, меньше заданной вероятности мутации. Потенциальное место мутации заносится в параметры ресурса Система, оно вычисляется следующим образом. Номер байта вычисляется как равномерно распределенное на интервале от одного до количества байтов в строке-хромосоме псевдослучайное целое число. Номер бита определяется как псевдослучайное целое число, равномерно распределенное на интервале от одного до восьми, если байт не последний в строке-хромосоме и распределенное на интервале от одного до количества битов в последнем байте для последнего байта.

Если мутация имела место, то в параметры особи-мутанта Мутация_байт и Мутация_бит для анализа происхождения особи заносятся значения, определяющие место мутации, и накапливается количество мутаций в данном поколении. Это реализовано с помощью алгоритмической функции Если_мутация, которая сравнивает величину параметра Система.Число с вероятностью мутации и выдает значение первого своего параметра, если мутация имеет место, и значение второго параметра, если нет.

Собственно вычисление битовой строки мутанта производит алгоритмическая функция Параметр_мутанта, текст которой приведен ниже.

```

$Function Параметр_мутанта : integer [0..255]
$Type = algorithmic
$Parameters
    Номер : integer
    Параметр : integer [0..255]
$Body
    Calculate_if Номер <> Система.Мутация_байт
        Параметр_мутанта = Параметр
    Calculate_if Номер = Система.Мутация_байт and
        Извлечь(Параметр, Система.Мутация_бит, 1) = 1
        Параметр_мутанта = Параметр -
            Биты_строки(Система.Мутация_бит+1)
    Calculate_if Номер = Система.Мутация_байт and
        Извлечь(Параметр, Система.Мутация_бит, 1) = 0
        Параметр_мутанта = Параметр+Биты_строки(Система.
            Мутация_бит+1)
$End

```

Функция имеет два параметра: номер рассматриваемого байта из битовой строки-хромосомы и соответствующий байт. Если значение первого параметра не совпадает с номером байта, в котором имеет место мутация, то значение функции равно значению второго параметра. В противном случае значение вычисляется как результат инвертирования соответствующего бита второго параметра функции. Инвертирование производится путем прибавления или вычитания двойки в соответствующей степени.

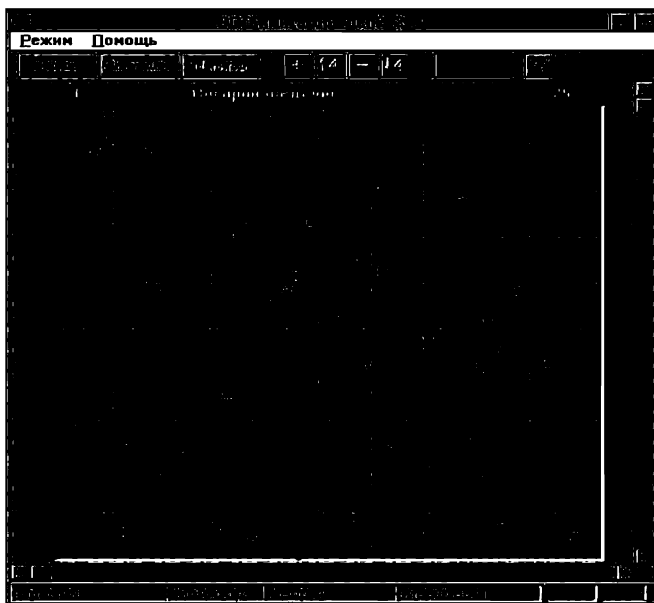


Рис. 5.8. Первое поколение популяции

Расположение особей популяции в пространстве поиска в начале процесса оптимизации и на 20 шаге представлено на рисунках (рис. 5.8–5.12). Напомним, что оптимальное значение функции пригодности находится в верхнем правом углу пространства поиска.

5.3. Система с подстройкой параметров генетических алгоритмов

Для эффективного решения задач система моделирования должна иметь интеллектуальную надстройку, позволяющую заменить специалиста-разработчика. Необходимо решить проблемы, связанные с накоплением и использованием знаний, их пополнением, выводом новых знаний на основе имеющихся в системе, повышением эффективности моделирования. Получение, обобщение и хранение знаний от специалистов-разработчиков может быть проведено с высоким качеством при использовании экспертных систем (ЭС) [136–137].

Сочетание ЭС с имитационной моделью позволяет получить качественно новую ступень в создании инструментальных средств. Если имитационная модель носит описательный характер, то модели, используемые в ЭС, имеют преобразовательный характер, отражая деятельность специалиста при проектировании. Поэтому задачи, решаемые традиционно на имитационной модели с помощью специалиста, могут решаться в ЭС без его участия.

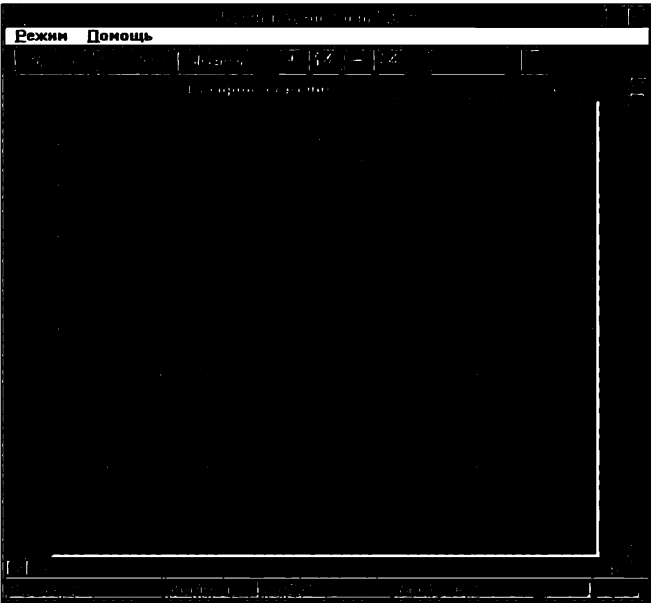


Рис. 5.9. Пятое поколение популяции

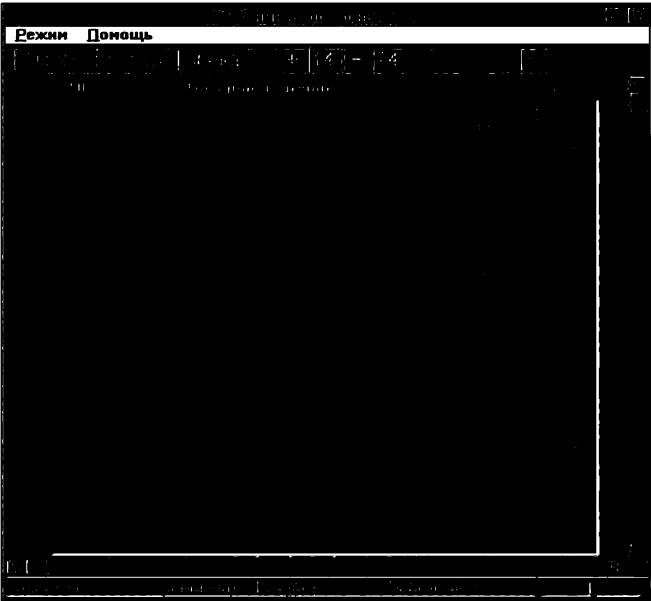


Рис. 5.10. Десятое поколение популяции

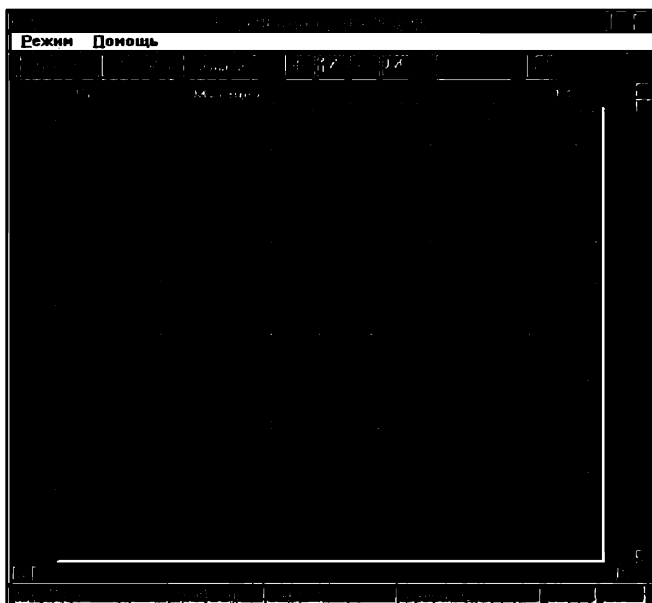


Рис. 5.11. Пятнадцатое поколение популяции

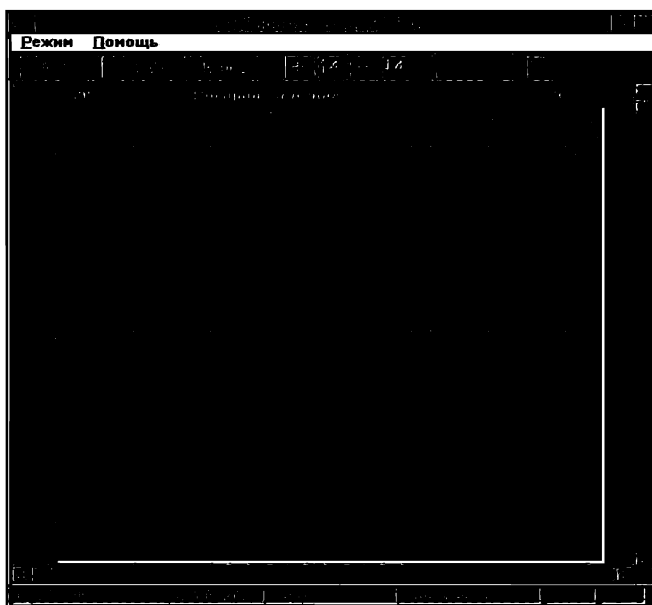


Рис. 5.12. Двадцатое поколение популяции

Возможен ряд вариантов взаимодействия ЭС и системы моделирования. В частности, ЭС может играть роль интеллектуального интерфейса, позволяющего пользователю выходить на имитационные модели и методы оптимизации. При этом гибридная система реализует функции не только интеллектуального интерфейса, но и интеллектуального вычислителя. Состав типовой гибридной системы, включающей в себя указанные составляющие, приведен на рис. 5.13.

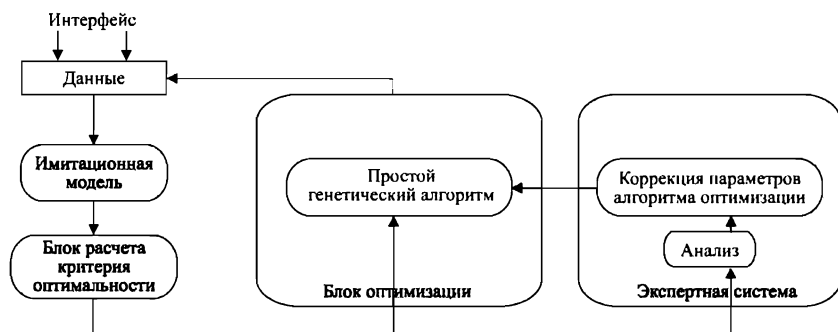


Рис. 5.13. Структура типовой гибридной схема

Рассмотрим функционирование отдельных составляющих такой системы на примере планирования работы некоторого сложного объекта (например, производственного):

- имитационная модель служит для составления плана и использует для этого набор эвристических правил для определения приоритета того или иного заказа включаемого в план работ;
- блок оптимизации обеспечивает подбор приоритетных правил для составления планов работы с наилучшими показателями. В теории расписаний рассматривается большое количество приоритетных правил, поэтому необходим выбор лучших правил для текущей ситуации, а также выбор оптимальных значений их параметров;
- ЭС предназначена для изменения параметров поиска на основе некоторых представлений человека-оператора о перспективности той или иной стратегии поиска.

Рассмотрим работу такой гибридной системы на примере системы планирования работы цеха [135–139].

Гибридная система предназначена для планирования цеха, состоящего из станков различных видов, на которых обрабатываются детали нескольких типов. Маршрут обработки детали представлен графом технологических операций и может содержать альтернативные варианты обработки (рис. 5.14). Для каждой операции известны: номер станка, время выполнения обработки, время наладки станка. Все детали связаны с некоторыми заказами. Заказ описывается следующими параметрами: серийным номером, числом деталей составляющих заказ, временем начала обработки и сроком выполнения.

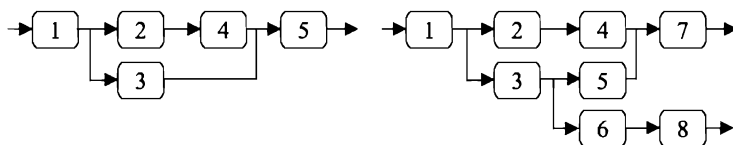


Рис. 5.14. Примеры маршрутов деталей

Все элементы участка представлены в имитационной модели в терминах среды РДО как ресурсы.

Для планирования в относительно простой имитационной модели были описаны следующие операции (они описаны в РДО как модифицированные производственные правила):

- обработка партии на станке; время обработки рассчитывается как число деталей в партии, умноженное на время выполнения операции;
- транспортирование партии между двумя станками или между центральным складом и станком, время транспортирования определяется на основе известной матрицы расстояний;
- наладки станков;
- изменения маршрута обработки партии, когда время превышает минимальное время начала обработки заказа.

Ситуации, в которых необходимо принимать некоторое решение, называются точками решения. В модели их две:

- после завершения некоторой технологической операции, когда необходимо назначить станок для следующей операции. Если существует только одна возможность, это сделать легко; но иногда имеются несколько возможностей. В этом случае мы выбираем для операции станок с наименьшим числом намеченных к обработке партий деталей. Этот параметр станка рассчитывается динамически в ходе моделирования;
- когда станок закончил процесс обработки партии, мы должны решить, какая партия будет следующей обрабатываться на этом станке. В этом случае для каждой партии, намеченной для обработки на этот станок, но еще не обработанной, рассчитывается значение приоритета, и затем партия с минимальным приоритетом отбирается для обработки;

Цель блока оптимизации гибридной системы — улучшение решения посредством выбора значений управляемых переменных. Для этих целей используется ПГА, описанный выше.

Цель экспертной системы в составе гибридной системы — улучшение показателей ПГА, прежде всего, повышение сходимости процесса оптимизации посредством включения в процесс некоторых представлений (знаний) человека-оператора о перспективности той или иной стратегии поиска. В этом случае ЭС выполняет функцию «селекционера» целенаправленно изменяя параметры ПГА для сокращения времени вычисления.

ЭС осуществляет направленный выбор таких параметров ПГА, как: размер популяции, вероятности скрещивания и мутации. Кроме того, она применяет некоторые правила для сохранения особей с высоким значением

функции пригодности из поколения в поколение в ходе их воспроизводства, регулируя число особей с одинаковым значением функции пригодности и т. п.

ЭС таким образом представляет область комбинации знаний о генетических алгоритмах, вычислительной математике, искусственном интеллекте и знаний эксперта. Прикладная область для ЭС хорошо не определена, и пространство поиска плохо структурировано; именно поэтому ЭС работает наряду с ПГА на основе текущих данных относительно популяции и текущего состояния имитатора.

В отличие от имитатора, в ПГА и ЭС нет потребности учитывать временной аспект. Поэтому образцы действий в РДО для ПГА и ЭС представляют собой обычные продукционные правила.

С помощью ПГА оптимизировалось расписание операций для цеха, состоящего из семи станков; портфель заказов содержал 10 позиций; общее количество операций, выполняемых над деталями, составляло 64 (от 3 до 13 операций на деталь). Управляемые переменные при оптимизации были приоритеты заказов. Приоритет заказа был задан целым числом в диапазоне от 1 до 8, увеличение значения приоритета ускоряет обработку заказа. Приоритет кодируется тремя битами, длина битовой строки-хромосомы равнялась 30 битам (10 раз по 3). Функция оценки качества получаемого расписания S представляла собой сумму штрафов для каждого заказа. Значение штрафа вычислялось следующим образом. Если заказ опоздал к плановому сроку, то штраф равнялся значению величины опоздания в квадрате, если изготовление заказа завершалось раньше планового срока, но не более, чем на 4 часа, то штраф равнялся нулю. Если же опережение было больше четырех часов, то штраф равнялся значению величины опережения минус 4 часа.

Задача оптимизации формулировалась следующим образом: найти сочетание приоритетов заказов, минимизирующее значение штрафной функции.

Функция пригодности особей вычислялась по следующей формуле:

$$f(\omega_i) = \frac{1}{1 + S/20},$$

где S — функция оценки качества расписания.

Поскольку функция оценки качества расписания неотрицательна, то функция пригодности меняется от 1 при $S = 0$ (наилучшее расписание) до 0 при S , стремящимся к бесконечности.

Предварительные результаты минимизации этой функции с помощью ПГА с различным значением размера популяции N показывают возможность улучшить качество расписания. Так, в различных прогонах модели с $N = 50 \dots 400$, среднее значение функции пригодности увеличено с приблизительно 0,18 (первое поколение) до $0,64 \div 0,68$ в десятом двадцатом поколениях, что соответствует значениям S -функции, уменьшающейся от 90 до менее чем 10. Некоторые результаты минимизации функции пригодности с помощью ПГА при различных значениях его параметров показаны на графиках (рис. 5.15). На графиках приведены средние значения функции

пригодности по популяции (отмеченная кривая) и максимальные значения функции пригодности (пунктирная кривая). Изменение по оси X представляет последовательность поколений.

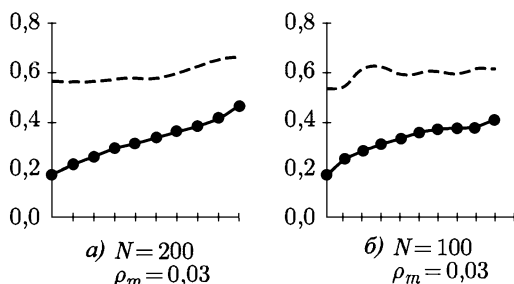


Рис. 5.15. Результаты оптимизации расписания

Некоторые исследовательские прогоны модели, когда процесс поиска управляется ЭС (рис. 5.16), показывают возможность увеличить эффективность поиска, даже применяя довольно простые правила в ЭС. Примеры правил:

- Если ($f(\omega_i)$ лучшей особи в текущем поколении меньше чем $f(\omega_i)$ лучшей особи в предыдущем поколении) То (заменить самую плохую особь в текущем поколении на лучшую особь в предыдущем поколении);
- Если $((f_{\max} - f_{\min})/f_{\text{mean}} < 1)$ То ($P(\text{OK}) = 0,9$, $P(\text{OM}) = 0,7$);
- Если $((f_{\max} - f_{\min})/f_{\text{mean}} > 1)$ То ($P(\text{OK}) = 0,6$, $P(\text{OM}) = 0,1$),

где $f_{\max}$, $f_{\min}$, f_{mean} — максимальное, минимальное и среднее значения функции пригодности последнего поколения.

Как видно из результатов исследования (рис. 5.16), лучшее значение функции пригодности без ЭС — 0,511, в то время как с использованием ЭС — 0,648 (для $N = 50$).

Из полученных результатов видно, что гибридные системы обеспечивают моделирование и управление расписанием работ, объединяя преимущества имитационного моделирования, ЭС и ПГА. Существование развитых инструментальных средств создания гибридных систем типа интеллектуальной среды имитационного моделирования РДО обеспечивает проектирование гибридных систем.

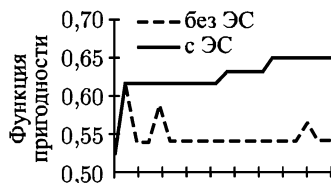


Рис. 5.16. Результаты оптимизации расписания с ЭС

5.4. Эволюция популяции автоматов

Развитие искусственного интеллекта характеризуется интеграцией различных подходов, концепций и направлений, что связано с пересмотром места ряда базовых понятий и возвратом в контексте происходящей интеграции к ряду старых идей и моделей. Так, к числу таких идей и методов

относятся эволюционное моделирование, начало которому было фактически положено теорией самовоспроизводящихся автоматов Дж. фон Неймана [140], а также два крупных направления, зародившиеся и активно развивавшиеся еще в 60-х годах в Советском Союзе — теория коллективного поведения автоматов [47, 141, 142], которая актуализируется в связи с развитием работ по распределенному искусственному интеллекту и много-агентным системам [143, 144], и семиотическое моделирование, связанное с организацией ситуационного управления в сложных системах [145] и развитием интегрированных интеллектуальных систем. Подобные идеи, модели и подходы, объединяясь с неклассическими математическими подходами, приводят к формированию новых интегральных направлений, как в самом искусственном интеллекте, так и в пограничных областях. Примерами здесь служат «Мягкие вычисления» (Soft Computing), «Машинный или вычислительный интеллект» (Computational Intelligence) и «Искусственная жизнь» (Artificial Life) [146–147].

Естественным фундаментом работ по искусственной жизни служат биологические дисциплины, позволяющие определить основной круг понятий, рассматриваемых в искусственной жизни: самовоспроизведение, эволюция, естественный и искусственный отбор, генетическая связь, морфогенез, онтогенез и филогенез, генотип и фенотип, хромосома, организм, популяция и пр.

В России у истоков исследований научного направления «искусственная жизнь» стояли Н. Амосов [148], В. Глушков [149], А. Колмогоров [150], А. Ляпунов [151], Д. Поспелов [152–154], М. Цетлин [141] и др. Они считали, что сущность жизни определяется не столько свойствами материального субстрата жизни (белковых соединений или структур ДНК), сколько организацией элементов и процессов в целостную систему. Если искусственно созданная организация в существенных чертах эквивалентна организации живого, а функции на выходе этой системы и обычной биологической структуры одинаковы, то такую систему (модель) можно назвать живой. Комбинация большого числа дискретных элементов создает и новые качества.

Одним из наиболее важных математических методов, имеющих отношение к искусственной жизни, является теория автоматов, в первую очередь, теория коллективного поведения автоматов и теория клеточных автоматов.

В основе моделей коллективного поведения, которые рассматривались в 60-х годах М. Цетлиным и его последователями [141], лежало допущение о полном априорном незнании автоматом свойств той среды, в которой он действует. Автомат, в частности, не имел априорной информации о наличии кроме него самого каких-либо других автоматов. А их действия воспринимались им как свойства самой среды (рис. 5.17). Автомат может воспринимать и интерпретировать сигналы, поступающие из среды в определенные моменты времени, и может выдавать в среду формируемые им сигналы. То есть, автомат обладает свойством реактивности. Он способен оценивать сигналы среды как полезные (поощрение за правильно выданный в среду сигнал) или вредные (штраф за неправильно выданный сигнал) с точки зрения достижения своей внутренней цели.

М. Цетлина интересовал вопрос о возможности моделирования устройством (автоматом) минимальной сложности целесообразного поведения в подобной среде. Термин «целесообразное поведение» понимался при этом следующим образом. Пусть на месте автомата A находится датчик равновероятных сигналов, значения которого берутся из некоторого множества M . Когда на вход этого датчика подается очередной сигнал о состоянии среды, датчик реагирует на него выдачей одного сигнала из M . Датчик, таким образом, не использует никак информацию из среды. Если за достаточно большой отрезок «своей жизни» датчик накапливал суммарное значение положительных откликов из среды, равное W^* , а некоторый автомат за тот же период накапливал W , то поведение агента естественно считать целесообразным, если $W > W^*$, и нецелесообразным в противном случае.

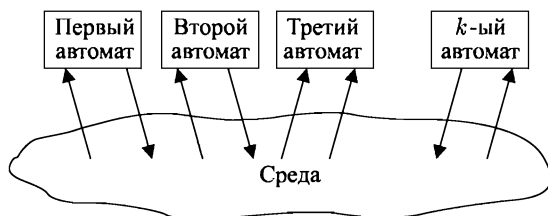


Рис. 5.17. Взаимодействие автоматов со средой

Как показал М. Цетлин, если среда является вероятностной и стационарной (за действия автомата реакция среды выдается с неизменным для каждого действия распределением), то конечный автомат определенной простой структуры достаточно целесообразно взаимодействует со средой.

В [152] эта модель обсуждается с точки зрения поведения живых существ. Она полностью соответствует процедурам адаптации к неизвестным условиям среды, изучаемым в рамках стимульно-реактивной модели поведения животных и человека. Однако относительно простые автоматы оказались малопригодными в средах, где стационарность отсутствует.

Нас же, с точки зрения эволюционного моделирования, интересует поведение популяции автоматов, обладающих свойством воспроизведения себе подобных. Способность к репликации рассматривается многими исследователями как наиболее характерная особенность жизни [155]. Понимание механизма возникновения репликации в ходе эволюции отождествляется с собственно решением проблемы происхождения жизни. Внимание к явлениям репликации в контексте проблемы происхождения жизни оправдано. Оно действительно играет ключевую роль. Но не потому, что в репликации состоит феномен жизни, а потому, что репликация — необходимый элемент трансформации акта упорядочения на микроскопическом уровне в макроскопическое событие. Репликация реализует итеративный характер биологического упорядочения.

Итак, исследуем вопрос способности коллектива автоматов к саморегулированию по численности с целью оптимизации численности коллектива таким образом, чтобы повысить «уровень жизни» каждого индивида по-

средством стремления самого коллектива к рациональному числу особей в популяции по отношению к ресурсам внешней среды.

Концептуальная схема, которая лежит в основе моделирования, следующая:

– популяция из K автоматов существует в некоторой среде. Каждый из них взаимодействует со средой самостоятельно, не зная не только о действиях других членов популяции, но и об их существовании. Для каждого автомата остальные участники коллектива как бы растворяются в среде, выступают по отношению к данному автомату как часть среды;

– с другой стороны, популяция выступает как одно целое. Она интерпретируется как многоклеточный организм, который, состоя из большого числа клеток, молекул, казалось бы независимых друг от друга, в то же время выступает как единое целое, объединенное общими целями и задачами. В данном случае уместно употреблять выражение «интеллект роя». В основе поведения популяции лежит гипотеза простоты, высказанная М. Цетлиным. Суть ее сводится к тому, что любое достаточно сложное поведение складывается из совокупности простых поведенческих актов. Их совместная реализация и простейшее взаимодействие приводит в результате к весьма сложным поведенческим процессам. Клетки человеческого тела, пчелы улья или муравьи муравейника могут выступить в качестве иллюстрирующего примера.

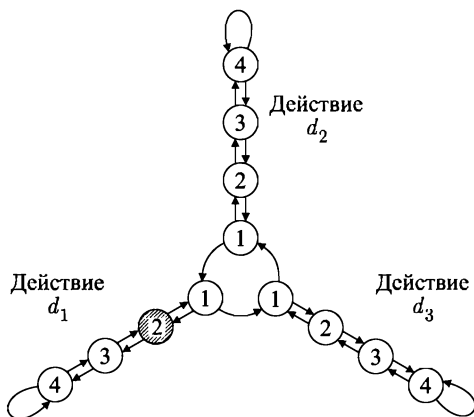


Рис. 5.18. Схема автомата с линейной тактикой

Первый рассматриваемый тип автомата — это автомат с линейной тактикой; пример такого автомата, способного выдавать в среду три различных сигнала, приведен на рисунке (рис. 5.18) [47]. Число лепестков равно числу действий, доступных автомату. В каждом лепестке выделено четыре устойчивых состояния, в которых может находиться автомат. В любом из состояний, образующих лепесток ромашки, устойчиво выдает в среду сигнал действия, приписанного данному лепестку. Смена состояний происходит с учетом сигналов оценок за действия, поступающих от внешней среды. Это — двоничные сигналы: (штраф, приз). Сплошные стрелки показывают,

как происходит изменение состояния автомата при получении им приза, а пунктирные — при получении штрафного сигнала от среды.

В приведенном на рисунке автомате в каждом лепестке — по четыре состояния. Каждый лепесток может содержать не четыре, а любое другое число состояний. Обозначим это число через q . Оно называется глубиной памяти автомата. Выбор этого числа состояний произволен. Смысл этого параметра заключается в следующем. Чем больше q , тем более инерционен автомат, ибо тем большая последовательность штрафов вынуждает его к смене действий.

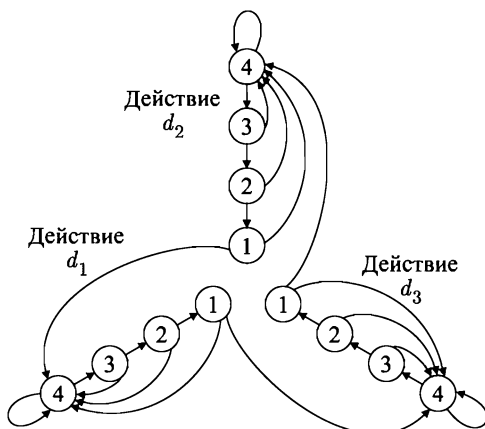


Рис. 5.19. Схема автомата Роббинсона

На рисунке 5.19 приведена схема автомата, предложенного Г. Роббинсом. Он похож на автомат с линейной тактикой, и действует при получении штрафа аналогично автомату с линейной тактикой. Но при получении штрафа его поведение резко отличается от поведения автомата с линейной тактикой. Такой автомат можно назвать «доверчивым» [47].

Эволюция популяции автоматов в стационарной среде не представляет большого интереса, так как в этом случае скорее всего должен существовать стационарный режим, в который и должна прийти популяция. Поэтому рассмотрим нестационарную среду и проведем имитационное моделирование популяции автоматов в этой среде.

Законы изменения параметров внешней среды могут быть различными. Далее для моделирования динамической среды с помощью единственной последовательности, генерирующей случайные числа с равномерным распределением, выбиралась реакция среды в зависимости от того выпавшее число больше или меньше границы, определяющей получение реакции «Приз» или «Штраф». Изменение положения границы определяет изменение вероятностей получения реакции «Приз» или «Штраф». а, следовательно, и динамичность среды.

Отличительной особенностью внешней среды является то, что по сути не требуется ее динамическое изменение в течении прогона имитационной

модели. Стоит задача анализа способности популяции автоматов к саморегулированию (достижение некоторого оптимума) по численности в данных конкретных условиях внешней среды. То есть, мы устанавливаем параметры среды и затем отслеживаем поведение популяции автоматов в данной среде с целью определения их поведения и характера приближения популяции к оптимальной численности. Затем, меняя характеристики среды проводим те же наблюдения, но в среде с другими параметрами. Изменение характера среды, в основном, сводится к изменению вероятностей получения автоматами реакции «Штраф» и «Приз». Именно это определяет среду как Дружелюбную, Нейтральную или Враждебную. Вид среды влияет на поведение автоматов, на их способность к размножению и, в конечном итоге, на численность популяции.

Дружелюбная среда характеризуется тем, что вероятность получения реакции «Приз» в популяции возрастает для всех автоматов одинаково в зависимости от количества призов, полученных ими на предыдущем шаге. Дифференцированная дружелюбная среда отличается тем, что взаимодействует с автоматами индивидуально. То есть, изменение вероятностей получения призов и штрафов происходит индивидуально на каждой ветке каждого автомата в отдельности независимо от поведения других членов популяции (рис. 5.20, сплошная линия).

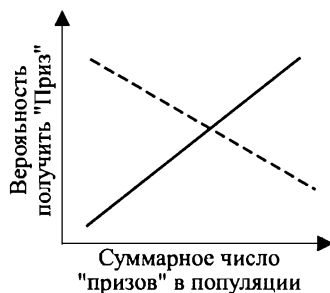


Рис. 5.20. Дружелюбная и враждебная среды

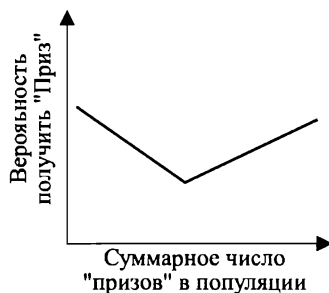


Рис. 5.21. Меняющаяся среда

Враждебная среда — это ситуация, при которой вероятность получения реакции «Приз» в популяции убывает дискретно для всех автоматов одинаково в зависимости от количества призов, полученных автоматами на предыдущем шаге. Дифференцированная враждебная среда аналогична дифференцированной дружелюбной среде за исключением того, что вероятность получения призов уменьшается (рис. 5.20 пунктирная линия).

Меняющаяся среда, в которой вероятность получения призов убывает до некоторого числа накопленных в системе призов, а затем начинает расти (рис. 5.21). Дифференцированная меняющаяся среда, в которой вероятность получения призов меняется от уменьшения к увеличению, отличается также тем, что это изменение распределяется на каждый автомат по отдельности (т. е. накопление призов до определенного значения происходит внутри каждого автомата независимо от других).

Интерес представляет, прежде всего, способность сообщества регулировать свою численность в зависимости от ресурсов внешней среды. Оптимальность численности коллектива характеризуется некоторой величиной, так называемым «уровнем жизни». Он определяется как отношение общего количества «Призов», полученного коллективом автоматов на данном шаге моделирования к общему количеству автоматов (особей) в сообществе на данном шаге моделирования. Суть заключается в том, чтобы проследить способность сообщества автоматов к самостоятельному саморегулированию по численности. Таким образом, оптимальность данного коллектива (его жизнеспособность) определяется «уровнем жизни» коллектива, который складывается из «уровней жизни» отдельных его членов. Чем больше «Призов» получает сообщество автоматов, тем казалось бы выше его жизнеспособность и оптимальность. Но при этом вовсе не значит, что «уровень жизни» отдельных членов коллектива достигает своего наиболее высокого значения. Таким образом, за характеристику оптимальности коллектива принимается «уровень жизни» организации, которая может принимать более совершенное устройство в том случае, когда «уровень жизни» каждого отдельного члена сообщества (доля от общего количества «Призов», полученного коллективом) выше.

Розыгрыш реакции среды в имитационной модели производится следующим образом:

- в программе используется последовательность, вычисляющая значения на основе генератора псевдослучайных чисел, как равномерно распределенные на заданном диапазоне вещественные числа;
- после того, как сгенерировано любое число на интервале от 0 до 10 осуществляется преобразование его в конкретную реакцию «Приз» или «Штраф». Для этого полученное число сравнивается с границей, которая разграничивает те области на интервале, которые относятся к реакциям «Приз» или «Штраф». То есть, если сгенерированное число больше границы, то реакции присваивается значение «Приз», а если меньше — то значение «Штраф». Изменение вероятности (т.е. случай динамической среды) осуществляется путем передвижения границ. В случае дифференцированных сред изменение границы происходит независимо для каждой ветки каждого автомата в зависимости от реакции, полученной автоматом на предыдущем шаге. В случае дифференцированной враждебной среды граница передвигается таким образом, чтобы увеличивать вероятность получения реакции «Штраф» в случае, если автоматом на предыдущем шаге была получена реакция «Приз». В случае дифференцированной дружелюбной среды граница передвигается таким образом, чтобы увеличивать вероятность получения реакции «Приз» в случае, если автомат на предыдущем шаге получил «Приз». В случае дифференцированной среды, изменяющей свое отношение с враждебного на дружелюбное, граница передвигается таким образом, чтобы увеличивать вероятность получения реакции «Штраф» до момента, когда автомат накопит определенное число «Призов», а затем увеличивает вероятность получения реакции «Приз». Это происходит для каждого автомата в отдельности, независимо от поведения и состояния дру-

гих автоматов. В случае недифференцированной среды, изменяющей свое отношение с враждебного на дружелюбное, граница передвигается таким образом чтобы увеличивать вероятность получения реакции «Штраф» до момента, когда в системе накопится определенное число «Призов» (как сумма «Призов» полученных всеми автоматами), а затем увеличивает вероятность получения реакции «Приз». Граница меняется для всех автоматов одинаково.

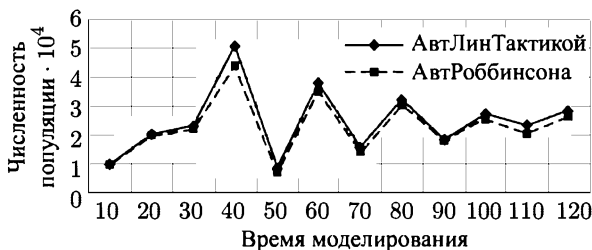


Рис. 5.22. Изменение численности автоматов во времени

Результаты имитационного моделирования в дифференцированной среде, свойства которой менялись в зависимости от числа особей в популяции, приведены на рисунках 5.22 и 5.23. В начале имитации в популяции было по 10 автоматов каждого типа. Из графиков видно, что в системе имеется некоторый установившийся режим, к которому стремится численность популяции. При этом автоматы Роббинсона в среднем быстрее находят этот режим.

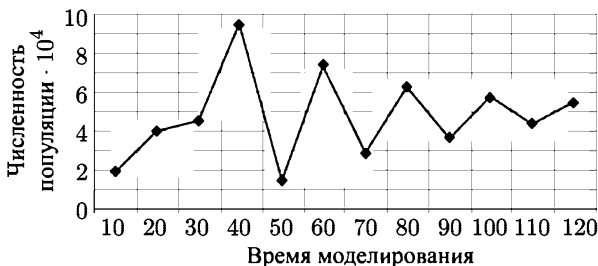


Рис. 5.23. Изменение общей численности автоматов в популяции

Таким образом, популяция сравнительно простых автоматов, способных к воспроизведению себе подобных, демонстрирует способность адаптироваться в изменяющейся среде.

5.5. Многоагентные системы

5.5.1. Взаимодействие агентов в МАС

Разработка технологии искусственных агентов, создание многоагентных систем (МАС) и виртуальных организаций представляет собой одну из наиболее важных и многообещающих областей развития новых информационных и коммуникационных технологий, где широко используются ме-

тоды искусственного интеллекта. В настоящее время существует большое число определений агентов, их классификаций, многоагентных систем. Так, агентов можно классифицировать по следующим двум признакам: *а)* степень развития внутреннего представления внешнего мира и *б)* способ поведения. Следуя этой классификации агентов, рассмотренные в предыдущем разделе автоматы могут быть отнесены к реактивным агентам [143, 144]. Вторым классом агентов, выделенных по первому из названных признаков, являются интеллектуальные (когнитивные, рассудочные) агенты. Существенное различие между реактивными агентами (популяция которых представляет «интеллект роя») и когнитивными связано с возможностью последними прогнозирования изменений внешней среды и, как следствие, предвидеть результаты своей деятельности, анализировать ее результативность и планировать свое дальнейшее поведение. Реактивные агенты практически не способны планировать свою деятельность, хотя, как мы видели, обладают способностью адаптироваться в изменяющейся среде.

Когнитивные агенты гораздо более автономны, чем реактивные, их, как правило, в популяции немного и они при этом демонстрируют целесообразное поведение в популяции аналогичных агентов, в значительной степени независимых друг от друга. У специалистов сформировалось и используется представление об интеллектуальных агентах как активных, автономных, коммуникабельных, а главное, мотивированных объектах, «живущих» и «действующих» в сложных, динамических и, чаще всего, виртуальных средах.

Решение некоторой задачи одним агентом на основе его знаний представляет собой централизованную систему искусственного интеллекта, в которой агент имеет полную информацию о решаемой проблеме и имеет все необходимые ресурсы, знания и данные для ее решения. Однако для реальных условий такое положение вещей встречается весьма редко. Как показано в [47], система, возникшая в процессе эволюции, не может управляться централизованной системой управления. Поэтому для решения действительно сложной проблемы требуется взаимодействие некоторого множества агентов, объединяющих для достижения общей цели свои ресурсы и знания. Это множество агентов и представляет собой МАС. Таким образом, организации кооперации агентов является одной из центральных и наиболее сложных проблем построения МАС.

В МАС предполагается, что отдельный агент может иметь лишь частичную (локальную) информацию об общей задаче и соответственно общей цели и способен выполнить лишь некоторую подзадачу. Гетерогенность и высокая неопределенность подсистем сложной системы приводит к тому, что в МАС должны взаимодействовать агенты различных типов, отличающихся различными наборами знаний и ресурсов, используемыми ими для решения соответствующих задач. При этом агенты могут разрабатывать планы действий, исходя из анализа планов и намерений других агентов в МАС.

В настоящее время используются различные модели координации поведения агентов в МАС. Среди них исследователи рассматривают модели

контрактных сетей, теоретико-игровые модели, модели планирования коллективного поведения, использование Belief–Desire–Intention (BDI)–архитектур, модели координации поведения на основе конкуренции и другие.

Взаимодействие агентов — это первое, что необходимо рассматривать, говоря о создании MAC. Взаимодействие означает установление двусторонних и многосторонних динамических отношений между агентами. Оно является также предпосылкой модификации как самих агентов, так и отношений между ними. Взаимодействие между агентами имеет направленность — оно может быть положительным и отрицательным, т. е. носить характер содействия или противодействия, кооперации или конкуренции, сотрудничества или конфликта. Содействие агентов друг другу означает их взаимопомощь в интересах повышения эффективности всей MAC. Содействие перерастает в кооперацию при наличии общей цели, взаимной адаптации и широком использовании возможностей друг друга.

Рассмотрим использование для координации поведения интеллектуальных агентов в MAC генетических оптимизационных алгоритмов. Этот подход продемонстрируем на решении типовой организационной задачи — планирования перевозок грузов.

5.5.2. Транспортная задача и структура системы

Одной из типовых задач для производственных систем является управление автоматизированной транспортной системой. Основная решаемая проблема — это определение оптимального порядка обслуживания заявок автоматическими транспортными устройствами при заданном критерии. Заявки случайным образом возникают в различных местах производственной системы. Далее рассматривается децентрализованная система управления сложной дискретной системой, осуществляющей транспортировку грузов различного типа. Решается задача управления: минимизировать суммарный путь, пройденный в процессе развозки грузов всеми имеющимися транспортными устройствами.

Система создается как интеллектуальная многоагентная. Каждое транспортное устройство представляет собой некоторый мобильный агент, способный принимать решения. Цель каждого агента — обеспечить свое оптимальное функционирование в среде других агентов. Агенты рассматриваются как интеллектуальные единицы MAC, способные находить свое наилучшее (рациональное) поведение на основе имеющейся у них информации о состоянии системы и знаний. В том случае, когда агенты оптимизируют свои цели по отдельности, могут возникнуть конфликты, когда либо планы по перевозке грузов нескольких агентов пересекаются, либо когда некоторые из грузов оказываются «вне поля зрения» агентов. Так, если все агенты одинаковые и у них одинаковые начальные условия для принятия решений, то может случиться, что все они примут абсолютно одинаковые планы по перевозке грузов — глобальный конфликт в системе. Организация коллективного поведения множества агентов призвана скоординировать индивидуальные действия агентов (может с ухудшением эффективности каждого из них) для достижения общесистемной цели. В качестве механизма ко-

ординации поведения агентов предлагается использовать генетический оптимизационный алгоритм как вероятностный механизм самоорганизации популяции агентов.

Система представляет собой склад, который обслуживается некоторым количеством автоматических транспортных устройств. Абстрагируясь от конкретных условий, систему можно представить как коллектив автономных мобильных агентов, взаимодействующих друг с другом опосредованно через окружающую среду. Каждый агент может одновременно перевозить ограниченное количество грузов, определяемое его грузоподъемностью и габаритами грузов. Имеется портфель заказов, содержащий текущий список грузов, которые требуется перевезти эффективным с точки зрения всей системы образом. В момент принятия решения портфель заказов можно считать заданным и постоянным. При поступлении новых заказов в портфель задача решается заново в новых условиях (часть грузов перевезена, появились новые грузы, мобильные агенты находятся в других точках системы). В качестве критерия эффективности рассматривается суммарный путь, который прошли все мобильные агенты, перевоза заказы.

Мобильный агент может перемещать одновременно несколько грузов, количество которых известно. Поэтому, как только определены все грузы для перевозки данным агентом, то перед ним встает задача выбора последовательности объезда стеллажей, при которой пройденный путь был бы минимальным. Таким образом, мобильный агент ставит себе целью перемещение возможно большего числа попутных грузов при минимизации транспортного пути. Данная цель служит мотивацией его поведения. Для достижения указанной цели мобильный агент должен выбрать «выгодные» для него грузы и решить оптимизационную задачу по их развозке.

Таким образом, каждый агент в системе решает оптимизационную задачу, определяя выгодные для себя условия функционирования. Решив оптимизационную задачу для себя, мобильный агент может вступить в конфликт из-за грузов с другими агентами, которые также запланировали перемещение тех же самых грузов. Поэтому для достижения стоящей перед всей системой (общей для всех агентов) цели агенты должны образовать некоторую коалицию и, может ухудшив свои персональные показатели, обеспечить общесистемный выигрыш [156].

5.5.3. Функционирование мобильного агента

Задача оптимальной развозки решается мобильным агентом в условиях, когда грузы для перевозки ему полностью определены внешней средой (в нашем случае — ГА). Ему необходимо решить, в какой последовательности необходимо увозить и привозить грузы, чтобы пройденный путь был минимальным. Решение находится с использованием механизма точек принятия решений, реализующего алгоритм поиска на графе состояний [37, 134]. Вершины графа — это состояния системы, а дуги — продукционные правила. ЦФ мобильного агента представляет собой суммарный путь всех мобильных агентов, который они проходят, перевоза грузы.

В РДО механизм точек решений работает следующим образом: при имитации после наступления очередного события происходит останов моделичного времени и осуществляется решение о продолжении имитации. При этом осуществляется поиск на графе (в глубину, в ширину или информированный) заданном неявным образом.

Точки принятия решений в РДО описывают способы использования образцов для моделирования процесса и принятия решений на уровне событий. Объект точек принятия решений имеет следующий формат:

```
<описание_точки_принятия_решений> | <блок_активностей>
{ <описание_точки_принятия_решений> | <блок_активностей> }
```

Описание каждой точки принятия решений имеет следующий формат:

```
«$Decision_point» <имя_точки> «:» <тип_точки> [<признак_трассировки>]
«$Condition» <условие_активизации_точки>
[«$Term_condition» <терминальное_условие>
«$Evaluate_by» <оценка_стоимости_оставшегося_пути_на_графе>
«$Compare_tops» «=» ( «YES» | «NO» ) ]
<блок_активностей>
```

В РДО имеются два типа точек принятия решений:

Some — просмотреть все активности данной точки, проверить предусловия, выполнить ту активность, предусловия которой удовлетворяются;

Search — реализовать поиск на графе состояний.

В случае планирования маршрута агента алгоритм поиска в ширину работает следующим образом: выбирается начальная вершина — пункт поступления грузов, т. е. начальная отправная точка движения мобильного агента. Затем раскрываются все вершины, инцидентные начальной, и подсчитывается стоимость пути от начальной вершины до каждой из этих вершин (рис. 5.24).

При интерпретации графа поиска примем, что вершины, связанные с начальной вершиной, — это стеллажи, в которые необходимо доставить грузы, или пункт приема грузов, куда надо доставить грузы со стеллажей. В результате определяется груз, который нужно отвезти в первую очередь. Затем раскрываются все вершины, инцидентные этой вершине, и определяются стоимости путей до остальных нераскрытых вершин, и так до тех пор, пока не будут раскрыты все вершины. В результате такого поиска находим оптимальную последовательность грузов, при этом грузам присваивается приоритет развоза по стеллажам и в пункт приема. Когда всем грузам, которые необходимо перевезти, присвоены приоритеты, начинается собственно развозка грузов и строится новый граф, где в качестве стоимости пути выступает приоритет груза.

РДО-имитатор позволяет использовать не одну точку принятия решения, а несколько, если это необходимо. В результате в модели строятся два и более графа состояний, т. е. реализуются две и более системы управления, отдельно для каждого мобильного агента. Возможный вид графа поиска агентом приведен на рисунке (рис. 5.25).

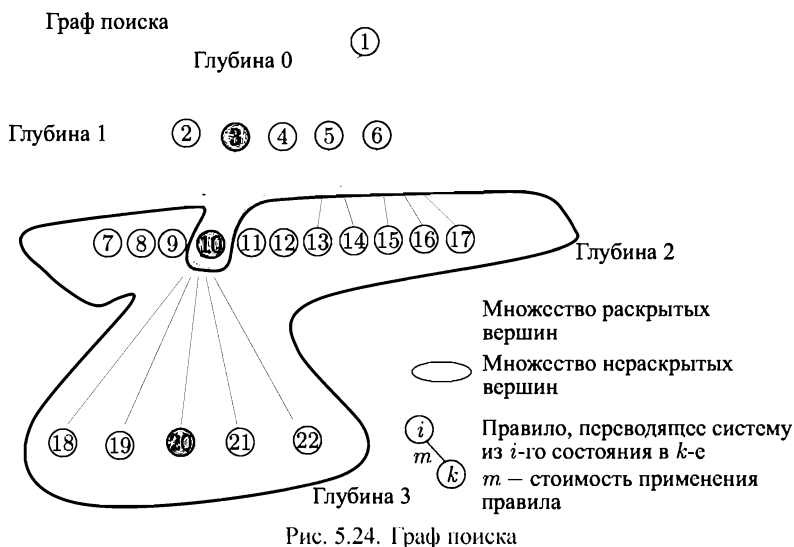


Рис. 5.24. Граф поиска

При поиске новые вершины (или состояния) графа раскрываются от той вершины, в которой ЦФ минимальна независимо от того, сколько в данном состоянии перевезено грузов. Даже придя к такому состоянию, когда выполняется терминальное условие, поиск на графе будет продолжаться до тех пор, пока «будет возможно» найти меньший путь. На рисунке 5.25 представлен граф поиска оптимального пути мобильным агентом, при транспортировке трех грузов. Из графа видно, что последовательность развозки должна быть следующей (2–3–1).

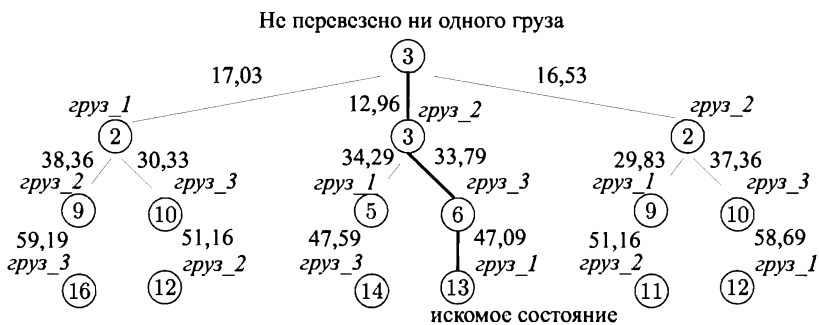


Рис. 5.25. Пример графа поиска

Рассмотрим случай, когда мобильный агент может перевозить только один груз, но он обслуживает заявки, как на ввоз, так и на вывоз грузов со стеллажей склада. Здесь необходимо решить, в какой последовательности необходимо увозить и привозить грузы, чтобы пройденный путь погрузчика был минимальным.

При планировании маршрута транспортных перевозок алгоритм поиска в ширину работает следующим образом: выбирается начальная вершина — пункт поступления грузов, т. е. начальная отправная точка движения транспортного устройства. Затем раскрываются все вершины, инцидентные начальной, и подсчитывается стоимость пути от начальной вершины до каждой из этих вершин. Вершины, связанные с начальной вершиной — это стеллажи, в которые необходимо доставить грузы или пункт приема грузов, куда надо доставить грузы со стеллажей. В качестве стоимости пути выступает путь, который должен пройти мобильный агент. В результате мы находим груз, который нужно отвезти в первую очередь. Затем раскрываются все вершины, инцидентные этой вершине, и определяются стоимости путей до остальных нераскрытых вершин, и так до тех пор, пока не будут раскрыты все вершины. В результате такого поиска мы находим оптимальную последовательность грузов, при этом грузам присваивается приоритет развоза по стеллажам и в пункт приема. Когда всем грузам, которые необходимо перевезти, присвоены приоритеты, начинается собственно развозка грузов и строится новый граф, где в качестве стоимости пути выступает приоритет груза. Таким образом, каждый агент сам выбирает, какой груз привезти к стеллажу, а какой отвезти, в зависимости от его приоритета (приоритет меньше у того груза, ехать до места назначения которого меньше).

Так как РДО-имитатор позволяет использовать не одну точку принятия решения, а несколько, если это необходимо, то для каждого мобильного агента в модели строится граф поиска в пространстве состояний. В результате, если в системе всего два транспортных устройства, то в модели строятся два графа состояний, т. е. реализуются две системы управления.

В модели с двумя мобильными агентами описание точек принятия решения на РДО выглядит следующим образом:

```

$Activities
  Появился_груз_который_надо_привезти :
  Поступление_запроса_на_поставку_груза
$End
$Decision_point Минимальный_путь : search trace_all
$Condition
  Exist (Грузы: Грузы.Состояние – ожидает) and For_All (Грузы:
  Грузы.Куда_везти <> 33) and For_All(Грузы: Грузы.Откуда_везти <> 33)
$Term_condition
  For_All (Грузы: Грузы.Состояние = готов_к_перевозке)
$Evaluate_by 0
$Compare_tops = YES
$Activities
  Освобождение : Освобождение_погрузчика
  Поиск          : Нахождение_минимального_пути * valueafter
                   Оценка(транспортер.Путь_для_поиска)
$End
$Decision_point Перевозка : search trace_all
$Condition

```

```

Exist(Грузы: Грузы.Состояние = готов_к_перевозке)
$Term_condition
For_All (Грузы: Грузы.Состояние = обслужен)
$Evaluate_by 0
$Compare_tops = YES
$Activities
  Отвести : Перевозка_груза_от_стеллажа value after груз.Приоритет
  Привезти : Перевозка_груза_к_стеллажу value after груз.Приоритет
  Освобождение_1 : Освобождение_погрузчика_1
$End

$Decision_point Промежуточная : some trace_all
$Condition NoCheck
$Activities
  Откуда_тип_1 : Установление_откуда_везти_груз_типа_1
  Куда_тип_1 : Установление_куда_везти_груз_типа_1
  Откуда_тип_2 : Установление_откуда_везти_груз_типа_2
  Куда_тип_2 : Установление_куда_везти_груз_типа_2
  Конец : Исчезновение_грузов
$End

```

Первый блок активностей необходим для случайного поступления запроса на заявку перевезти груз. Первая точка принятия решений «Минимальный_путь» обеспечивает поиск на графе. Иначе говоря, ищется последовательность перевозки грузов таким образом, чтобы пройденный путь агентом был минимальным. Во второй точке принятия решений «Перевозка» осуществляется собственно имитация перевозки либо к стеллажу, либо от него, и поиск на графе происходит в зависимости от приоритета груза. Третья точка принятия решений — вспомогательная, с помощью нее случайным образом устанавливается, куда и откуда вести случайно появившийся в системе груз.

Пусть имеются три груза (рис. 5.26): Груз_1 — надо перевезти к стеллажу 2; Груз_2 — надо отвезти от стеллажа 26; Груз_3 — надо отвезти от стеллажа 3.

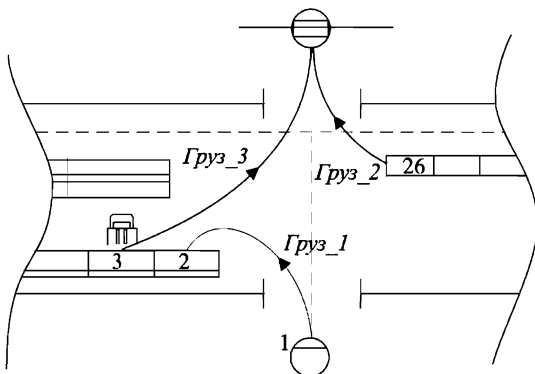


Рис. 5.26. Задача развозки трех грузов

После того, как генетический алгоритм «разделил» грузы по агентам, каждый из них решает оптимизационную задачу развоза грузов при минимизации собственного пути (рис. 5.28). Функция пригодности особи представляет собой сумму путей, пройденных каждым из мобильных агентов. Для ее оценки с учетом динамики протекающих в транспортной системе процессов используется имитационная модель, т.е. вся система создается как гибридная и разрабатывается в интегрированной многофункциональной имитационной среде РДО.

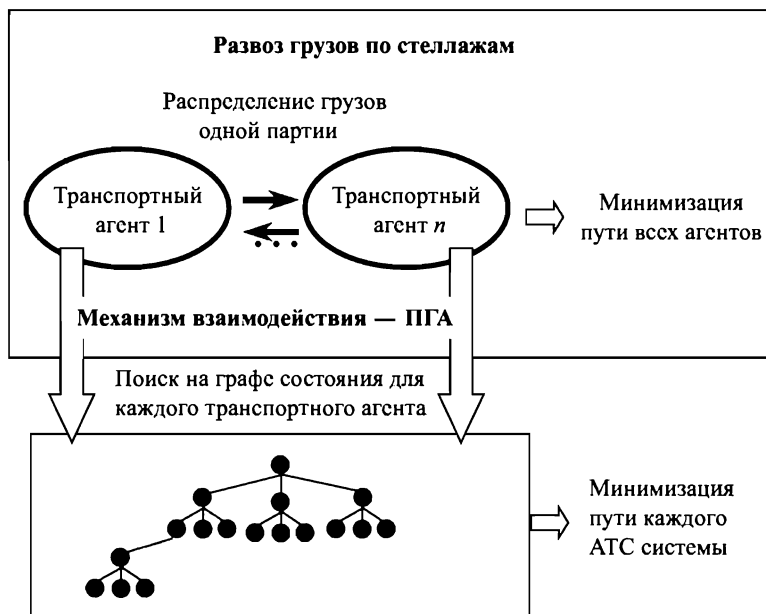


Рис. 5.28. Организация взаимодействия агентов

Таким образом, система строится как гибридная, в которой присутствуют разнородные подсистемы — оптимизации маршрутов, генетический алгоритм, имитационная модель для определения пройденного пути. Для ее реализации используется, как было сказано выше, среда РДО. РДО-имитатор, в отличие от большинства систем и языков имитационного моделирования, для принятия оптимальных решений имеет встроенный механизм точек принятия решений.

Был рассмотрен склад, на котором имелось 87 стеллажей, два мобильных агента, пункт поступления грузов. Данный склад предназначен для хранения 7 типов грузов, причем для каждого типа груза предна-

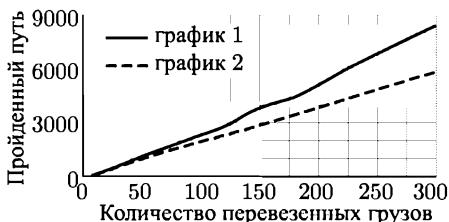


Рис. 5.29. Результаты моделирования

значен определенный стеллаж. Каждый тип груза поступает с различной интенсивностью. Мобильные агенты имели грузоподъемность 3 груза. Результаты моделирования приведены на графике (рис. 5.29). На графике 1 представлены результаты моделирования без оптимизации. На графике 2 представлена система управления, в которой используется ПГА для оптимизации распределения грузов между агентами и поиск на графе состояний при оптимизации каждым мобильным агентом своего оптимального пути.

5.6. Искусственная жизнь и МАС

Как уже указывалось выше в настоящей главе, сегодня основные направления в МАС — распределенный искусственный интеллект и искусственная жизнь (в узком смысле). Чаще всего исследователи рассматривают второе направление как выживание, адаптацию и самоорганизацию в динамической враждебной среде большого числа простых и, необязательно, интеллектуальных агентов. В этом плане рассмотренная популяция автоматов (§ 5.3) и представляет собой «коллективный интеллект» (collective intelligence) или «интеллект роя» (swarm intelligence). Здесь механизмы реакции агентов на воздействия внешней среды и локальных взаимодействий агентов не используют прогнозирование, планирование, знания и т. п., хотя и позволяют решать сложные задачи. Как мы видели, коллектив довольно простых автоматов демонстрирует не только адаптивные способности по отношению к среде, но и способность регулировать численность своей популяции в зависимости от внешней среды. Как один из следующих шагов в направлении развития искусственной жизни, рассмотрим популяцию интеллектуальных агентов, но не взаимодействующих друг с другом для достижения общей цели, как это рассматривалось в предыдущем разделе, а решающих некоторую общую задачу и существующих в эволюционирующей популяции. То есть, нас интересуют интеллектуальные агенты как некоторые особи, образующие популяцию, решающие некоторую задачу, и, соответственно, эволюция популяции при решении этой задачи (выполнении некоторой деятельности).

В большинстве исследований МАС рассматривается без учета ее эволюции как популяции агентов. Агенты взаимодействуют друг с другом, решают свои подзадачи, усиливают действия других агентов своими ресурсами и т. п., но не «живут» в популяции, хотя и адаптируют свои характеристики в процессе взаимодействия с другими агентами. Однако, если проводить параллели между сообществом таких интеллектуальных агентов и биологическими системами, то видна необходимость улучшения каждым агентом своего функционирования в процессе существования популяции, накопления этой популяцией жизненного опыта и, как следствие, увеличение объема знаний каждым агентом. Здесь уже каждый агент (или группа агентов), как в централизованной системе управления, решает полностью некоторую задачу, но делает это параллельно с другими агентами (или группами). Поэтому популяция таких агентов в своей эволюции получает сразу столько решений стоящей перед ними задачи, сколько имеется в ней интеллекту-

альных агентов (или групп агентов). Это принципиальное отличие МАС и системы искусственной жизни (в широком смысле этого понятия).

Рассмотрим далее один из подходов к реализации искусственной жизни, как эволюционирующей популяции интеллектуальных агентов. При этом цель рассмотрения данного подхода заключается в повышении качества (или сокращении времени вычислений) при решении множества подобных задач. Иначе говоря, речь идет о параллельном решении некоторой задачи множеством агентов, когда каждый из них находит решение, и, как следствие, мы имеем в результате сразу множество решений, отличающихся друг от друга значением функции пригодности.

При представлении сложной системы в виде МАС используется некоторое множество когнитивных агентов $\{A_1, A_2, \dots, A_k\}$. При этом для оптимального управления некоторой сложной системой с помощью данной МАС необходимым является оптимизация поведения каждого агента в МАС. Каждый i -й агент характеризуется набором параметров $(a_{i1}, a_{i2}, \dots, a_{in})$, которые и определяют его поведение в системе. Рассмотрим систему однотипных по структуре когнитивных агентов с одинаковым набором параметров:

$$\text{MAC} = \{A_1(a_{11}, a_{12}, \dots, a_{1n}), A_2(a_{21}, a_{22}, \dots, a_{2n}), \dots, A_k(a_{k1}, a_{k2}, \dots, a_{kn})\}.$$

В качестве примера такой системы можно привести производственный участок, состоящий из нескольких станков, транспортного робота, погрузочно-разгрузочного робота и конвейера. Здесь агент A_i моделирует i -й станок, а его параметры $\{a_{i1}, a_{i2}, \dots, a_{in}\}$ определяют приоритеты станка на вызов транспорта, на вызов погрузочно-разгрузочного робота, на погрузку деталей в тележку, на разгрузку деталей с конвейера и т. п.

Для решения задачи в МАС необходимо оптимизировать поведение каждого агента в системе. Это значит, что необходимо найти такую комбинацию параметров агента, при которой его поведение в системе будет оптимальным:

$$\text{MAC}^{\text{opt}} = \{A_1^{\text{opt}}, A_2^{\text{opt}}, \dots, A_k^{\text{opt}}\},$$

где

$$A_1^{\text{opt}} : A_1(a_{11}, a_{12}, \dots, a_{1n}) \rightarrow \text{opt};$$

$$A_2^{\text{opt}} : A_2(a_{21}, a_{22}, \dots, a_{2n}) \rightarrow \text{opt};$$

.....

$$A_k^{\text{opt}} : A_k(a_{k1}, a_{k2}, \dots, a_{kn}) \rightarrow \text{opt}.$$

В основу модели положена эволюционная стратегия, включающая три механизма: воспроизведение, скрещивание и мутация. В некотором пространстве поиска создается популяция особей-агентов (их также можно назвать реактивными агентами), в которой один реактивный агент связан с одним когнитивным агентом МАС. У каждого реактивного агента име-

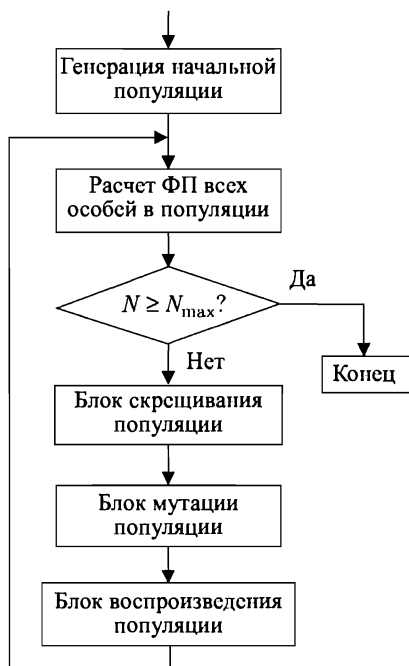


Рис. 5.30. Схема алгоритма эволюции

ведение популяции) позволяют оптимизировать их функции пригодности и, как следствие, оптимизировать поведение когнитивных агентов исходной МАС.

Эволюционная стратегия реализуется следующим образом (рис. 5.30).

1. **Генерация начальной популяции.** На всем пространстве поиска случайным образом создается начальная популяция особей. Пространство поиска, число особей в популяции и параметры особей определяются из условий оптимизационной задачи.

2. **Расчет ФП всех особей популяции.** Определяются ФП каждой особи в популяции. При необходимости рассчитываются максимальные и минимальные ФП, а также определяется средняя ФП по популяции.

3. $N \geq N_{\max}$? Проверяется условие окончания работы алгоритма, а именно, условие окончания развития популяции, при котором текущее число поколений N становится больше или равно максимальному числу поколений для данной популяции $N_{\max}$.

4. **Блок скрещивания популяции.** Реализует скрещивание особей в популяции. В данном блоке происходит формирование потомков особей. Каждая особь в популяции может иметь три личных потомка. Первый потомок — «обязательный», образуется в результате самовоспроизведения особи, а два других — «необязательные», образуются в результате скрещивания

есть своя (отличная от других) целевая функция или функция пригодности, которую он оптимизирует в процессе своего развития в популяции. В формулу для расчета функции пригодности реактивного агента входят параметры связанного с ним когнитивного агента или параметры, зависящие от параметров когнитивного агента. Данная формула составляется таким образом, что при достижении оптимума функции пригодности реактивного агента параметры когнитивного агента становятся оптимальными. Необходимо отметить, что все значения, которые может принимать функция пригодности каждого реактивного агента, лежат в пространстве поиска всей популяции реактивных агентов. Коллективное поведение реактивных агентов, их взаимодействие друг с другом посредством законов эволюции (скрещивание и мутация) и параллельное развитие (воспроиз-

с другой особью в популяции. Схема алгоритма скрещивания изображена на рис. 5.31. Скрещивание особей в популяции происходит следующим образом:

- **Выбор очередной особи в популяции (Особь i).** Выбирается очередная особь в популяции с номером i . Параметр i представляет собой значения счетчика, изменяющегося в пределах от 1 до числа особей в популяции. Таким образом перебираются по очереди все особи в популяции.

- **Подбор пары (Особь j) для i -й особи.** Реализует подбор пары для i -й особи. Для этого случайным образом выбирается Особь j из всех особей в популяции. Если при этом выбралась Особь i , т.е. $i = j$, тогда процесс выбора пары для i -й особи повторяется.

- **$P(OK) > P(OK)_3$?** Определяется, произойдет ли скрещивание между очередными двумя особями в популяции. Для этого случайным образом на интервале $(0, 1)$ генерируется число $P(OK)$ — вероятность скрещивания. Если $P(OK)$ больше, чем заданная вероятность скрещивания $P(OK)_3$, то происходит скрещивание между данными особями. При этом Особь i будет иметь три потомка. Если же данное условие не выполняется, то скрещивания не происходит, и Особь i будет иметь одного потомка.

- **Определение 2-го и 3-го потомков i -й особи.** Реализуется в случае выполнения условия скрещивания особей. При этом образуются второй и третий потомки i -й особи посредством оператора скрещивания. Оператор скрещивания представляет собой механизм одноточечного скрещивания двух особей с разрывом хромосом по одной позиции.

- **Определение 1-го потомка i -й особи.** Реализует получение 1-го потомка i -й особи посредством самовоспроизведения. При этом образуется потомок, который является точной копией родителя. Необходимо отметить, что, независимо от условий скрещивания, для каждой особи в популяции определяется первый потомок.

- **Рассмотрены все особи?** Проверяет условие окончания блока скрещивания для популяции. Если все особи в популяции получили потомков, то осуществляется переход к следующему блоку алгоритма.

5. Блок мутации популяции. Реализует мутацию потомков особей в популяции. Необходимо отметить, что в данной схеме мутации подвержены

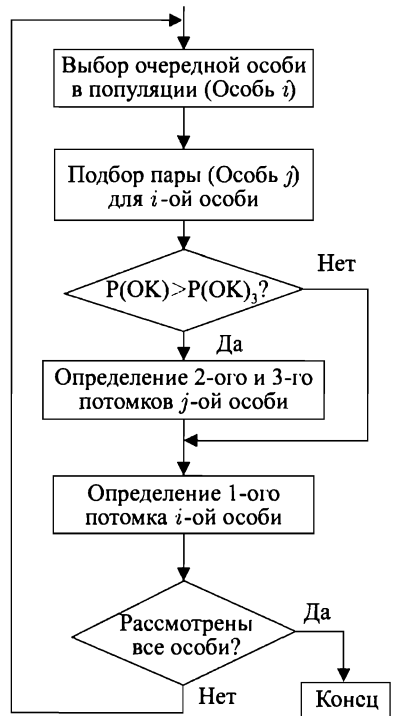


Рис. 5.31. Схема скрещивания

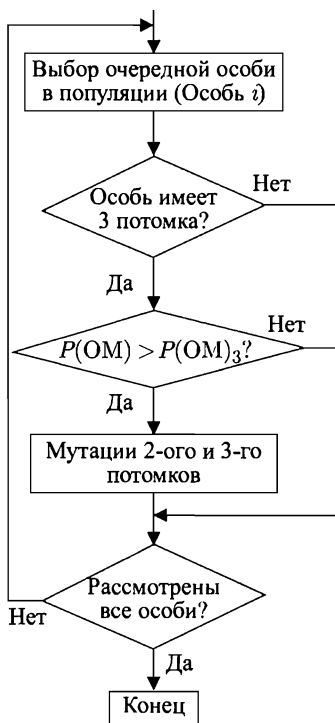


Рис. 5.32. Схема блока мутации

только второй и третий потомки каждой особи. Сделано это с целью поддержания накопленного уровня развития «эволюционной линии» каждой особи. При этом гарантируется, что каждая особь в последующем поколении не хуже своего родителя из предыдущего поколения. Схема мутации изображена на рис. 5.32. Мутация производится следующим образом:

- **Выбор очередной особи в популяции (Особь i).** Выбирается очередная особь в популяции с номером i . Параметр i представляет собой значения счетчика, изменяющегося в пределах от 1 до числа особей в популяции. Таким образом, перебираются по очереди все особи в популяции.

- **Особь имеет 3 потомка?** Если особь имеет три потомка, тогда переход к следующему блоку. Если же особь имеет одного потомка, то мутации потомков не происходит.

- **$P(OM) > P(OM)_3$?** Определяется, произойдет ли мутация 2-го и 3-го потомков i -й особи. Для этого случайным образом на интервале $(0, 1)$ генерируется число $P(OM)$ — вероятность мутации. Если $P(OM)$ больше, чем заданная вероятность

мутации $P(OM)_3$, то происходит мутация 2-го и 3-го потомков.

- **Мутация 2-го и 3-го потомков.** Реализуется в случае выполнения условия мутации. При этом применяется оператор мутации ко второму и третьему потомкам i -й особи. Оператор мутации представляет собой механизм однобитной мутации, при которой инвертируется один бит строки-хромосомы.

- **Рассмотрены все особи?** Проверяет условие окончания блока мутации для популяции. Если для потомков всех особей в популяции получили потомков, то осуществляется переход к следующему блоку алгоритма.

6. Блок воспроизведения популяции. Реализует воспроизведение особей и формирование новой популяции. Для каждой особи в последующую популяцию отбирается один лучший потомок (с наибольшей ФП), тем самым, продолжая «эволюционную линию» данной особи. Особь родитель при этом погибает. Таким образом, число особей в последующей популяции равно числу особей в предыдущей популяции и неизменно при переходе популяции от поколения к поколению. Схема воспроизведения популяции изображена на рис. 5.33. Она работает следующим образом:

- **Создание новой популяции особей.** Создается новая популяция особей, которая будет состоять из потомков особей текущей популяции, причем

в новую популяцию отбирается по одному потомку от каждой особи текущей популяции. Номер поколения N для новой популяции увеличивается на единицу по сравнению с номером поколения текущей популяции.

- **Выбор очередной особи в популяции (Особь i).** Выбирается очередная особь в текущей популяции с номером i . Параметр i представляет собой значения счетчика, изменяющегося в пределах от 1 до числа особей в популяции. Таким образом перебираются по очереди все особи в популяции.

- **Выбор лучшего потомка i -й особи (Потомок i).** Если Особь i имеет три потомка, то выбирается лучший (с максимальной ФП) потомок. Если же особь имеет одного потомка, то он и является лучшим потомком i -й особи.

- **Добавление в новую популяцию потомка i .** Потомок i добавляется в новую популяцию под номером i , тем самым, продолжая «эволюционную линию» i -й особи.

- **Рассмотрены все особи?** Проверяет условие окончания блока воспроизведения популяции. Если новая популяция полностью сформирована, то старая популяция удаляется.

- **Удаление старой популяции особей.** Старая популяция удаляется. При этом «погибают» старые особи.

Покажем работу модели «Искусственной жизни» на тестовой задаче. Задача заключается в следующем.

Имеется дискретное (с шагом 2^{-8}) двумерное пространство с изменением координат от 0 до 1. В пространстве имеется 25 точек-оптимумов (для каждой особи-агента), равноудаленных друг от друга. В данном пространстве «живут» 25 особей-агентов и каждая особь-агент имеет свой оптимум, к которому стремится в результате развития популяции. Визуально пространство поиска показано на рисунке (рис. 5.34). Здесь в кружках, равномерно распределенных по пространству поиска, проставлены номера агентов, для которых данные кружки являются оптимальным расположением по отношению к другим агентам.

Для сравнения, эта же задача решается также методом случайного перебора, в котором: количество генераций случайных особей-агентов на каждом поколении равно числу агентов потомков в модели «Искусственная жизнь». Сделано это для того, чтобы уравнивать вычислительные

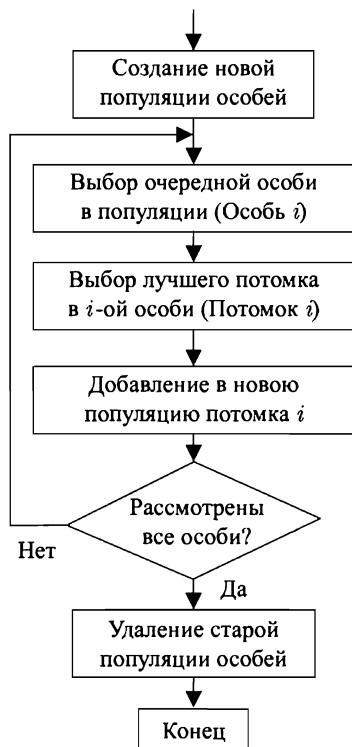


Рис. 5.33. Схема воспроизведения

ресурсы, затрачиваемые в каждой из моделей, что необходимо для сравнения результатов.

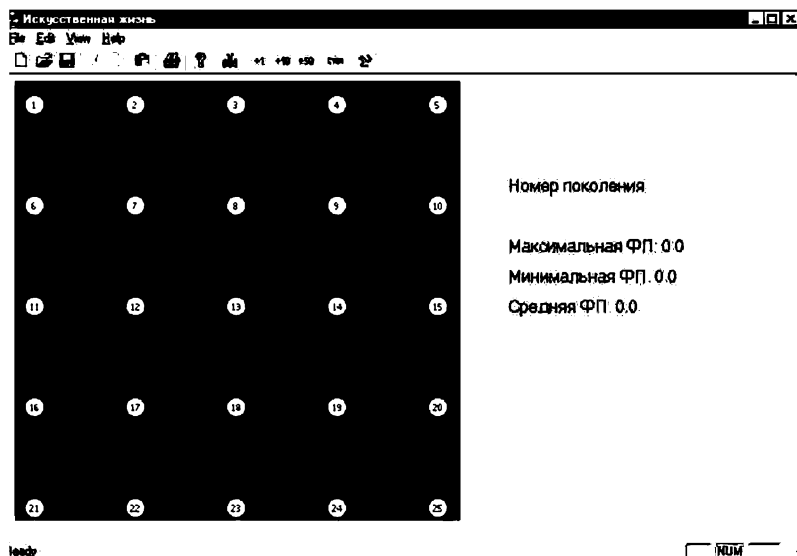


Рис. 5.34. Пространство поиска тестовой задачи

Агенты графически представлены в виде жучков с соответствующими номерами. В процессе эволюции своих популяций они должны (если эволюционный алгоритм сходится) занять соответствующие места на пространстве поиска. Процесс развития популяций в обеих моделях приведен на экранных формах имитатора. Приведены расположения агентов на 5, 2 и 100 поколениях развития популяций для обеих моделей. (рис. 5.35, а, б, в).

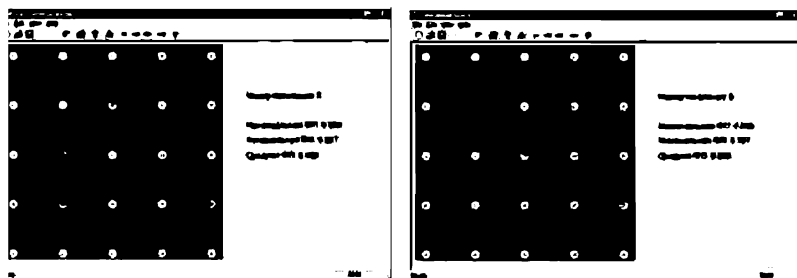


Рис. 5.35, а. Вид 5-ой популяции (справа случайный поиск)

Из экранных форм видно, что агенты в модели искусственной жизни 100-й популяции оптимизировали свои параметры, в то время как при случайном поиске еще имеются рассогласования текущих значений параметров и оптимальных. Сравнительные результаты имитации эволюции

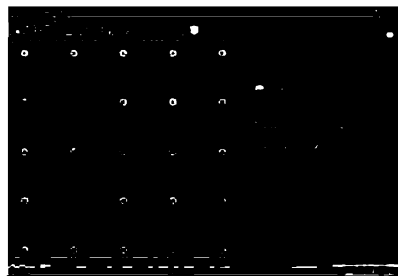


Рис. 5.35, б. Вид 20-ой популяции (справа случайный поиск)

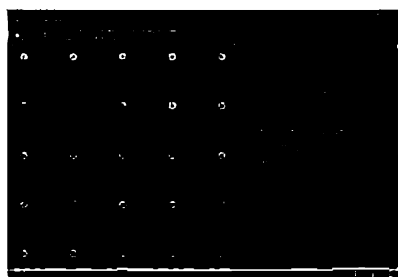


Рис. 5.35, в. Вид 50-ой популяции (справа случайный поиск)

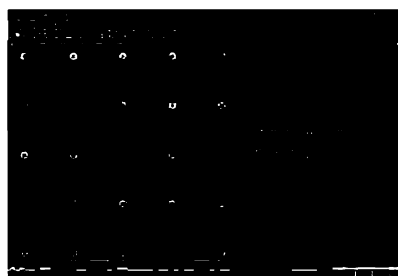
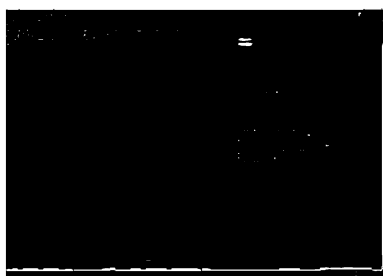


Рис. 5.35, г. Вид 100-ой популяции (справа случайный поиск)

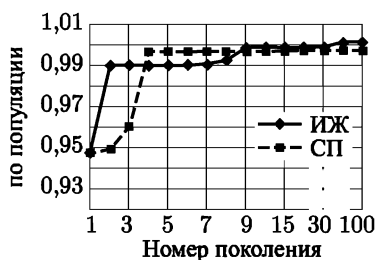


Рис. 5.36, а. Сравнение зависимостей максимальных ФП в процессе развития



Рис. 5.36, б. Сравнение зависимостей минимальных ФП в процессе развития

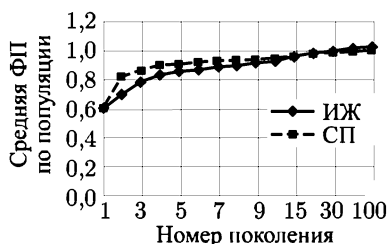


Рис. 5.36, в. Сравнение зависимостей средних ФП в процессе развития популяции

популяций агентов по поколениям приведены на графиках, здесь сравниваются максимальные значения ФП (рис. 5.36, а), минимальные (рис. 5.36, б) и средние значения ФП (рис. 5.36, в).

Из графиков видно, что модель Искусственной жизни медленнее выходит на установившееся функционирование, но при этом обеспечивает лучшее решение, чем случайный поиск. Следовательно, можно говорить о перспективности развития данного направления для решения однотипных задач популяцией интеллектуальных агентов.

5.7. Многомодельные системы

Помимо взаимодействия друг с другом, функционирование агентов в МАС связано с проблемами формализации, принятием решений, его информационной поддержкой, адаптацией к изменяющимся условиям и многими другими. Для этого агенты должны использовать моделирование, обеспечивающее прогностические свойства агентов и, следовательно, возможность принятия эффективных решений, адаптируемости и т. п.

Так как сложная система принципиально является многоаспектной, то она не может быть описана одной моделью, и необходимо разрабатывать ряд моделей одной и той же системы, предназначенных для решения различных задач или только одной задачи. Имитационные модели часто являются единственным методом при исследовании и управлении сложными дискретными системами и процессами. Большие возможности, обеспечиваемые использованием имитационных моделей, не означают, что наряду с ними не могут быть применены статистические, аналитические и другие модели. Поэтому создание децентрализованной системы управления приводит к построению гибридных гетерогенных структур моделей (далее многомодельных структур), обеспечивающих моделирование структур сложной системы, поддерживающих процесс принятия решений и эволюцию системы.

Широкий круг решаемых задач локальными системами управления, входящих в децентрализованные системы управления, приводит к тому, что должен быть разработан ряд моделей различных видов и различной степени детализации, каждая из которых ориентирована на решение определенных специфических задач. При этом необходимо, чтобы эти модели можно было использовать совместно друг с другом, т. е. осуществлять некоторый информационный обмен на уровне моделей. Для этого они должны использовать единый формат информационной среды, обмениваться знаниями и данными, допускать легкую настройку на конкретные условия, использовать результаты работы друг друга, допускать динамическую мо-

дификацию в процессе эксплуатации по мере накопления знаний и данных об исследуемом объекте. Это взаимодействие моделей должно происходить в процессе решения интеллектуальным агентом задач принятия решений. Следовательно, стоит задача создания системы моделей для решения отдельных задач и средств (метамodelей) их «взаимопонимания», а также средств обеспечения взаимодействия моделей в рамках многоагентной системы (рис. 5.37). Возникает некоторая совокупность моделей, в которой имеет место преобладание горизонтальных связей над вертикальными.

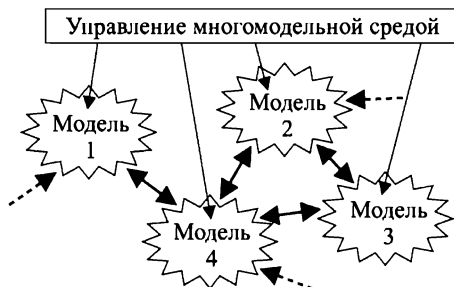


Рис. 5.37. Условное представление многомодельной среды

Многомодельная среда, создаваемая на основе комплексного использования имитационного моделирования, методов исследования операций, теории принятия решений и математической статистики, накладывается на структуру децентрализованной системы управления, т. е. на структуру многоагентной системы.

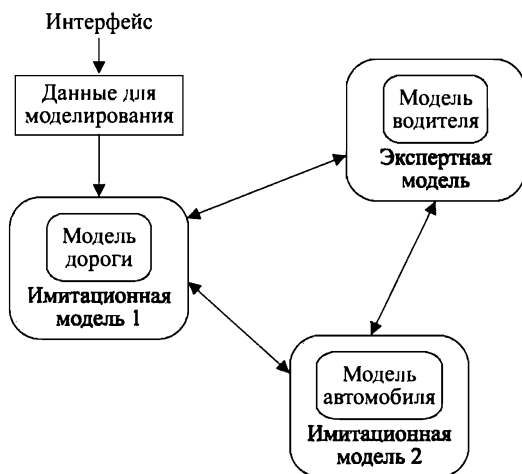


Рис. 5.38. Пример многомодельной среды

Каждый отдельный агент обладает определенным объемом знаний и должен иметь доступ к общей информационной базе гибридной многомо-

дельной системы, т. е. иметь возможность использовать эту информацию при принятии решений. Это приводит к некоторому уровню централизации принятия решений.

На рис. 5.38 приведена структура гибридной системы, включающей в себя две имитационные модели (автомобиля и участка дороги) и ЭС, моделирующую принятие решений водителем. Здесь модели ориентированы на решение определенных специфических задач. При этом необходимо, чтобы эти модели можно было использовать совместно друг с другом, т. е. осуществлять некоторый информационный обмен на уровне моделей. Для этого они должны использовать единый формат информационной среды, обмениваться знаниями и данными, допускать легкую настройку на конкретные условия, использовать результаты работы друг друга, допускать динамическую модификацию в процессе эксплуатации по мере накопления знаний и данных об исследуемом объекте. Это взаимодействие моделей должно происходить в процессе принятия решений. Возникает некоторая совокупность моделей, в которой имеет место преобладание горизонтальных связей над вертикальными. Следовательно, мы имеем некоторую популяцию, где особь представляет собой некоторую модель, а вся популяция — это модель предметной области.

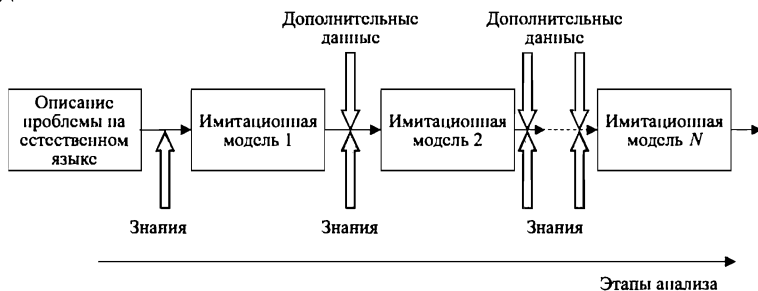


Рис. 5.39. Эволюция моделей в процессе анализа сложной системы

Многомодельная среда, создаваемая на основе комплексного использования имитационного моделирования, методов исследования операций, теории принятия решений и математической статистики, накладывается на структуру децентрализованной системы управления. Каждая отдельная система принятия решений должна обладать определенным объемом знаний и должна иметь доступ к общей информационной базе гибридной многомодельной системы, т. е. иметь возможность использовать эту информацию при принятии решений. Это приводит к некоторому уровню централизации принятия решений.

Вопросы эволюции моделей в многомодельной среде, насколько нам известно, пока никем не рассматривались, хотя имеется ряд работ по интеграции моделей, их взаимодействию и пр. [157, 158].

Следует отметить, что процесс разработки имитационных моделей и проведение на них исследований представляет собой достаточно сложный итерационный процесс, который можно рассматривать как процесс эволюции (рис. 5.39). Дело в том, что такая модель не может быть построена один

раз для сложной системы. По мере исследований появляется много новой уточняющей информации, снимается часть как внешней, так и внутренней неопределенности модели. Более того, имитационная модель сама выступает как некоторое знание о моделируемой системе и источник новых знаний о ней. поэтому по мере эволюции сложной системы должна происходить эволюция моделей, используемых в подсистемах управления.

5.8. Генетический поиск с миграцией особей

Генетические алгоритмы, в которых используется одна популяция особей для поиска решения, не всегда оказываются эффективными. Основной их недостаток — попадание в локальный оптимум. В § 4.2 уже была представлена модифицированная схема миграции с искусственной селекцией (рис. 4.8), основанная на механизмах макроэволюции. В данном разделе рассмотрим еще одну модифицированную схему генетического поиска с миграцией особей. В этой схеме на макроуровне параллельно эволюционируют две идентичных популяции, между которыми существует миграция особей (рис. 5.40). Обе популяции осуществляют поиск наилучшего решения для одной и той же задачи. Миграция особей призвана «взбалтывать» популяции и выводить их из локальных оптимумов.

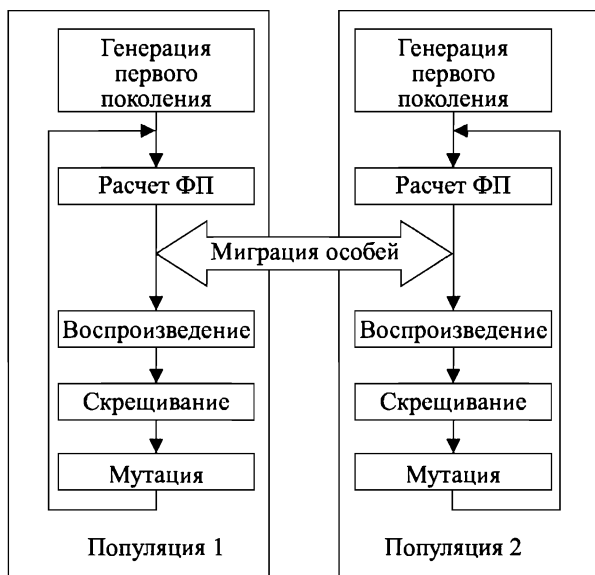


Рис. 5.40. Схема генетического поиска с миграцией особей

В приведенной на рисунке схеме в каждой популяции используется ПГА. Все основные параметры работы ПГА для каждой популяции, такие как число особей в популяции, вероятность скрещивания и вероятность мутации, приняты одинаковыми. На этапе работы ПГА между операторами Расчет ФП и Воспроизведение происходит миграция особей, т. е. обмен

одинаковым количеством особей между популяциями, который называется процессом миграции.

Миграция особей может происходить по разным законам. По аналогии с человеческим обществом это может быть «утечка мозгов», когда популяцию покидают особи с наилучшими значениями ФП, или миграция «низкоквалифицированной дешевой рабочей силы», когда мигрируют особи с наименьшим значением ФП. Помимо качественного состава мигрантов (значения ФП) на эффективность миграции может оказывать влияние и его количественный состав (какой процент популяции участвует в обмене). Таким образом, в данной схеме процесс миграции определяется двумя величинами — числом мигрирующих особей и их ФП:

1. k — число мигрантов. Данный параметр определяет число мигрантов, которое переходит из популяции в популяцию на каждом шаге работы алгоритма.

2. $\Delta\text{ФП}$ — способ миграции. Данный параметр определяет, какие особи будут использоваться для миграции, и рассчитывается на основе ФП мигрирующих особей:

$$\Delta\text{ФП} = \text{ФП}_1 - \text{ФП}_2,$$

где ФП_1 — значение ФП особи из популяции 1; ФП_2 — значение ФП особи из популяции 2.

В гибридных системах выбор оптимальных значений этих переменных должна обеспечивать ЭС, основываясь на анализе работы схемы при решении оптимизационной задачи конкретной проблемной области. Мы же исследуем свойства рассматриваемой схемы на примере решения простой оптимизационной задачи (см. пп. 5.2.1). Задача оптимизации заключается в нахождении переменных X_1 и X_2 , на которых достигается максимум значения функции: $f = \frac{X_1^2 + X_2^2}{2} \rightarrow \max$. Область определения переменных — $[0, 1]$. Как следует из вида функции, максимальное значение достигается в точке с координатами (1, 1) и оно равно 1.

При решении задачи с помощью ПГА переменные X_1 и X_2 кодируются в виде битовой строки длиной l битов каждая. Значения переменных определяются следующим образом: битовая строка рассматривается как двоичное целое без знака в диапазоне $[0, 2^l - 1]$, это число делится на $2l$. Следовательно, получаемое при этом значение лежит в диапазоне $[0, 1 - 2^{-l}]$, а пространство оптимизации становится дискретным с шагом сетки 2^{-l} .

Исследования проводились на имитационной модели, моделирующей развитие популяций и миграцию особей между ними.

Количество поколений работы алгоритма модифицированной схемы с миграцией зависит от момента остановки алгоритма, который, в свою очередь, в данной схеме определяется из условия:

$$(\text{ФП}_{\max} - \text{ФП}_{\text{ср}}) / \text{ФП}_{\max} < 5\%,$$

где $\text{ФП}_{\max}$ — максимальное значение функции пригодности для особи

в пределах обеих популяций; $\Phi P_{\text{ср}}$ — среднее значение функции пригодности по популяции, в которой существует особь с ΦP_{max} .

Количество особей в каждой популяции модифицированной схемы определяется в зависимости от доверительного интервала разброса значений ФП для особей, который определяется по формуле

$$L = (\Phi P_{\text{max}} - \Phi P_{\text{min}}) / \Phi P_{\text{max}} \cdot 100\%,$$

с доверительной вероятностью $P = 90\%$; где ΦP_{min} — минимальное значение функции пригодности по популяции, в которой существует особь с ΦP_{max} .

Для данной задачи была получена зависимость доверительного интервала в установившемся режиме от числа особей в популяции, которая позволяет выбрать наиболее рациональное число особей в популяции с учетом времени работы алгоритма. Данная зависимость приведена на рис. 5.41.

Так как время работы алгоритма пропорционально числу особей в популяции, то необходимо выбрать такое число особей в популяции, при котором достигается достаточная точность работы алгоритма, это зависит от величины доверительного интервала и относительно небольшого времени работы алгоритма. Учитывая это, окончательно выбираем по зависимости, изображенной на рис. 5.41, число особей в популяции, равное 20. При этом значение доверительного интервала $L = 0,07$, с доверительной вероятностью $P = 0,9$. Вероятность скрещивания в ПГА принимаем равной 0,6, вероятность мутации равна 0,1.

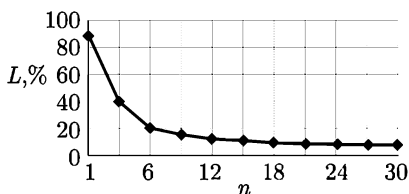


Рис. 5.41. Доверительный интервал в установившемся режиме

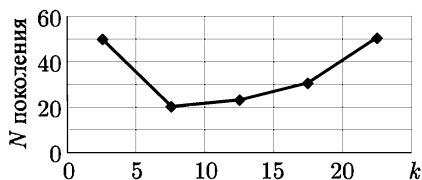


Рис. 5.42. Зависимость скорости сходимости от числа мигрантов

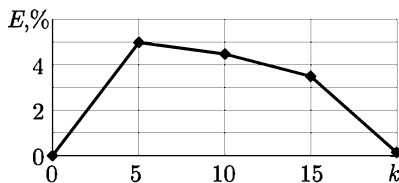


Рис. 5.43. Зависимость эффективности от числа мигрантов

На рис. 5.42 и 5.43 приводятся зависимости скорости сходимости алгоритма (модифицированной схемы) и эффективности использования модифицированной схемы от числа мигрантов (т. е. первого параметра процесса миграции).

Эффективность модифицированной схемы оценивается в сравнении с ПГА и рассчитывается по формуле:

$$E = \frac{\Phi P_M - \Phi P_{\text{ПГА}}}{\Phi P_M} \cdot 100 (\%),$$

где $\Phi_{ПМ}$ — среднее (по результатам 10 экспериментов) значение максимальных $\Phi_{П}$ при использовании модифицированной схемы генетического поиска; $\Phi_{ППА}$ — среднее (по результатам 10 экспериментов) значение максимальной $\Phi_{П}$ при использовании ПГА.

Из приведенных зависимостей видно, что при числе мигрантов, равном 5, достигается наибольший эффект. Количество особей в популяции равно 20, поэтому наибольший эффект достигается при числе мигрантов, равном 25% от числа особей в популяции.

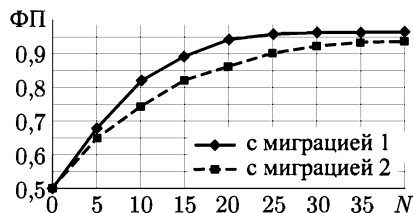


Рис. 5.44. Изменение $\Phi_{П}$ по поколениям для различных случаев миграции

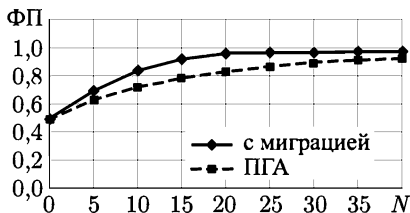


Рис. 5.45. Изменение $\Phi_{П}$ по поколениям

На рис 5.44 изображены графики изменения $\Phi_{П}$ по поколениям для различных случаев миграции, т. е. в зависимости от значения $\Delta\Phi_{П}$. Кривая «С миграцией 1» имеет значение $\Delta\Phi_{П1}$, а кривая «С миграцией 2» — значение $\Delta\Phi_{П2}$, причем $\Delta\Phi_{П1} > \Delta\Phi_{П2}$. Анализируя данные зависимости, можно сделать вывод о том, что при $\Delta\Phi_{П} = \max$ достигается наибольший эффект от миграции особей. То есть, чем больше разница $\Phi_{П}$ у особей мигрирующих из разных популяций, тем эффективнее работает схема.

На рис 5.45 изображены графики изменения $\Phi_{П}$ по поколениям для модифицированной схемы с миграцией и ПГА, по которым визуально можно видеть эффект от миграции особей, для простейшей задачи оптимизации.

6.1. Постановка оптимизационных задач

Выражения, отображающие зависимость между входными и выходными переменными, а также переменными состояния, называют математическим описанием модели решаемой задачи [113, 159]:

$$\left\{ \begin{array}{l} y_1 = y_1(x_1, \dots, x_n; z_1, \dots, z_r), \\ \vdots \\ y_m = y_m(x_1, \dots, x_n; z_1, \dots, z_r). \end{array} \right.$$

$$Y = F(X, Z). \quad (6.1)$$

Условно модель F можно представить как ее структуру St и вектор параметров $C = (c_1, c_2, \dots, c_k)$:

$$F = \langle St, C \rangle.$$

Создание модели осуществляется в два этапа:

- структурного синтеза, когда определяется структура Sl , т. е. вид элементов, из которых состоит модель и отношения между ними;
- параметрического, связанного с определением численных значений параметров модели $C = (c_1, c_2, \dots, c_k)$.

По известной структуре описания задачи и значениям векторов X и Z создают физические или математические модели F и на их основе определяют оценки значений вектора Y . Процедуры анализа систем обычно выполняют методом моделирования. Одновариантный анализ включает задачи статики и динамики в частотной области. Многовариантный анализ заключается в определении изменений вектора Y при заданных изменениях векторов X и Z . Многовариантный анализ включает анализ чувствительности и статистический анализ. При анализе чувствительности оценивают влияние внутренних и внешних переменных на выходные переменные путем расчета коэффициентов чувствительности. Статистический анализ заключается в определении вида закона распределения и расчете параметров распределения вектора Y при заданных статистических значениях о распределении случайных векторов X и Z .

Функционально математические модели обычно отражают свойства исследуемой системы, связанные с различными процессами ее функционирования. Функциональные математические модели чаще всего представляют в виде систем уравнений, описывающих фазовые параметры, внутренние, внешние и выходные переменные. Структурные математические модели обычно отображают структурные, в частности, геометрические, свойства объектов моделирования. Структурные математические модели делятся на топологические и геометрические. Топологические дают сведения о составе и взаимосвязи элементов моделируемого объекта и, в основном, используются при принятии решений, когда осуществляется привязка конструктивных элементов к определенным позициям плоскости. Для представления топологических математических моделей применяют множества, отношения, графы, гиперграфы, матрицы, таблицы, списки, фреймы.

Геометрические модели дают сведения о геометрических свойствах объектов, их взаимном расположении, форме и т. п. Они обычно задаются совокупностью уравнений, линий и поверхностей, алгебраических соотношений, графами и фреймами, отображающими конструкциями и используются, в основном, в инженерных задачах при конструкторском, функциональном и технологическом проектировании.

Технологические математические модели отображают свойства, связанные с процессами переработки информации и технологиями изготовления объекта. Их используют при задании исходных данных на разработку технологических процессов, построении технологических карт и т. п. В качестве технологических математических моделей применяются графы, гиперграфы, их матричные и списковые представления.

Теоретические математические модели получают на основе изучения физических закономерностей функционирования объекта, построения аналитических зависимостей, асимптотических оценок и т. п.

Эмпирические математические модели строят на основе опыта ЛПР при детальном изучении внешних проявлений свойств модели. Отметим, что каждому уровню или аспекту при иерархическом подходе к принятию решений соответствуют свои модели. Считают, что математические модели на микроуровне описывают физические процессы, которые протекают в пространстве и времени. Математические модели на микроуровне представляют, в основном, в виде дифференциальных уравнений в частных производных. Здесь независимыми параметрами являются пространственные координаты и время. Используя эти уравнения, рассчитывают поля механических напряжений и деформаций, температуру, давление и др. Математические модели в виде таких уравнений используют только для отдельных элементов или небольших фрагментов моделируемых систем.

При разработке математических моделей на макроуровне используют укрупненную дискретизацию пространства (плоскости в частном виде) по различным функциональным признакам. На этом уровне математические модели, как правило, записывают в виде систем обыкновенных дифференциальных уравнений. В них независимым параметром является время, а зависимые переменные представляются вектором фазовых переменных, которые характеризуют состояние укрупненных элементов дискретного пространства. В качестве таких переменных можно указать силу тока, напряжение, давление и т. п. Известно, что такие уравнения являются универсальными моделями на макроуровне, так как они пригодны для анализа как динамических, так и статических состояний модели. Порядок указанных уравнений обычно зависит от числа рассматриваемых элементов. Если он превышает 100, то исследовать такие модели затруднительно и, чаще всего, переходят к построению математических моделей на метауровне.

В этом случае в качестве элементов выбирают фрагменты блоков или их сложные совокупности. Как правило, математические модели на метауровне также представляют в виде различных уравнений. Основное отличие от математических моделей на макроуровне в том, что здесь не описываются внутренние для элементов фазовые переменные, а рассматриваются только переменные, относящиеся к взаимным связям элементов. Такой подход позволяет строить математические модели, размерность которых приемлема для записи и обработки информации на ЭВМ. В основном, к математическим моделям на метауровне относят модели массового обслуживания, которые применяют для анализа и описания процессов функционирования сложных систем.

Полной математической моделью называют модель, которая получается непосредственным объединением моделей в общую укрупненную систему. Тогда аппроксимацию полной математической модели называют макро-моделью [159]. Она описывает объекты при укрупненном рассмотрении элементов.

Аналитические математические модели обычно представляются формулами, описывающими зависимость между выходными, входными и внутренними параметрами.

Алгоритмическая математическая модель задает отношение или соответствие между выходными внутренними и внешними переменными, записанными в форме алгоритмов.

Имитационная модель позволяет показать поведение моделируемой системы во времени при задании внешних воздействий. Другими словами, имитационная модель — это по существу алгоритм или программа, а эксперимент над программой соответствует рассмотрению результатов расчетов по этой программе. Примерами имитационных моделей являются модели динамических объектов, описываемых в виде различных уравнений, соотношений, логических выражений, схем алгоритмов, и разнообразных моделей систем массового обслуживания. Имитационные модели используются при решении инженерных задач в неопределенных и нечетких условиях, т. е. в тех случаях, когда не удастся построить модели других видов.

Аналоговые математические модели — это модели, свойства которых определяются законами, аналогичными законам изучаемой системы, а переменные и параметры — непрерывные величины.

Дискретные математические модели — это модели, в которых все переменные и параметры — дискретные величины.

Опишем теперь требования, предъявляемые к математическим моделям. Основными являются адекватность, точность, степень универсальности и экономичность.

Адекватность математической модели — это соответствие модели моделируемой задаче или процессу принятия решений, причем адекватность рассматривают по тем свойствам модели, которые для ЛПР являются наиболее важными в данный момент времени. Под адекватностью математической модели также понимают способность отображать ее заданные свойства с текущей погрешностью ε_T , не выше заданной погрешности ε_3 . Математическая модель называется адекватной по вектору Y , если погрешность расчета на ее основе значений выходных параметров $y_i \in Y$ не превышает заданных. Как известно, адекватность математической модели обычно рассматривают в ограниченной области изменения входных переменных. Эту область называют областью адекватности математической модели. В ней выполняется неравенство $|\varepsilon_i| \leq \varepsilon_{i,g}$, где ε_i — относительная погрешность определения переменной y_i , возникающая из-за приближенного характера математической модели; $\varepsilon_{i,g}$ — допустимая погрешность ($\varepsilon_{i,g} \geq 0$) [159].

Отметим, что любая математическая модель описывает лишь некоторое подмножество свойств задачи, поэтому ее точность определяется как степень совпадения значений переменных реального объекта и значений тех же переменных, полученных на основе исследуемой математической модели. При определении точности математических моделей важно определять их погрешности. Пусть, как и выше, $Y = (y_1, y_2, \dots, y_n)$ — вектор выходных переменных; $y_{i,0}$ — эталонное значение выходной переменной, определенное на основе экспертных оценок; $y_{i,MM}$ — значение i -й выходной переменной, рассчитанное на модели. Тогда относительную погрешность ε_i при

расчете выходной переменной определяют следующим образом:

$$\varepsilon_i = (y_{i, \text{ММ}} - y_{i, \text{Э}}) / y_{i, \text{Э}}.$$

Погрешность модели для всех значений переменных будет представлять вектор $\varepsilon = (\varepsilon_1, \varepsilon_2, \dots, \varepsilon_n)$, где n — число выходных переменных. Отметим, что значения погрешности, полученные таким образом, зависят от свойств математической модели, особенностей решаемых задач и достоверности экспертных оценок. В этой связи при решении новых практических задач необходимо пересматривать оценки точности математической модели.

Как отмечалось выше, математическая модель — это только частичное отображение реального объекта (или системы). Поэтому степень ее универсальности соответствует полноте учета в ней свойств реальной задачи. Например, модель конструкции схемы инженерных сетей в виде гиперграфа будет отображать только ее коммутационные свойства, не учитывая протекающих в ней химических, физических и информационных процессов. Усложняя модель, т.е. увеличивая степень универсальности, можно увеличивать приближение математической модели к реальной задаче.

Экономичность математической модели обычно характеризуют затратами вычислительных ресурсов ЭВМ при их реализации. Основными из таких ресурсов являются машинное время T_M и объем используемой памяти $V_M = V_{\text{ОП}} + V_{\text{ВН}}$, где $V_{\text{ОП}}$ — объем оперативной памяти; $V_{\text{ВН}}$ — объем внешней памяти. Отметим, что так как $V_{\text{ВН}} \gg V_{\text{ОП}}$, то при анализе затрат памяти в большинстве случаев оценку можно вести по $V_{\text{ВН}}$. Очевидно, что чем меньше T_M и V_M , тем математическая модель считается экономичнее. Величину T_M определяют как усредненное число операций N , выполняемых при однократном обращении к модели. Величину V_M определяют, в основном, числом B переменных математической модели. Сравнение математических моделей по экономичности состоит в сравнении значений N и B .

Запишем алгоритм построения математической модели:

1. Определение свойств решаемой задачи, которые должны отобразиться в модели. Основным здесь является определение перечня выходных переменных $y_i \in Y$ и входных переменных $x_j \in X$.
2. Выбор структуры модели St в виде схем, «воспринимаемых» ЛПР. Установление взаимно однозначного соответствия и правил однозначного преобразования схем в математические соотношения.
3. Решение задачи идентификации, при которой определяются численные значения параметров модели $C = (c_1, c_2, \dots, c_k)$ для заданной структуры модели.
4. Выбор тестовых задач и на их основе определение погрешности. Если она больше допустимого значения $\varepsilon_{i, g}$, то осуществляется переход к 2 с выбором новой структуры математической модели; если меньше, то — переход к 5.

5. Определение значений $c_{l \min}$ и $c_{l \max}$. (Это способствует правильному выбору результатов принятия решений.)
6. Конец работы алгоритма.

Заметим, что задача идентификации в алгоритме ставится как экстремальная. Найти $\min \varepsilon_0(C)$ при $C = (c_1, c_2, \dots, c_k)$, $c_i \in C_D$, где C_D — область, в которой выбирают значения параметров математической модели; ε_0 — обобщенная оценка погрешности математической модели. Для нахождения $\min \varepsilon_0(C)$ применяют методы математического программирования, эвристического поиска, эволюционные и генетические алгоритмы.

Отметим, что для получения математической модели используются неформальные и формальные методы. Неформальные методы применяются на разных иерархических уровнях для построения элементов моделей. Они включают в себя анализ и изучение закономерностей процессов и явлений, принятие различных ограничений, допущений и упрощений. В настоящее время для этих целей начинают использоваться эвристические алгоритмы и динамические ЭС [41, 116, 159]. Построенные математические модели элементов можно применять итерационно как строительные блоки при построении иерархических математических моделей систем из этих элементов. При построении математических моделей практических задач, в основном, используются неформальные и эвристические методы на основе нечетких множеств, многоагентных, самоорганизующихся систем и нетрадиционных алгебр логики и эволюционного моделирования [17–19, 144].

Отметим, что в математической модели задачи выражение (6.1) устанавливает взаимно однозначное соответствие между входными и выходными переменными и является ее детерминированным описанием. В реальной практике принятия решений параметры задачи часто имеют стохастический и нечеткий характер. Тогда параметры модели задаются как случайные величины, выбираемые согласно заданному закону распределения вероятностей.

Приведем неравенство для параметров задачи:

$$c_i^{\text{ном}} - \delta_i \leq c_i \leq c_i^{\text{ном}} + \delta_i, \quad i = 1, 2, \dots, n,$$

где $c_i^{\text{ном}}$ — номинальное значение параметра модели c_i , которое считается его математическим ожиданием; δ_i — допуск на значение данного параметра, который определяется как половина интервала со значением $c_i^{\text{ном}}$ в центре.

Обозначим вектором $\varphi = (\varphi_1, \dots, \varphi_r)$ множество параметров состояния $\varphi \subseteq C$, которые, являясь случайными величинами, влияют на значения выходных переменных. Кроме того, обозначим вектором $\psi = (\psi_1, \psi_2, \dots, \psi_k)$ совокупность параметров состояния $\psi \subseteq C$, которые, являясь нечеткими величинами, влияют также на значения выходных переменных. При этом $\psi_i = (c_i, \mu_i)$, где μ_i — функция принадлежности элемента c_i к множеству ψ , т. е. $\mu_i : \psi \rightarrow [0, 1]$. Тогда математическое описание модели примет вид:

$$Y = F(X, Z, \varphi, \psi).$$

Нечеткий модифицированный алгоритм построения математического описания задачи можно представить в следующем виде:

1. Определение существенных с точки зрения ЛПР свойств задачи, которые необходимо отразить в ее математическом описании.
2. Разделение выбранных свойств на внешние, внутренние, неконтролируемые (случайные) и выходные переменные.
3. Выбор математической формы записи для выражения функциональных зависимостей между входными и выходными переменными.
4. Построение математического описания в виде модели.
5. Программно-аппаратная реализация математической модели, позволяющая по заданным входным переменным получать значения либо оценки выходных.
6. Проведение математического, имитационного или эволюционного моделирования для оценки численных значений параметров модели и проверки адекватности математической модели и реальной задачи.
7. Оценка точности, сложности и адекватности математического описания для обеспечения компромисса между ожидаемой точностью результатов математического моделирования и затратами вычислительных ресурсов.
8. Конец работы алгоритма.

Заметим, что для каждой задачи можно построить большое число математических описаний, которые отражают те или иные ее свойства. В процессе приятия решений при проектировании часть параметров может варьироваться в некоторых пределах, что позволяет считать их управляемыми переменными. Обычно считают все параметры управляемыми. Область пространства управляемых переменных, в которых выполняется система накладываемых на их значения ограничений, называется областью поиска D_z . Эта система ограничений имеет вид:

$$z_i^- \leq z_i \leq z_i^+, \quad i = 1, 2, \dots, r,$$

здесь z_i^- , z_i^+ — соответственно нижнее и верхнее предельно допустимое значения для i -й управляемой переменной, а r — число управляемых переменных.

В процессе принятия решений стремятся выбирать значения вектора управляемых переменных, принадлежащего области поиска $Z \in D_z$, таким образом, чтобы удовлетворить требования ЛПР. При формализации задачи удовлетворение требований сводится к выполнению системы соотношений между выходными переменными y_j и их предельно допустимыми по техническому заданию значениями y_j^- и y_j^+ , называемыми условиями работоспособности.

Область пространства управляемых переменных, в которой выполняются все условия работоспособности, называют областью работоспособности D_R . Тогда множество $D = D_z \cap D_R$ называется областью допустимых решений. Область D может оказаться выпуклым или невыпуклым множеством, которое может быть односвязным или многосвязным.

В случае невыпуклого множества D его необходимо разбивать на выпуклые подмножества $D_1, D_2, \dots, D_k$, причем $D_1 \cup D_2 \cup \dots \cup D_k = D$, а $D_1 \cap D_2 \cap \dots \cap D_k = \emptyset$. Как отмечено в [113], введение области D_R позволяет оценивать процент годных вариантов на основе определения вероятности условий работоспособности:

$$P_z = P\{y_j^- \leq y_j(\varphi, \psi) \leq y_j^+\}, \quad j = 1, 2, \dots, m.$$

Вычисление статистического показателя P_z , в котором отражается случайный φ и нечеткий ψ характер параметров, может быть определен методом Монте–Карло или генетическим алгоритмом. Область D определяет множество всех работоспособных вариантов решений задачи, каждый из которых однозначно описывается вектором управляемых переменных $z \in D$. Одна из проблем принятия решений — реализовать процедуру выбора из различных работоспособных вариантов, одного или некоторого множества, предпочтительных с точки зрения ЛПР.

Принимая предварительное решение, ЛПР дихотомически разбивает область D на два подмножества: $D = D_1 \cup D_2$. В подмножество D_1 включаются отобранные по заданному критерию варианты решения, а в D_2 — все отклоненные варианты, причем $D_1 \cap D_2 = \emptyset$, но на каждой итерации принятия решения ряд вариантов могут переходить из D_1 в D_2 и наоборот.

Модифицированный процесс построения математической модели нечетких процедур принятия решений в общих чертах следующий:

- выбор параметров в качестве управляемых переменных;
- построение области допустимых решений D ;
- задание механизмов эволюции и выбора правил генерации в области D работоспособных вариантов и определение подмножеств D_1 и D_2 .

Сравнение работоспособных вариантов при принятии решений производится на основе бинарных четких и нечетких (расплывчатых) отношений предпочтения, определенных на множестве D . Для сравнения вариантов принятия решений вводится критерий оптимальности Q — количественный показатель, характеризующий качество модели, с помощью которого осуществляется измерение одного наиболее важного для объекта свойства.

Под инженерной оптимизационной задачей понимается задача, в которой необходимо найти решение, в некотором смысле наилучшее или оптимальное. Отметим, что наилучшего решения во всех смыслах быть не может. Оно может быть принято оптимальным на основе критерия (меры оценки исследуемого явления) или ЦФ. Существует большое число оптимизационных задач и они могут иметь различный характер. Однако постановка, но не решение, всех оптимизационных задач имеет много аналогий [111–114, 159].

Во-первых, при постановке оптимизационных задач часто указывается исходное множество альтернативных вариантов решений. Из этого множества решений и выбирается оптимальное. Это исходное множество решений называется пространством решений. В дальнейшем его будем обозначать через M .

Во-вторых, некоторые решения априорно отвергаются в качестве «плохих» решений. Другими словами, в пространстве решений задаются ограничения, которым должны удовлетворять оптимальные решения. Эти ограничения выделяют в пространстве решений M некоторое подмножество M' тех решений, которые удовлетворяют заданным ограничениям D . Как отмечалось выше, это — множество допустимых решений.

В-третьих, указывается принцип сравнения любых двух допустимых решений для того, чтобы можно было выяснить, какое из них лучше в интересующем нас смысле. Как правило, этот способ сравнения задается с помощью так называемого критерия оптимальности (или функционала, функции качества, функции полезности и т. п.). Критерий оптимальности представляет собой отображение, определенное на множестве допустимых решений и имеющее в качестве значений вещественные неотрицательные числа. Обозначим это отображение, т. е. критерий, через Q . Тогда

$$Q : M' \rightarrow R^+,$$

где R^+ — множество неотрицательных вещественных чисел.

Зная функцию Q , можно реализовать процедуру сравнения вариантов решений. При этом решение $m \in M'$ лучше, чем решение $m' \in M'$, если $Q(m) < Q(m')$. В этом случае говорят, что оптимизационная задача состоит в минимизации критерия Q , т. е. требуется найти такое допустимое решение $m'' \in M'$, что

$$Q(m'') = \min Q(m'), \quad (m' \in M').$$

Итак, оптимизационную задачу запишем в виде кортежа длины три: $\langle M, D, Q \rangle$, где M — пространство решений; D — ограничения, выделяющие в M область допустимых решений $M' \subseteq M$; $Q : M' \rightarrow R^+$ — критерий оптимизации. Требование оптимизации $Q(m) \rightarrow \min$ или $Q(m) \rightarrow \max$. Решение $m'' \in M'$, удовлетворяющее требованию оптимизации, называется оптимальным [111, 114].

Оптимизационная задача удовлетворяет двум основным требованиям: должны существовать как минимум два решения; надо знать, в каком смысле искомое решение должно быть наилучшим. Выбор задачи заканчивается ее содержательной постановкой. Модель описывает зависимость между исходными данными и искомыми величинами. Математическая модель оптимизационной задачи состоит из трех составляющих: целевой функции, ограничений, граничных условий. Часто классификацию оптимизационных задач проводят по трем основным характеристикам: область применения; содержание задачи; класс математических моделей [111].

Обычно в области применения задачи делят на задачи управления, принятия решений, проектирования.

По содержанию можно выделить следующие виды задач:

- распределение ресурсов;
- оптимизация параметров объектов;
- оптимизация структуры интеллектуальных ИС;

- оптимизация процесса функционирования интеллектуальных ИС;
- проектирование интеллектуальных ИС;
- конструирование интеллектуальных ИС;
- оптимизация технологического проектирования интеллектуальных ИС и др.

Классификацию математических моделей проводят: (а) по элементам модели; (б) по искомым переменным; (с) по зависимостям, описывающим ЦФ, ограничениям и граничным условиям. При этом выделяют модели [111]. Для а: детерминированные; случайные. Для б: непрерывные; дискретные. Для с: линейные и нелинейные.

Комбинации элементов математических моделей приводят к различным классам оптимизационных задач. Наиболее простые — линейные оптимизационные задачи, однако на практике чаще всего встречаются нелинейные. При решении оптимизационных задач возможна одна из двух взаимоисключаемых постановок:

- при заданных условиях и ограничениях максимизировать получаемый результат;
- при заданном результате минимизировать используемые ресурсы.

Максимум и минимум ЦФ в оптимизационных задачах объединяют определением экстремума. Наибольшее или наименьшее значение функции без учета того, где находится такое значение — внутри заданного интервала или на его границе, называют не экстремумом, а оптимумом. Оптимум — более общее понятие, чем экстремум. Экстремум есть не у всех функций, а в оптимизационных задачах оптимум есть всегда. Оптимум может быть локальным и глобальным. Глобальным максимумом (минимумом) называют такой максимум (минимум), который больше (меньше) всех остальных. В общем случае определение глобального оптимума при решении оптимизационной задачи сводится к нахождению всех локальных оптимумов, если это возможно, и выбора из него наилучшего с точки зрения заданной ЦФ.

Оптимальное решение $x^* \in D$ называется точкой локального минимума (локальным решением), если $\forall x \in d(x^*, \varepsilon)$ истинно высказывание $Q(x^*) \leq Q(x)$, здесь $d(x^*, \varepsilon)$ — ε -окрестность точки x . Оптимальное значение $x^* \in D$ называется точкой глобального минимума (глобальным решением), если ни в одной другой точке области допустимых решений функция $Q(x)$ не принимает меньших значений: $Q(x^*) \leq Q(x) \forall x \in D$. Следовательно, глобальный минимум — это наименьший из всех локальных минимумов.

На каждом этапе принятия решений выполняется диалог с ЛПР, выбор подходящей модели, решение задачи комбинаторной оптимизации с управлением и корректированием стратегии решения. После того, как выбрана и зафиксирована конкретная оптимизационная задача, ставится вопрос о выборе той или иной модели, которая отождествляется с комбинаторной задачей дискретной оптимизации. Тогда принятие решений можно интерпретировать как процесс отбора в том или ином смысле наилучших аппроксимаций из данного класса, в основе которого лежат определенные принципы, предписывающие выбор именно тех или иных моделей из этого

класса. Различные модели решения одной и той же оптимизационной задачи принятия решений будут различаться принципами, положенными в их основу.

Предположим, что рассматривается некоторое множество исходных моделей и исследуется определенная оптимизационная задача, процесс решения которой понимается как оптимальная аппроксимация исходной задачи из некоторого базового класса. Тогда можно сказать, что на множестве исходных задач задана модель решения поставленной оптимизационной задачи, если указан некий принцип или правило, согласно которому произвольной матрице или графу ставится в соответствие некоторое подмножество альтернатив. Определение принципа выбора оптимальных альтернатив приводит к переборной задаче, формализуемой в виде комбинаторной оптимизационной задачи на графах.

6.2. Генетические алгоритмы разбиения графов

Одной из важнейших оптимизационных задач является разбиение графа на заданное или произвольное число частей [159, 160]. Задача разбиения графа на части имеет много практических применений. Она используется при проектировании устройств автоматики и вычислительной техники, создании систем управления, компьютерных и инженерных сетей, а также при решении различных задач ИИ. Отметим, что задача разбиения графа относится к классу NP-полных проблем, т. е. не существует эффективных алгоритмов ее решения с полиномиальной временной сложностью.

В полиномиальных алгоритмах временная сложность составляет $O(n)$, $O(n^2)$, $O(n^3)$, ... Для экспоненциальных алгоритмов (NP) временная сложность алгоритма составляет $O(n^n)$, $O(n^{2n})$, $O(n^{3n})$, ... Класс NP-полных задач включает такие задачи, для которых не найдены полиномиальные алгоритмы, однако и не доказано, что их не существует. Для рассмотрения вопроса о NP-полноте оптимизационной задачи их преобразуют в задачу разрешения. Обычно в качестве такой задачи рассматривают проверку, является ли некоторое число верхней или нижней границей для оптимизируемой величины. Если для оптимизационной задачи имеется быстрый алгоритм, то и полученную из нее задачу разрешения можно решать быстро. Для этого надо сравнить ответ этого алгоритма с заданной границей. Говорят, что алгоритм решает оптимизационную задачу за время $O(T(n))$, если на входных данных длины n алгоритм работает время $O(T(n))$ [161]. Отметим, что время работы алгоритма $T(n) = \Theta(n^3)$ означает, что найдутся такие константы $c_1, c_2 > 0$ и такое число n_0 , что $c_1 n^3 \leq T(n) \leq c_2 n^3$ при всех $n \geq n_0$. Эта запись включает две оценки: верхнюю и нижнюю. Следуя [161], для любых двух функций $f(n)$ и $g(n)$ свойство $f(n) = \Theta(g(n))$ выполнено, когда $f(n) = O(g(n))$. Это выражение является нижней границей, если найдется такая константа $c > 0$ и n_0 , что $0 \leq f(n) \leq c g(n)$ для всех $n \geq n_0$. Выражение $f(n) = \Omega(g(n))$ — верхняя граница, если найдутся $c > 0$ и n_0 , что $0 \leq c g(n) \leq f(n)$ для всех $n \geq n_0$. В этой связи разрабатываются различные эвристики, основанные на идеях последовательных и итерационных алгоритмов.

В [118] дана классификация существующих методов разбиения графа на части. При решении оптимизационной задачи большой размерности наибольший интерес, с точки зрения реализации, представляют последовательные (конструктивно-начальные) и итерационные эвристические методы. Временная сложность таких методов лежит в интервале $O(n) - O(n^3)$, где n — число вершин графа. В некоторых случаях для разбиения графа могут быть использованы методы случайного поиска, но они не гарантируют получения локального оптимума за конечное число шагов. В настоящее время для решения задач разбиения графов широкое применение получили методы генетического поиска [16–19, 91, 92]. Эти методы используют ЭМ и критерии типа «выживания сильнейших» для получения оптимальных решений. Опишем модифицированные ГА с управлением процессом поиска на основе синергетических принципов и адаптации для разбиения графа на подграфы. Они отличаются применением нетрадиционных архитектур генетического поиска, а также генетической оптимизации, ориентированных на использование знаний о решаемых задачах.

Сформулируем постановку задачи разбиения графа на заданное или произвольное число частей. Пусть задан граф $G = (X, E, W)$, где X представляет множество вершин графа, E — множество ребер, а W — общий суммарный вес вершин. Вес вершины соответствует интегральной оценке, в которую могут входить различные конструкторско-технологические ограничения на исследуемую модель, причем значения $w_i \in W$ не превышают некоторой пороговой величины.

Пусть $B = \{B_1, B_2, \dots, B_s\}$ — множество разбиений графа G на части $B_1, B_2, \dots, B_s$ такие, что $B_1 \cap B_2 \cap \dots \cap B_s = \emptyset$, и $B_1 \cup B_2 \cup \dots \cup B_s = B$. Тогда задача разбиения графа G на части заключается в получении разбиения $B_i \in B$, удовлетворяющего трем условиям и ограничениям:

$$(\forall B_i \in B) \quad (B \neq \emptyset),$$

$$(\forall B_i, B_j \in B)$$

$$\left([B_i \neq B_j \rightarrow X_i \cap X_j = \emptyset] \wedge [(E_i \cap E_j = E_{ij}) \vee (E_i \cap E_j = \emptyset)] \right),$$

$$\bigcup_{i=1}^s B_i = B, \quad \bigcup_{i=1}^n E_i = E, \quad \bigcup_{i=1}^n X_i = X, \quad |E_{i,j}| = K_{i,j}.$$

Целевая функция для разбиения графа G запишется так:

$$K = \frac{1}{2} \sum_{i=1}^s \sum_{j=1}^s K_{i,j}, \quad i \neq j, \quad (6.2)$$

где $K_{i,j}$ — число связей между частями B_i и B_j при разбиении графа G на части; s — количество частей в разбиении; K — суммарное количество ребер при разбиении графа на части.

Стандартная задача разбиения заключается в минимизации K ($K \rightarrow \min$). Минимизация K при разбиении графа на части позволяет косвенно

учитывать многие критерии и конструкторско-технологические ограничения исследуемой модели при решении оптимизационной задачи. Рассмотрим задачу разбиения графа G с минимизацией K . Следовательно, задача разбиения состоит в отыскании такого разбиения B_i из множества возможных разбиений B некоторого графа или гиперграфа G , при котором минимизируется (либо максимизируется) некоторая величина K , являющаяся ЦФ разбиения, и учитываются все поставленные в задаче ограничения и граничные условия, если они существуют.

Разбиение графа относится к задачам дискретной условной оптимизации из-за прерывности ее ЦФ и наличия множества ограничений на переменные. Количество методов решения задач условной оптимизации достаточно велико, хотя эффективные алгоритмы известны лишь для специальных классов задач, таких, как задачи линейного, квадратичного или выпуклого программирования; наиболее распространены для разбиения графов симплекс-метод и метод нелинейного программирования (метод штрафных и барьерных функций) [159]. Однако применительно к задаче разбиения большинство из вышеприведенных методов не может быть использовано в связи с ее дискретностью, которая приводит к существенному росту трудностей при поиске оптимума. Поэтому данные задачи выделяют в особый класс оптимизационных комбинаторных задач на графах. В задачах такого класса общее число вариантов решения равно числу перестановок из n вершин графов, т. е. $C_n = n!$, а с учетом ограничений на формирование подмножеств (частей графа в задаче разбиения) — числу сочетаний из n вершин по m частей, т. е.:

$$C_n^m = \frac{n!}{m!(n-m)!}.$$

Из вышесказанного следует, что решение задачи разбиения графа на основе полного перебора затруднительно из-за экспоненциальной сложности процесса. В этой связи разрабатываются различные эвристики решения данных задач. К ним относятся итерационные методы парных и групповых перестановок, методы последовательного приближения, релаксации, поиска в глубину и ширину, направленного перебора и др. В последнее время появились эвристики, использующие различные методы случайности. Это методы ЭМ, отжига, генетического поиска и их модификации. Все они в каком-то смысле основаны на анализе и переборе вариантов решения, однако различаются по технологии и принципам реализации.

Комбинаторные методы предназначены для поиска оптимального решения на некотором конечном множестве. В большинстве случаев они используют различные варианты сокращенного перебора, поскольку полный перебор в реальных задачах неосуществим из-за слишком большой размерности. Наиболее распространен для разбиения графов небольшой размерности метод ветвей и границ и его модификации. Эти методы основаны на идее последовательного разбиения множества на подмножества и формировании оценок, позволяющих рассматривать только те подмножества, которые могут содержать решение задачи (т. е. отбрасываются заведомо плохие ва-

рианты). Основной недостаток этих методов заключается в необходимости корректного выбора оценок, в противном случае (при неверном выборе) данный метод будет осуществлять полный перебор. Временная сложность алгоритмов данного типа лежит в пределах от $O(n^3)$ до $O(n^5)$.

В основе эвристических методов лежат нечеткие логические рассуждения. Данные методы не гарантируют нахождение оптимального решения и сильно зависят от класса и описания решаемой задачи. Эвристические методы на практике позволяют получать неплохие результаты, однако предсказать их поведение невозможно. Они могут использоваться как самостоятельно, так и совместно с другими. Это позволяет сокращать время работы. Временная сложность алгоритма в них может быть любой, но большинство известных алгоритмов имеют сложность от $O(n)$ до $O(n^2)$.

Методы динамического программирования и дискретной оптимизации при поиске новых решений используют информацию, полученную на предыдущих шагах (итерациях) применяемого алгоритма [112, 113]. Знание о получаемых решениях способствует формированию определенного поискового пространства, в котором будет производиться нахождение новых улучшенных решений. Это позволяет сократить время нахождения оптимума по сравнению с методами, реализующими полный перебор. Естественно, данные методы требуют больших ресурсов памяти для хранения промежуточных результатов и для обработки массивов данных. Временная сложность алгоритма для данного метода составляет в среднем $O(n^2)$.

Сущность большинства алгоритмов разбиения графов заключается в выборе некоторого начального разбиения исходного графа и последующего его улучшения с помощью итерационного, парного или группового обмена вершин из различных частей разбиения. При этом для каждой итерации осуществляется перестановка таких вершин, которая обеспечивает максимальное уменьшение числа связей между частями разбиения графа или минимальное приращение числа внешних ребер. Среди наиболее известных алгоритмов разбиения графа на части отметим следующие [159–166]:

- А. Мелихова и др.; В. Кернигана, С. Лина и их улучшенные эвристики;
- формирования минимальных массивов;
- моделирования отжига;
- С. Фидучео, Р. Матеусса и их модификации;
- матричные, списковые и фреймвые;
- случайных назначений;
- эволюционного моделирования.

Сравнительная оценка существующих методов показывает, что алгоритмы, использующие высокоэффективную эвристику, дают приблизительно тот же результат, затрачивая при этом на каждую генерацию значительно меньше (до 10 раз) времени. Это связано с тем, что многие эвристики используют для решения не полный перебор, а направленный на улучшение ЦФ, отбрасывая множество заведомо неперспективных решений. Таким преимуществом обладают ГА, базирующиеся на методах ЭМ. Практика показывает, что ГА, в основном, дают преимущество на больших популяциях — более 1000 вершин в задаче разбиения. При уменьшении коли-

чества элементов возрастает риск преждевременной сходимости ГА, что является прототипом локального оптимума в градиентных методах поиска решения.

Приведем краткое описание известных на сегодняшний день алгоритмов, использующих ту или иную эвристическую базу. Одними из популярных алгоритмов разбиения графа $G = (X, U)$ на части являются итерационные алгоритмы, требующие для своей работы некоторого начального разбиения [160]. Они довольно чувствительны к начальному разбиению, что, в основном, сказывается на времени работы алгоритма. Кроме того, они часто попадают в локальный оптимум, когда размер графа велик ($n > 1000$). Один из путей преодоления этого недостатка состоит в группировке сильно связанных вершин в кластеры и в дальнейшем сборе этих кластеров в заданные части разбиения, а также использование фрактальных множеств для группирования сильно связанных вершин [165]. Итерационный эвристический алгоритм начинается со случайного разбиения и затем применяет парный обмен между двумя подмножествами, улучшая начальное решение путем многократной перестановки вершин. Разработано много модификаций этого подхода для того, чтобы алгоритм смог работать с гиперграфами, уменьшив временную сложность алгоритма до $O(n)$ для графов специального вида [160, 163].

Существенным преимуществом ГА является то, что они могут работать в любом классе задач, связанных с графами. Здесь важно только правильно представить постановку задачи в генетической форме, чтобы верно отобразить поисковое пространство ЦФ. Для этого поставленная задача анализируется с точки зрения применяемых критериев оптимальности и использования того или иного ГА, что в свою очередь определяет закодированную форму представления решений задачи.

Построим схему разбиения графа на части на основе генетического поиска. Такая схема дана на рис. 6.1. Здесь в первом блоке строится текущая (начальная) популяция решений, т. е. определяется заданное подмножество $B_1, B_2, \dots, B_k$ ($k \leq s$). На построение популяции оказывает влияние внешняя среда в виде ЛПР или ЭС. Отметим, что такое подмножество решений может быть получено случайным, направленным или случайно-направленным методами.

При наличии большого количества элементов наиболее предпочтительными будут конструктивные, итерационные, случайные и поисковые алгоритмы для получения текущей популяции. В качестве ЦФ выберем K (выражение (6.2)). В некоторых исследованиях, например, в [173], предлагается при решении задачи разбиения графа на части не минимизировать K (число связей между частями разбиения), а максимизировать количество связей (критерий A) внутри частей разбиения. Известно, что критерии K и A являются взаимобратными, т. е. минимизация K максимизирует A и наоборот. Для каждого элемента текущей популяции в блоке 2 вычисляется K и определяется $K_{\text{ср}}$ — среднее значение ЦФ на данной популяции (рис. 6.1). Блок 3 осуществляет сортировку популяции на основе ЦФ.

множества альтернативных решений, причем в такой схеме возможно концентрировать поиск на получение более перспективных решений. Отметим, что периодически в каждой итерации ГА можно проводить различные изменения в перспективных, неперспективных и в других решениях.

Рассмотрим последовательный ГА разбиения. Пусть задан граф $G = (X, U)$. Первоначально упорядочим все вершины графа по возрастанию локальных степеней вершин. Это соответствует списку вершин, а также может соответствовать тривиальному разбиению, когда количество групп разбиения равно упорядоченным вершинам графа. Далее начинаем составлять пары вершин и для каждой пары вычислять оценку связности:

$$\delta_{i,j} = e_{i,j} - (e_{i,t} + e_{j,t}), \quad (6.3)$$

где $e_{i,j}$ — число ребер между вершинами x_i и x_j , образовавшими пару; $e_{i,t}$ и $e_{j,t}$ — это число ребер, соединяющих выбранную пару со всеми остальными вершинами графа, причем $t = 1, 2, \dots, n - 2$.

Составим второй список, где будут оставлены такие пары вершин, у которых $\delta_{i,j} \geq 0$. Отметим, что если пар $\delta_{i,j} \geq 0$ не находится, то возможны два случая. В первом образуются пары с наименее отрицательным $\delta_{i,j}$. Во втором случае пары из двух вершин не образуются и сразу происходит переход к образованию групп из трех вершин. Далее процесс повторяется аналогично, пока не будет выполнено разбиение графа на заданное или произвольное число частей. Это — основная идея стандартной процедуры разбиения. При небольшом количестве вершин она дает удовлетворительные результаты, так как практически здесь осуществляется полный перебор вариантов решений. С увеличением числа вершин использование напрямую такого подхода становится нецелесообразно. Для сокращения числа просматриваемых пар применим простые и модифицированные генетические операторы. Отметим, что вместо выражения (6.3) можно использовать выражение

$$\tau_i + \tau_j \geq \tau_t, \quad (6.4)$$

где $\tau_i + \tau_j$ — внутренние ребра разбиения, τ_t ($t = 1, 2, \dots, n - 2$) — внешние ребра разбиения.

Рассмотрим пример. Пусть задан граф (рис. 6.2). Определим локальные степени вершин и упорядочим их в виде списка по возрастанию (табл. 6.1). Построим теперь пары вершин для определения разбиения графа, по две, вершины в каждой части.

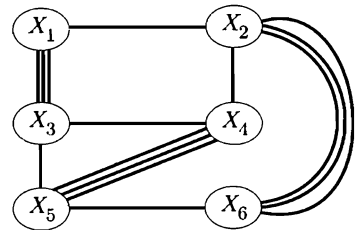


Рис. 6.2. Граф G

Таблица 6.1

x_i	x_2	x_3	x_4	x_5	x_6	x_1
ρ_{x_i}	5	5	5	5	4	4

На этом уровне выделим два блока разбиения: это вершины (x_1, x_3) , (x_2, x_6) . Для удобства записи вместо $(x_1, x_2, \dots)$ будем иногда записывать

их соответствующие номера (1, 2, ...). Если стоит задача разбить граф на части по две вершины в каждой с минимизацией K , то в качестве третьего блока разбиения можно взять блок с наименьшим отрицательным значением выражения (6.4). Такое разбиение показано на рис. 6.3.

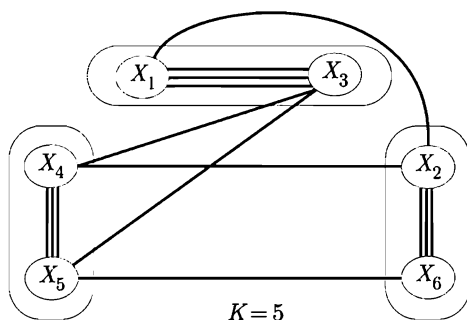


Рис. 6.3. Разбиение графа G на три части

В табл. 6.2 приведены все пары, которые можно получить, объединяя все вершины заданного графа. Очевидно, что для большого числа вершин данная процедура нецелесообразна. Применим одну из множества модифицированных схем генетического поиска, описанных в предыдущих разделах.

Таблица 6.2

$(x_i x_j)$	(1,2)	(1,3)	(1,4)	(1,5)	(1,6)	(2,3)	(2,4)	(2,5)
$\tau_i + \tau_j \geq \tau_t$	$1 \geq 7$	$3 \geq 3$	$0 \geq 9$	$0 \geq 9$	$0 \geq 8$	$0 \geq 10$	$1 \geq 8$	$0 \geq 10$

$(x_i x_j)$	(2,6)	(3,4)	(3,5)	(3,6)	(4,5)	(4,6)	(5,6)
$\tau_i + \tau_j \geq \tau_t$	$3 \geq 3$	$1 \geq 8$	$1 \geq 8$	$0 \geq 9$	$3 \geq 4$	$0 \geq 9$	$1 \geq 7$

Построим случайным образом некоторую популяцию решений $P = \{p_1, p_2, p_3, p_4, p_5, p_6\}$, $p_1 : (1, 2)$. Теперь проверим выполнение условия (6.4) для всех элементов популяции P . Для первого элемента — $1 \geq 7$ (условие не выполняется). Продолжая далее, получим:

$$p_1: (1,2) \sim 1 \geq 7, \quad p_3: (3,4) \sim 1 \geq 8, \quad p_5: (5,6) \sim 1 \geq 7,$$

$$p_2: (2,3) \sim 0 \geq 10, \quad p_4: (4,5) \sim 3 \geq 4, \quad p_6: (3,6) \sim 0 \geq 9.$$

В качестве ЦФ взято выражение (6.4). Определим для каждой хромосомы в популяции значение ЦФ. Произведем сортировку анализируемой популяции и получим:

$$p_4, \quad p_5, \quad p_1, \quad p_3, \quad p_6, \quad p_2.$$

Произведем селекцию и образуем родительские пары:

$$p_4 : (4, 5), \quad p_1 : (1, 2), \quad p_6 : (3, 6)$$

$$p_5 : (5, 6), \quad p_3 : (3, 4), \quad p_2 : (2, 3).$$

Применим стандартный ОК и получим новые хромосомы:

$p_4:(4,5)$	$p_1:(1,2)$	$p_6:(3,6)$
$p_5:(5,6)$	$p_3:(3,4)$	$p_2:(2,3)$
$p'_1:(4,6) - 0 \geq 9$	$p'_1:(1,4) - 0 \geq 9$	$p'_6:(3,3)$
$p'_5:(5,5)$	$p'_3:(3,2) - 0 \geq 10$	$p'_2:(2,6) - 3 \geq 3$ (да).

В результате однократного применения ОК мы получили лучшее решение $p'_2:(2,6)$. Тенденции преимущества ГА видны даже на данном простом примере.

Использование стандартного ОК может приводить к несуществующим решениям (разбиения p'_5, p'_6). В этом случае можно предложить несколько модификаций ОК. В частности, при скрещивании пар потомки могут получаться путем объединения первых и вторых элементов родителей с анализом полученных решений. При этом повторяющиеся элементы удаляются, а отсутствующие добавляются. Тогда в нашем примере получим:

$$p''_4:(4,5), p''_1:(1,3), p''_6:(3,2).$$

$$p''_5:(5,6), p''_3:(2,4), p''_2:(6,3).$$

Здесь, кроме потомков, повторяющих родителей (p''_1, p''_5, p''_6), находится элемент p''_1 с одним из лучших значений целевой функции ($3 \geq 3$). Продолжая аналогично, построим группы разбиения по три вершины.

На рис. 6.4 показано разбиение графа G (рис. 6.2) на две части по три вершины в каждой. При этом значение целевой функции $K = 5$. Отметим, что генетическая процедура поиска может быть применена как единственная, так и множественная оптимизационная процедура. Кроме того, генетические операторы и операторы поиска могут быть вставлены в структурную схему генетического поиска как вспомогательные блоки для повышения качества последовательных алгоритмов.

В первом случае сложность алгоритма разбиения графа будет лежать в пределах $O(n) \div O(n^3)$, причем крайний случай с $O(n^3)$ будет иметь место при популяции больше ста и выше тысячи генераций алгоритма.

Фrakталами, как отмечалось выше, обычно называют множества, которые обладают масштабной инвариантностью, т.е. в любом масштабе они выглядят практически одинаково. Алгоритмы решения большинства нелинейных оптимизационных задач могут использовать идеи фрактальных множеств.

Рассмотрим алгоритм разбиения графов на части на основе модифицированной агрегации фракталов [165]. Процесс создания кластеров основан на идеях построения минимальных и квазiminимальных массивов в графе

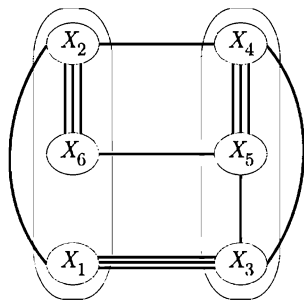


Рис. 6.4. Разбиение графа G на две части

или гиперграфе. Кластер — это часть графа $\Phi \subseteq G$, причем $\Phi = (X_1, U_1)$, $X_1 \subseteq X$, $U_1 \subseteq U$. Вершины кластера соединены внутренними ребрами с остальными вершинами $X \setminus X_1$ графа G . Внешние ребра не входят в подмножество ребер кластеров. Мощность подмножества внешних ребер кластера обозначим f . Кластер Φ_i называется минимальным, если для любого другого кластера $\Phi_j \subset \Phi_i$ выполняется условие $f_i \leq f_j$. Другими словами, удаление произвольных вершин из Φ_i ($\Phi_i \setminus X_m$) приводит к новому кластеру с большим числом внешних ребер. По определению будем считать:

$$(\forall \Phi_i \subset G) (\Phi_i \neq \emptyset), \\ (\forall x_i \in X) (x_i \text{ — минимальный кластер}).$$

Это означает, что минимальный кластер не может быть пустым. Кроме того, в тривиальном частном случае каждая вершина графа образует минимальный кластер. Кластер будем называть квазiminимальным, если выполняется условие:

$$f_i + \varepsilon \leq f_j,$$

где ε — коэффициент, определяющий, на какую величину можно увеличить число внешних ребер в минимальном кластере Φ_i . Он определяется в процессе анализа математических моделей графа, на основе решения ЛПР или ЭС.

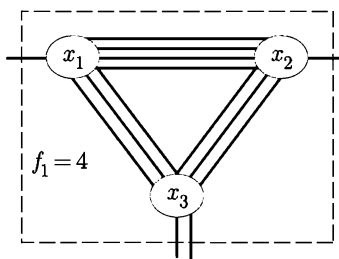


Рис. 6.5. а. Минимальный кластер на три вершины

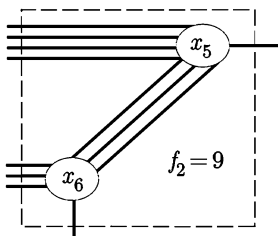


Рис. 6.5. б. Кластер на две вершины

На рис. 6.5, а показан минимальный кластер на три вершины с $f_1 = 4$. Соответственно на рис. 6.5, б показан простой кластер с $f_2 = 9$. На рис. 6.6, а показан минимальный кластер на две вершины с $f_3 = 3$. При присоединении к нему вершины x_8 получим квазiminимальный кластер на три вершины с $f_4 = 5$ (рис. 6.6, б). В этом случае величина $\varepsilon = 1$.

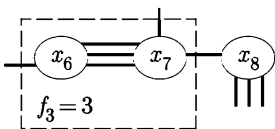


Рис. 6.6. а. Минимальный кластер на две вершины

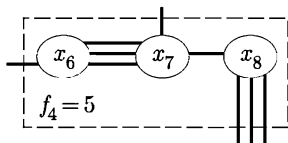


Рис. 6.6. б. Квазiminимальный кластер на три вершины

Построение кластера связано с выделением почти всех подмножеств множества X , а оно составляет 2^n , если $|X| = n$. В этой связи продолжают разрабатывать различные эвристики выделения набора простых кластеров. Эти эвристики основаны на теоремах о минимальных массивах в графах [165] и принципах построения квазимиимального кластера.

Теорема 6.1. Если Φ_i и Φ_j являются минимальными кластерами и $\Phi_i \not\subseteq \Phi_j \wedge \Phi_j \not\subseteq \Phi_i$, то $\Phi_i \cap \Phi_j = \emptyset$.

Теорема 6.2. Пусть Φ_i, Φ_j — минимальные кластеры. Если $\Phi_s = \Phi_i \cup \Phi_j \wedge f_s < f_i \wedge f_s < f_j$, то Φ_s — минимальный кластер. Если $\Phi_s = \Phi_i \cup \Phi_j \wedge f_s + \varepsilon \leq f_i \wedge f_s \leq f_j$, то Φ_s — квазимиимальный кластер.

Теорема 6.3. Пусть $\Phi_1, \Phi_2, \dots, \Phi_L$ — минимальные кластеры, причем любое объединение части из них не является минимальным кластером. Если $\Phi_i = \Phi_1 \cup \Phi_2 \cup \dots \cup \Phi_L \wedge f_i < f_1 \wedge f_i < f_2, \dots, \wedge f_i < f_L$ то Φ_i — минимальный кластер. Если $\Phi_i = \Phi_1 \cup \Phi_2 \cup \dots \cup \Phi_L \wedge f_i + \varepsilon \leq f_2, \dots, \wedge f_i + \varepsilon \leq f_L$ то Φ_i — квазимиимальный кластер.

Доказательства теорем следуют из описанных способов построения кластеров.

Из теоремы 6.1 следует, что в общем случае при полном переборе процедура построения минимального кластера сходящаяся и решение единственно при разбиении графа на части, когда не задано общее число вершин в частях разбиения. При разбиении графа на заданное число вершин количество решений в общем случае — это мощность некоторого подмножества альтернатив разбиений графа. На основе теорем 6.2 и 6.3 возможно агрегировать (объединять) кластеры и строить минимальные и квазимиимальные кластеры. На основе теорем 6.1, 6.2, 6.3 опишем модифицированный эвристический нечеткий алгоритм построения минимальных и квазимиимальных кластеров в графе.

1. Упорядочим все локальные степени вершин графа как тривиальные минимальные и квазимиимальные кластеры.
2. Начиная с вершин с большей локальной степенью, попарно анализируем все вершины графа G . Если определяется пара вершин (x_i, x_j) , для которой

$$f = \rho(x_i) + \rho(x_j) - 2r_{i,j}$$

и $\Phi = (X_1, U_1)$, $X_1 = \{x_i, x_j\}$, то Φ является минимальным кластером. Здесь $\rho(x_i), \rho(x_j)$ — локальные степени вершин x_i, x_j ; $r_{i,j}$ — число ребер, соединяющих вершины x_i, x_j между собой, f — число ребер, соединяющих минимальный кластер с остальными вершинами графа G , причем $f < \rho(x_i), f < \rho(x_j)$. Если определяется пара вершин (x_i, x_j) , для которой

$$f = \rho(x_i) + \rho(x_j) - 2r_{i,j} + \varepsilon,$$

причем $f + \varepsilon \leq \rho(x_i), f + \varepsilon \leq \rho(x_j)$, то $\Phi = (X_1, U_1)$, $X_1 = \{x_i, x_j\}$ и Φ является квазимиимальным кластером. Минимальные или квазимиимальные кластеры Φ заносим в специальный список, а вершины

x_i, x_j исключаем из рассмотрения. Переход к 2. Если новых минимальных или квазiminимальных кластеров не образовалось, то переход к 3.

3. Выполняем факторизацию. Вершины x_i, x_j в минимальных или квазiminимальных кластерах заменяем одной $x_{i,j}$. При этом ребра, соединяющие x_i и x_j , удаляются, а ребра из вершины x_i, x_j к другим вершинам приписываются общей вершине $x_{i,j}$. Получается граф G' .
4. В графе G' анализируются все вершины. Если определится тройка вершин x_a, x_b, x_c , для которой справедливо:

$$f = \rho(x_a) + \rho(x_b) + \rho(x_c) - 2r_{a,b} - 2r_{b,c} - 2r_{a,c},$$

причем $f < \rho(x_a)$, $f < \rho(x_b)$, $f < \rho(x_c)$, то объединенная вершина $\Phi = (x_a, x_b, x_c)$ образует минимальный кластер согласно теореме 6.3. Если определится тройка вершин x_a, x_b, x_c , для которой справедливо:

$$f = \rho(x_a) + \rho(x_b) + \rho(x_c) - 2r_{a,b} - 2r_{b,c} - 2r_{a,c} + \varepsilon,$$

причем $f + \varepsilon \leq \rho(x_a)$, $f + \varepsilon \leq \rho(x_b)$, $f + \varepsilon \leq \rho(x_c)$, то вершины x_a, x_b, x_c образуют квазiminимальный кластер согласно теореме 6.3. Переход к 2. Минимальный или квазiminимальный кластер заносится в список. Если в минимальных или квазiminимальных кластерах больше нет, то переход к 4.

5. Увеличим параметр мощности минимального или квазiminимального кластера на единицу и переходим к 3. Если мощность минимального или квазiminимального кластера равна заданному или равна числу вершин графа, то переход к 6.
6. Конец работы алгоритма.

Здесь ε — наименьшее относительное отклонение от минимального кластера ($\varepsilon = 0, 1, 2, \dots$). После построения набора минимальных или квазiminимальных кластеров, если в графе G остались свободные вершины, необходимо провести процедуру агрегации. Она заключается в пробном помещении вершин в минимальный или квазiminимальный кластер с анализом ЦФ. Вершина помещается в минимальный кластер, если нарушение происходит на величину, не превышающую ε . Если после второй процедуры остались свободные вершины, реализуются процедуры, основанные на моделировании эволюций Ч. Дарвина, де Фризе, Ж. Ламарка и К. Поппера, описанные выше. Предпочтение отдается модифицированным операторам мутации, инверсии, сегрегации, удаления и вставки. Рассмотрим пример, показанный на рис. 6.7. Приведен граф $G = (X, U)$, $|X| = 14$, $|U| = 36$, который должен быть разбит на части с наименьшим значением K (6.2).

Отметим, что число частей разбиения заранее не задано и определяется в процессе реализации алгоритма после каждого шага построения минимальных и квазiminимальных кластеров. Матрица смежности R графа G

ИМЕЕТ ВИД:

	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	x_9	x_{10}	x_{11}	x_{12}	x_{13}	x_{14}
x_1	0	0	0	1	1	2	2	0	0	0	0	1	0	0
x_2	0	0	1	0	0	0	0	0	0	0	0	0	0	4
x_3	0	1	0	0	0	0	0	0	2	0	2	1	0	0
x_4	1	0	0	0	2	2	0	0	0	0	0	0	0	0
x_5	1	0	0	2	0	1	2	0	0	0	0	0	0	0
x_6	2	0	0	2	1	0	0	0	0	0	0	0	0	0
x_7	2	0	0	0	2	0	0	0	0	0	0	0	0	0
x_8	0	0	0	0	0	0	0	0	0	2	0	0	2	1
x_9	0	0	2	0	0	0	0	0	0	0	1	2	1	0
x_{10}	0	0	0	0	0	0	0	2	0	0	0	0	2	0
x_{11}	0	0	2	0	0	0	0	0	1	0	0	2	0	0
x_{12}	1	0	1	0	0	0	0	0	2	0	2	0	0	0
x_{13}	0	0	0	0	0	0	0	2	1	2	0	0	0	0
x_{14}	0	4	0	0	0	0	0	1	0	0	0	0	0	0

По матрице определим локальную степень каждой вершины и построим упорядоченный по возрастанию список степеней этих вершин. Сначала определяется минимальный кластер $\Phi_1 = \{x_2, x_{14}\}$ при этом $K_1 = 2$, $|U_1| = 4$, $|U_1| - K_1 = 2 > 0$. Производится факторизация. Строки и столбцы x_2, x_{14} матрицы R представляются объединенной строкой и столбцом (x_2, x_{14}) , т.е. вершины x_2, x_{14} с соединяющими их ребрами заменяются вершиной Φ_1 и продолжается поиск минимального или квазiminимального кластеров, состоящих из двух вершин. Таких кластеров больше нет. Переходим к поиску указанных кластеров, состоящих из трех вершин. Согласно алгоритму получим: $\Phi_2 = \{x_8, x_{10}, x_{13}\}$, при этом $K_2 = 1$, $|U_2| = 6$, $|U_2| - K_2 = 5 > 0$. При определении K_2 не учитываются связи Φ_1 с Φ_2 , чтобы не определять одни и те же внешние ребра дважды. Продолжая реализацию алгоритма, аналогично получим: $\Phi_3 = \{x_3, x_9, x_{11}, x_{12}\}$, $K_3 = 1$,

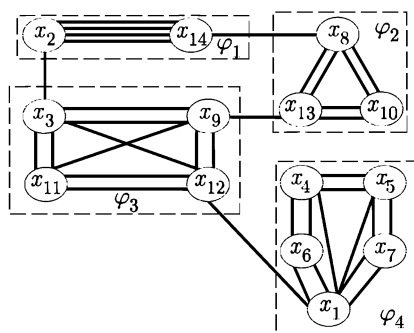


Рис. 6.7. Граф G

$|U_3| = 10$, $|U_3| - K_3 = 9 > 0$ и $\Phi_4 = \{x_4, x_5, x_6, x_7, x_1\}$, $K_4 = 0$, $|U_4| = 12$, $|U_4| - K_4 = 12 > 0$. Общее число внешних связей между частями разбиения графа G (рис. 6.7) равно $K = K_1 + K_2 + K_3 + K_4 = 2 + 1 + 1 = 4$.

Рассмотрим другой пример, показанный на рис. 6.8.

Матрица смежности графа G имеет вид

$$R = \begin{array}{c|cccccccccccc} & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & 11 & 12 \\ \hline 1 & 0 & 3 & 2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 2 & 3 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 3 & 2 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 4 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 1 \\ 5 & 0 & 0 & 1 & 0 & 0 & 5 & 0 & 0 & 0 & 2 & 0 & 0 \\ 6 & 0 & 0 & 0 & 1 & 5 & 0 & 5 & 1 & 0 & 0 & 0 & 0 \\ 7 & 0 & 0 & 0 & 1 & 0 & 5 & 0 & 0 & 1 & 0 & 1 & 0 \\ 8 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 2 & 0 & 0 & 0 \\ 9 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 2 & 0 & 0 & 0 & 0 \\ 10 & 0 & 0 & 0 & 0 & 2 & 0 & 0 & 0 & 0 & 0 & 3 & 0 \\ 11 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 3 & 0 & 3 \\ 12 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 3 & 0 & 0 \end{array}.$$

По матрице определим локальную степень каждой вершины и построим упорядоченный по возрастанию список степеней этих вершин. В результате выполнения алгоритма определим минимальный кластер: $\Phi_1 = \{x_1, x_2\}$, $\Phi_2 = \{x_8, x_9\}$. Представив Φ_1 и Φ_2 в виде вершин, получим новый граф, матрица смежности которого примет вид

$$R_1 = \begin{array}{c|cccccccc} & (1,2) & 3 & 4 & 5 & 6 & 7 & (8,9) & 10 & 11 & 12 \\ \hline (1,2) & 0 & 3 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 3 & 3 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 4 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 1 \\ 5 & 0 & 1 & 0 & 0 & 5 & 0 & 0 & 2 & 0 & 0 \\ 6 & 0 & 0 & 1 & 5 & 0 & 5 & 1 & 0 & 0 & 0 \\ 7 & 0 & 0 & 1 & 0 & 5 & 0 & 1 & 0 & 1 & 0 \\ (8,9) & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 10 & 0 & 0 & 0 & 2 & 0 & 0 & 0 & 0 & 3 & 0 \\ 11 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 3 & 0 & 3 \\ 12 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 3 & 0 \end{array}.$$

Так как минимальных кластеров мощности 2 больше нет, то построим согласно алгоритму минимальный кластер мощности 3. Получим $\Phi_3 = \{x_1, x_3\}$, $\Phi_4 = \{x_5, x_6, x_7\}$, $\Phi_5 = \{x_{10}, x_{11}, x_{12}\}$. Работа алгоритма закончена, больше минимальных кластеров нет. Вершину x_4 , которая, вообще говоря, есть тривиальный минимальный кластер, можно случайным образом добавлять во все остальные минимальные кластеры с вычислением ЦФ. В результате получим два варианта разбиения:

1-й вариант: $\{x_1, x_2, x_3, x_4\}$, $\{x_5, x_6, x_7\}$, $\{x_8, x_9\}$, $\{x_{10}, x_{11}, x_{12}\}$. При этом $K = 9$.

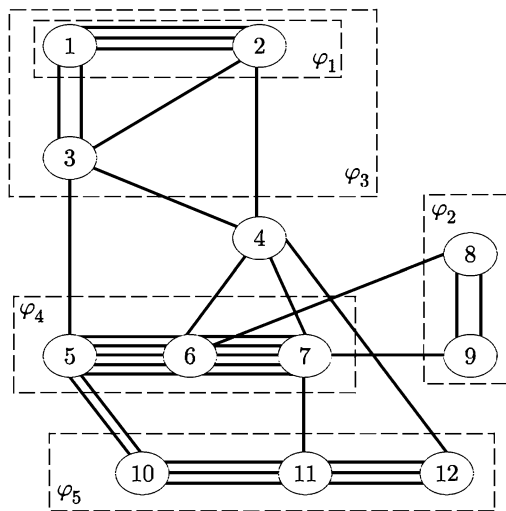


Рис. 6.8. Граф G

2-й вариант: $\{x_1, x_2, x_3\}$, $\{x_4, x_5, x_6, x_7\}$, $\{x_8, x_9\}$, $\{x_{10}, x_{11}, x_{12}\}$. При этом $K = 9$.

Например, если задано разбиение графа на три части, вариант 1 после процедуры эволюции запишется так:

3-й вариант: $\{x_1, x_2, x_3, x_4\}$, $\{x_5, x_6, x_7, x_8, x_9\}$, $\{x_{10}, x_{11}, x_{12}\}$. В этом случае $K = 7$.

Рассмотрим основную идею итерационного разбиения гиперграфа $H = (X, E)$ на части с минимизацией K . Для каждого ребра гиперграфа $e_i \in E$ можно определить разрезающую функцию, определяющую число вершин дерева, моделирующего это ребро, попадающих в различные блоки. Если хотя бы одна вершина ребра гиперграфа лежит не в той части, что остальные вершины этого ребра, то обязательно будет существовать, по крайней мере, одно разрезающее ребро. Просуммировав все разрезающие ребра, получим общее число K (связей) между блоками, которое необходимо минимизировать.

Предлагаемый алгоритм состоит из трех этапов. На первом этапе строятся популяции решений, которые могут быть получены случайным, последовательным или итерационным методами. На втором этапе для каждой хромосомы в популяции определяется ЦФ. В качестве ЦФ выбирается K . Затем полученная популяция сортируется и упорядочивается согласно ЦФ. Далее выполняется разбиение популяции на две подпопуляции: перспективных и неперспективных хромосом. Для первой подпопуляции используется точный поиск, т. е. применяются различные архитектуры ГА, описанные выше. Для второй подпопуляции используется быстрый поиск — ПГА. Затем реализуется оператор миграции хромосом между подпопуляциями. На третьем этапе, используя описанные схемы селекции, выбираются родительские пары или отдельные хромосомы для выполнения генетических операторов.

Перед выполнением генетических операторов в каждой хромосоме (текущий вариант разбиения) определяется стоимость существующих строительных блоков. Работа генетических операторов начинается с модифицированного ОК. При этом точка или точки ОК определяются не случайно, а направленно, выделяя лучшие строительные блоки в каждой хромосоме.

Например, на рис. 6.9 показана схема модифицированного ОК. Здесь p_1 , p_2 — хромосомы из популяции P . Строительные блоки, т. е. части разбиения, следующие: $\{a, b, c\}$, $\{d, e\}$, $\{f, g\}$; $\{d, e, c\}$, $\{a, f\}$, $\{b, g\}$. Определяя ЦФ, найдем, что K_2 и K_5 — наименьшие. Тогда точки ОК в p_1 направленным образом выделяют блок $\{d, e\}$, а в p_2 — блок $\{a, f\}$. Затем производится копирование лучших строительных блоков из p_1 и p_2 в p'_1 . Свободные места в p'_1 заполняются оставшимися элементами из p_1 и p_2 .

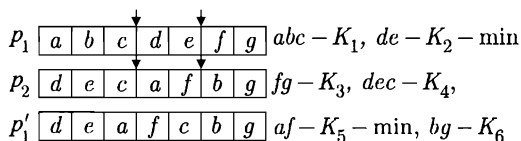


Рис. 6.9. Схема модифицированного оператора кроссинговера

Возможны случаи, когда выбранные строительные блоки могут иметь частичное или полное совпадение элементов. При полном совпадении элементов производится копирование в потомка только одного из строительных

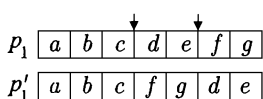


Рис. 6.10. Пример оператора мутации

причем элементы (члены) можно переставлять только между строительными блоками. При направленной мутации в каждом строительном блоке выбирается наихудший элемент.

Для выхода из локальных оптимумов предлагаются две процедуры. В основе первой лежит модифицированный оператор сегрегации. На рис. 6.11 показан пример данного оператора. Заметим, что строительные блоки из хромосом копируются полностью, если они не пересекаются с уже выделенными.

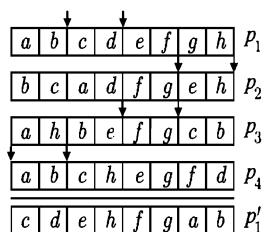


Рис. 6.11. Пример оператора сегрегации

В противном случае копируются только непересекающиеся элементы из строительного блока.

Важную роль в генетических операторах, как отмечалось выше, играет вероятность кроссинговера $P(OK)$ и вероятность мутации $P(OM)$. Эксперименты показали, что в задачах разбиения $P(OK)$, лежащая в пределах $0,5 \div 0,98$, и $P(OM)$ в пределах $0,01 \div 0,05$, позволяют получать большее число локальных оптимумов за $1000 \div 10000$ генераций [165].

Для выхода из локальных оптимумов предлагается ряд процедур. В основе первой лежат модифицированные операторы сегрегации, транслокации, удаления и вставки. В основе второй процедуры лежат идеи миграции хромосом внутри одной популяции и между популяциями. В зависимости от наличия вычислительных ресурсов можно первоначально строить не одну, а некоторый набор популяций. Далее выбирать «лучшего» представителя из каждой популяции и на этом множестве осуществлять ЭМ.

Определим асимптотическую трудоемкость данного алгоритма. Пусть N_p — размер популяции, N_n — число полученных потомков, N_G — число поколений ГА. Тогда общую трудоемкость алгоритма можно ориентировочно определить:

$$T \approx [\alpha N_p t_p + \beta N_n t_n + \gamma (N_p + N_n) t_c] N_G,$$

где t_p — трудоемкость построения одной хромосомы в популяции; t_n — трудоемкость генерации одного потомка; t_c — совокупная трудоемкость селекции и отбора, α, β, γ — коэффициенты, определяющие параметры генетического поиска. Заметим, что трудоемкость построения одной хромосомы может колебаться от $O(n)$ до $O(n!)$, а трудоемкость генерации одного потомка зависит от сложности применяемого генетического оператора и ориентировочно изменяется от $O(n)$ до $O(n \log n)$. Трудоемкость селекции и отбора может меняться от $O(n)$ до $O(n^2)$. Временная сложность описанного алгоритма ориентировочно составляет $O(n^2)$ для одной генерации.

Следует заметить, что с увеличением числа поколений в ГА время решения оптимизационной задачи разбиения повышается, но это повышение незначительное и может быть компенсировано получением множества локально-оптимальных решений. Генетические алгоритмы требуют больших затрат времени, чем алгоритмы парных перестановок и случайного разбиения графов. Но они позволяют получать набор локально-оптимальных решений.

Допустимое решение экстремальной задачи разбиения графа на части является n -мерным вектором. В том случае, когда задача принадлежит классу задач переборного типа, имеется конечное множество допустимых решений, в которых каждая компонента вектора может быть закодирована с помощью целого неотрицательного числа: $\beta_i \in [0, V_i]$, $i = \overline{1, n}$, где $(V_i + 1)$ — число возможных дискретных значений i -управляемой переменной в области поиска D [92]. Это позволяет поставить во взаимнооднозначное соответствие каждому вектору $p_i \in D$ вектор β с целочисленными компонентами. Согласно [92], символьная модель экстремальной задачи разбиения графа на части может быть представлена в виде множества бинарных строк, которые описывают конечное множество допустимых решений, принадлежащих области поиска D . Необходимо отметить, что выбор символьной модели исходной экстремальной задачи разбиения графа на части во многом определяет эффективность и качество применяемых алгоритмов.

Представим, например, разбиение графа $G = (X, U)$ на две части в виде бинарной строки $E(X_1, X_2)$, состоящей из n бит, расположенных в порядке возрастания их номеров. Каждому номеру бита поставим во взаимно-однозначное соответствие номер вершины графа (1-й бит соответствует вершине x_1 , 2-й бит — вершине $x_2, \dots, n$ -й бит — вершине x_n). Потребуем, чтобы бинарное значение α_l l -го бита указывало, какому подмножеству вершин (X_1 или X_2) принадлежит вершина x_l :

$$\alpha_l = \begin{cases} 1, & \text{если } l\text{-я вершина } x_l \in X \text{ входит в состав} \\ & \text{подмножества вершин } X_1; \\ 0, & \text{если } l\text{-я вершина } x_l \in X \text{ входит в состав} \\ & \text{подмножества вершин } X_2. \end{cases}$$

Число битов, содержащих «1» в бинарной строке $E(X_1, X_2)$, должно равняться мощности подмножества вершин подграфа $G_1 = (X_1, U_1)$, равной порядку этого подграфа n_1 . Так, для графа G (рис. 6.4) получим следующую кодировку разбиения в виде бинарных строк:

$$E(X_1, X_2): \begin{array}{|c|c|c|c|c|c|} \hline 1 & 1 & 0 & 0 & 0 & 1 \\ \hline \end{array}$$

1 2 3 4 5 6 — номер бита

x_1 x_2 x_3 x_4 x_5 x_6 — номер вершины

В задаче оптимального разбиения графа на части в качестве хромосомы выступает конкретное разбиение, удовлетворяющее заданным условиям, что позволяет интерпретировать решения задачи разбиения как эволюционный процесс, связанный с перераспределением вершин $x_i \in X$ графа G по частям разбиения.

В [92] для описания популяций введено два типа переменных признака, отражающих качественные и количественные различия между хромосомами. Качественные признаки — признаки, которые позволяют однозначно разделять совокупность хромосом на четко различимые группы. Количественные признаки — признаки, проявляющие непрерывную изменчивость, в связи с чем степень их описания задается числом.

В работе [92] предлагается генетический алгоритм дихотомического разбиения графа $G = (X, U)$ на два подграфа, основанный на работе [159]: $G_1 = (X_1, U_1)$ и $G_2 = (X_2, U_2)$, $X_1 \cup X_2 = X$, $U_1 \cup U_2 \cup U_3 = U$, где U_3 — подмножество ребер, соединяющих два подграфа, причем для любого ребра $U_i \in U_3$, $U_i = (x_i, x_j)$, $x_i \in X_1$ и $x_j \in X_2$ или $x_i \in X_2$ и $x_j \in X_1$. Целевая функция или критерий разбиения — это число ребер, соединяющих подграфы ($K = |U_3|$). Тогда оптимально дихотомическое разбиение является решением (X_1^*, X_2^*) следующей экстремальной задачи однокритериального вектора

$$Q^* = Q(X_1^*, X_2^*) = \min K, \quad X_1, X_2 \subset X,$$

причем $K = |U \setminus (U_1 \cup U_2)|$, где U_1, U_2 — подмножества ребер, соединяющих только вершины подмножеств X_1 и X_2 между собой.

Здесь столбцы матрицы R соответствуют заданным элементам графа, а строки — свойствам или выполненным функциям этих элементов. Например, единица на пересечении A_4 строки и 4 столбца означает, что элемент 4 обладает свойствами A_4 . Необходимо сгруппировать элементы, обладающие наибольшим числом одинаковых свойств. Данная задача относится к классу задач разбиения и может быть решена одним из эвристических алгоритмов.

Приведем основную идею предлагаемого эвристического алгоритма [16, 163]. Предварительно выполняется анализ матричной модели. В результате анализа предлагается удалить из рассмотрения строки матрицы, содержащие все единицы, все нули, а также строки, содержащие около 1% нулей или единиц. Такое удаление не будет влиять на создание требуемых групп, так как все или почти все элементы будут входить или не входить в формируемые группы. Тогда в рассматриваемом примере исключим строки A_5 и A_8 . Далее произведем сортировку матрицы R . При этом на первое место поместим столбец с наибольшим числом единиц и так далее по убыванию. В этом случае матрица R запишется:

$$R = \begin{matrix} & \begin{matrix} 12 & 4 & 7 & | & 8 & 9 & 11 & | & 1 & 5 & 6 & | & 2 & 3 & 10 \end{matrix} \\ \begin{matrix} A_1 \\ A_2 \\ A_3 \\ A_4 \\ A_6 \\ A_7 \end{matrix} & \begin{pmatrix} 1 & 0 & 1 & | & 1 & 1 & 1 & | & 1 & 1 & 0 & | & 0 & 0 & 0 \\ 1 & 1 & 1 & | & 1 & 0 & 1 & | & 0 & 0 & 0 & | & 0 & 1 & 0 \\ 1 & 0 & 1 & | & 0 & 1 & 1 & | & 1 & 1 & 0 & | & 0 & 1 & 0 \\ 1 & 1 & 0 & | & 1 & 0 & 0 & | & 0 & 0 & 1 & | & 1 & 0 & 1 \\ 0 & 1 & 0 & | & 1 & 1 & 0 & | & 0 & 1 & 1 & | & 0 & 0 & 1 \\ 1 & 1 & 1 & | & 0 & 1 & 1 & | & 1 & 0 & 1 & | & 1 & 0 & 0 \end{pmatrix} \end{matrix}.$$

Затем производится разбиение матрицы R на две части: R_1 «перспективных» хромосом и R_2 «неперспективных» хромосом. Разбиение, включающее элементы $\{12, 4, 7, 8, 9, 11\}$, относится к «перспективным», а разбиение $\{1, 5, 6, 2, 3, 10\}$ — к «неперспективным». В свою очередь, для распараллеливания процесса части R_1 и R_2 можно снова разбить на две части R'_1 , R''_1 и R'_2 , R''_2 . Данный процесс можно продолжать до получения подмножеств разбиения, которые легко поддаются обработке генетическими операторами. Предполагается процедура поэлементного сложения столбцов матрицы R по следующим правилам: $1 + 1 = 1$; $1 + 0 = 0$; $0 + 0 = 0$. Например, выполняя указанные сложения для «перспективных» столбцов матрицы R внутри частей R'_1 и R''_1 ($12 + 4$; $12 + 7$; $4 + 7$), получим:

$$\begin{array}{r} \begin{array}{r} 12: 1\ 1\ 1\ 1\ 0\ 1 \\ + \quad 4: 0\ 1\ 0\ 1\ 1\ 1 \\ \hline (12,4): 0\ 1\ 0\ 1\ 0\ 1 \end{array} \quad \begin{array}{r} 12: 1\ 1\ 1\ 1\ 0\ 1 \\ + \quad 7: 1\ 1\ 1\ 0\ 0\ 1 \\ \hline (12,7): 1\ 1\ 1\ 0\ 0\ 1 \end{array} \quad \begin{array}{r} 4: 0\ 1\ 0\ 1\ 1\ 1 \\ + \quad 7: 1\ 1\ 1\ 0\ 0\ 1 \\ \hline (4,7): 0\ 1\ 0\ 0\ 0\ 1 \end{array} \end{array}$$

В группе $(12, 4)$ — 3 общих свойства, в группе $(12, 7)$ — 4 общих свойства и в группе $(4, 7)$ — 2 общих свойства. Продолжая далее, согласно процедурам агрегации фракталов, получим:

$$\begin{array}{rcl}
 \begin{array}{r} 8: 1\ 1\ 0\ 1\ 1\ 0 \\ +\ 9: 1\ 0\ 1\ 0\ 1\ 1 \\ \hline (8,9): 1\ 0\ 0\ 0\ 1\ 0 \end{array} & \begin{array}{r} 8: 1\ 1\ 0\ 1\ 1\ 0 \\ +\ 11: 1\ 1\ 1\ 0\ 0\ 1 \\ \hline (8,11): 1\ 1\ 0\ 0\ 0\ 0 \end{array} & \begin{array}{r} 9: 1\ 0\ 1\ 0\ 1\ 1 \\ +\ 11: 1\ 1\ 1\ 0\ 0\ 1 \\ \hline (9,11): 1\ 0\ 1\ 0\ 0\ 1 \end{array}
 \end{array}$$

В группе (8, 9) — 2 общих свойства, в группе (8, 11) — 2 общих свойства и в группе (9, 11) — 3 общих свойства. Условимся фиксировать группы, имеющие три и более общих свойств. Выполняя аналогичные преобразования внутри «перспективных» частей R'_2 и R''_2 , получим следующие группы (1, 5) ~ 2 общих свойства; (1, 6) ~ 1; (5, 6) ~ 1; (2, 3) ~ 0; (2, 10) ~ 1; (3, 10) ~ 0. Производя сортировку полученных решений в группах, сформируем следующую популяцию: $P = \{(12, 7), (12, 4), (9, 11), (4, 7), (8, 9), (8, 11), (1, 5), (2, 10)\}$. Для реализации совместной эволюции (Ч. Дарвина, Ж. Ламарка и де Фризе) сформируем следующие родительские пары и выполним ОК:

$$\begin{array}{llll}
 p_1: 12 \mid 7 & p_2: 12 \mid 4 & p_5: 8 \mid 9 & p_6: 8 \mid 11 \\
 p_3: 9 \mid 11 & p_4: 4 \mid 7 & p_7: 1 \mid 5 & p_8: 2 \mid 10 \\
 p'_1: 12 \mid 11 & p'_2: 12 \mid 7 & p'_5: 8 \mid 5 & p'_6: 8 \mid 10 \\
 p'_3: 9 \mid 7 & p'_4: 4 \mid 4 & p'_7: 1 \mid 9 & p'_8: 2 \mid 11
 \end{array}$$

В группах после применения стандартного односточечного ОК имеем (12, 11) ~ 4 общих свойства; (9, 7) ~ 3; p'_2 совпадает с p_1 , поэтому не рассматривается, p'_4 ~ нелегальная группа; (8, 5) ~ 2; (1, 9) ~ 3; (8, 10) ~ 2; (2, 11) ~ 1. Отметим, что использование жадных и других модифицированных ОК позволит избежать получение нереальных групп. Сформируем теперь новую популяцию: $P_1 = \{(12, 7), (12, 11), (9, 11), (12, 4), (9, 7), (1, 9), (8, 5), (8, 10)\}$.

Здесь размер P_1 оставлен равным P . Как отмечалось выше, размер популяции можно варьировать в зависимости от качества получаемых решений. В принципе можно выполнить некоторое множество генераций данной операции. Произведем теперь склеивание (объединение) групп по критерию максимума общих свойств. Тогда получим следующие группы: (12, 7, 11) — 4 общих свойства; (12, 7, 9) — 3; (12, 11, 9) — 3; (7, 11, 9) — 3. Продолжая аналогично, получим группу (12, 7, 11, 9, 1) — 3 общих свойства.

Временная сложность данного алгоритма для одной генерации меняется от $O(n)$ до $O(n \log n)$. Данный алгоритм несложен и может выполняться параллельно. Ориентировочная трудоемкость алгоритма:

$$T \approx (N_p t_p + N_p t_{\text{ОК}} + N_p t_{\text{скл}}) N_G,$$

где $t_{\text{ОК}}$ — трудоемкость ОК, а $t_{\text{скл}}$ — трудоемкость операции склеивания.

Реализация алгоритмов разбиения показала преимущество ГА с нестандартными архитектурами поиска по сравнению с алгоритмами парного обмена и случайными алгоритмами. Управление процессом генетического поиска при разбиении позволяет находить оптимальные параметры. Применение модифицированных генетических операторов позволяет повысить качество разбиения. Следует заметить, что с увеличением числа

генераций в алгоритме время решения увеличивается, но оно незначительное и компенсируется получением множества локально-оптимальных решений. ГА требуют больших затрат времени, чем алгоритмы парных перестановок и случайного разбиения графов. ГА по скорости практически совпадают с итерационными и несколько хуже последовательных. Причем ГА быстрее, чем метод ветвей и границ и метод отжига. В отличие от всех рассмотренных алгоритмов разбиения, генетические позволяют получать набор квазиоптимальных результатов. Разбиение графов с применением методов ЭМ позволяет всегда получать локальные оптимумы, иметь возможность выхода из них и приближаться к получению оптимальных и квазиоптимальных решений, причем, что особенно важно, временная сложность алгоритма не уходит из области полиномиальной сложности ($\approx O(n \log n) \div O(n^3)$).

6.3. Размещение вершин графов в линейке и на плоскости

Построение и реализация инженерных оптимизационных задач на графах позволяет моделировать различные ветви эволюции. Одной из таких задач является размещение вершин графов на плоскости, в линейке, в объеме и области любой нечеткой конфигурации на основе заданных критериев оптимизации.

Алгоритмы размещения вершин графов согласно [118, 159] можно разделить на два основных класса. Это конструктивные (последовательные) алгоритмы и методы итеративного улучшения. Для их решения применяют шесть основных подходов [159]. К ним относятся: моделирование отжига; сила — направленное размещение; размещение посредством разбиения; числовые оптимизационные подходы; размещение на основе методов ЭМ; точные методы.

Опишем модифицированные ГА оптимизации размещения графовых моделей в линейке и решетке на основе метагенетической и других схем поиска. Это дает возможность, в отличие от известных алгоритмов, управлять процессом оптимизации и с полиномиальной временной сложностью получать локально-оптимальные решения.

Приведем постановку задачи. Основная цель алгоритмов размещения — минимизировать общую суммарную длину ребер графа или гиперграфа. Часто также ставится задача (или ограничение) минимизировать общее число пересечений ребер графов.

На рис. 6.12 показана модель плоскости для оптимизационной задачи размещения. На исходной плоскости устанавливается декартова система координат с осями s и t . В каждую ячейку плоскости может быть размещена вершина графа или гиперграфа. Расстояние между вершинами вычисляется на основе одной из известных формул:

$$d_{i,j} = |s_i - s_j| + |t_i - t_j| \quad (6.5)$$

или

$$d_{i,j} = \sqrt{|s_i - s_j|^2 + |t_i - t_j|^2}. \quad (6.6)$$

Здесь (s_i, t_i) ; (s_j, t_j) — координаты x_i, x_j элементов (вершин графа), $d_{i,j}$ — расстояние между элементами x_i, x_j на заданной плоскости.

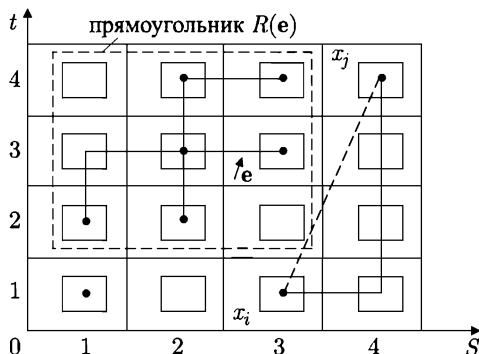


Рис. 6.12. Модель плоскости для размещения вершин графа

Выражение (6.5) позволяет определить манхеттоново расстояние, т. е. расстояние между двумя вершинами, определенное по вертикальным и горизонтальным направлениям. Шаг (расстояние) между двумя рядом лежащими вершинами по горизонтали и вертикали считается равным единице. Выражение (6.6) позволяет вычислить прямолинейное расстояние между двумя вершинами. При использовании гиперграфа длина его ребер подсчитывается как полупериметр прямоугольника, охватывающего их концевые точки (длина ребра e гиперграфа на рис. 6.12 равна 6). Тогда общая длина всех ребер графовой модели определяется по известной формуле:

$$L(G) = \frac{1}{2} \sum_{i=1}^n \sum_{j=1}^n d_{i,j} c_{i,j}. \quad (6.7)$$

Здесь $L(G)$ — общая суммарная длина ребер графа, $c_{i,j}$ — количество ребер, соединяющих вершины x_i и x_j , n — число вершин графа. Число пересечений ребер графа определяют по формуле:

$$P(G) = \frac{1}{2} \sum_{u_{i,j} \in U} P(u_{i,j}).$$

Здесь $P(G)$ — общее число пересечений ребер графа, $P(u_{i,j})$ — число пересечений ребра $u_{i,j}$, соединяющего вершины x_i, x_j .

Рассмотрим алгоритмы, минимизирующие $L(G)$ и $P(G)$ ($L(G) \rightarrow \min$, $P(G) \rightarrow \min$). Отметим, что минимизация суммарной длины ребер графа обычно приводит к минимизации числа пересечений. Такая одновременная минимизация длины и пересечений происходит до некоторого предела, начиная с которого уменьшение длины приводит к увеличению числа пересечений. Для одновременного учета длины и пересечений ребер графа вводится комплексный критерий

$$A(G) = \alpha L(G) + \beta P(G).$$

Здесь α, β — коэффициенты, учитывающие степень важности того или иного критерия ($\alpha + \beta = 1$). Значения α и β могут определяться на основе знаний ЛПР или экспертных оценок в оптимизационных задачах размещения.

В оптимизационных задачах размещения входными данными будут множества вершин графовой или гиперграфовой модели $X = \{x_1, x_2, \dots, x_n\}$, множество ребер графа $U = \{u_1, u_2, \dots, u_m\}$ или гиперграфа $E = \{e_1, e_2, \dots, e_m\}$ и множество позиций $Q = \{q_1, q_2, \dots, q_f\}$, причем $|X| \leq |Q|$, если производится размещение вершин на заданные позиции. Иногда производится размещение ребер, и тогда число позиций должно быть $\geq$ числа концевых точек размещаемых ребер.

Сформулируем задачу размещения как назначение каждой вершины графа (гиперграфа) в единственную позицию таким образом, чтобы оптимизировать ЦФ. В качестве ЦФ выбирается выражение (6.7). Выходными данными будут позиции назначения для каждой вершины. В качестве ограничений выбираются следующие условия. Для каждого $x_i \in X$ назначается только одна позиция $q_i \in Q$; каждой позиции $q_i \in Q$ соответствует, по крайней мере, один элемент $x_i \in X$. Следовательно, сформулирована задача размещения как оптимизационная. Известно, что проблема размещения относится к классу NP-полных [159, 160]. Поэтому постоянно производится поиск эвристических процедур для повышения качества и уменьшения сложности алгоритмов. Алгоритм считается «вероятностно хорошим алгоритмом», если его временная сложность $O(n)$, $O(n \log n)$, $O(n^2)$.

Рассмотрим размещение вершин графа на плоскости в решетке заданной конфигурации. В настоящее время выделяют два подхода применения ЭМ для размещения вершин графа на плоскости. Первый подход использует методологию Р. Клинга для размещения элементов, одинаковых по высоте и различных по ширине, на основе ЭМ [166]. Второй подход основан на идеях Д. Кохона, В. Париса, М. Майера, Й. Сааба, Н. Шервани, Р. Шахокара и П. Мазумдера [117, 167–172].

Первый подход заключается в следующем. Сначала формируется популяция решений, обычно случайным образом. Далее в каждом элементе популяции выполняется оператор мутации. Затем вычисляется ЦФ, производится сортировка решений и назначение элементов в заданные позиции. В случае необходимости производится возврат к ОМ, и вся процедура продолжается аналогично до преждевременной сходимости или пока не выполнится заданное число итераций. Данный подход является простым в реализации и имеет временную сложность алгоритма линейного типа, но получаемые локальные оптимумы далеки от глобального. Это связано с большим упрощением ЭМ и неиспользованием всей многогранности и широких возможностей моделей эволюции.

Предлагается модификация данного подхода для размещения вершин графа (гиперграфа) на плоскости и в линейке с оптимизацией выражения (6.7). Она заключается в использовании дополнительных генетических операторов, гомеостатического управления и синергетических и иерархических принципов, имеющих аналогии в ЕС. На рис. 6.13 приведена струк-

турная схема алгоритма размещения вершин графа на плоскости на основе эволюций Ч. Дарвина, Ж. Ламарка, де Фриза и К. Поппера. Отметим, что все разработанные архитектуры поиска, приведенные выше, используются для распараллеливания алгоритмов и совместного применения эвристического, стохастического поиска и ЭМ.

В ГА на рис. 6.13 начальная популяция будет конструироваться тремя способами A_1 , A_2 , A_3 . В случае A_1 популяция решений (заданных размещений) формируется случайным образом. В случае A_2 популяция решений получается путем неоднократного применения последовательного алгоритма. Формирование популяции A_3 производится совместно случайным и направленным образом. Отметим, что ЛПР на основе блока эволюционной адаптации может создавать любое число популяций другими способами. После формирования популяции к каждому ее элементу применяется оператор мутации, причем ОМ может выполняться независимо для каждого элемента популяции.

Заметим, что для задач размещения хорошие результаты (локальные оптимумы) дает применение следующих операторов мутации: ОМ метода золотого сечения, ОМ метода Фибоначчи, ОМ метода дихотомии, ОМ множества Кантора, а также ряд других модификаций, за счет получения реальных решений оптимизационных задач [171].

Затем выполняется оператор инверсии. Предлагаются две версии данного оператора: случайная и направленная. Здесь также предусматривается применение следующих операторов инверсии: специального, метода золотого сечения, метода Фибоначчи, метода дихотомии, а также ряд других модификаций для получения реальных решений оптимизационных задач.

Далее выполняются три версии оператора сегрегации. В первой используется «жадная» стратегия. Во второй — стратегия золотого сечения. В третьей используется метод Монте-Карло. Проводились эксперименты с операторами сегрегации метода золотого сечения, метода Фибоначчи, метода дихотомии и др.

Затем осуществляется переход к выполнению оператора транслокации. Также вводятся две основных версии оператора транслокации: случайная и направленная. Кроме этого, использовались операторы транслокации метода золотого сечения, метода Фибоначчи, метода дихотомии, множества Кантора и др.

Последними выполняются операторы кроссинговера. На рисунке ОК₁ — однотоочечный, ОК₂ — двухтоочечный, ОК₃ — упорядоченный, ОК₄ — циклический и ОК₅ — модифицированный. Наилучшие результаты показывают жадные, фрактальные и дихотомические стратегии. В данной схеме порядок выполнения генетических операторов зависит от внешней среды, блока эволюционной адаптации и может иметь любой установленный порядок. Блок эволюционной адаптации может задать хаотичный (случайный) порядок выполнения генетических операторов.

Заметим, что перед началом работы алгоритма предварительно вычисляется верхняя оценка минимального размещения для заданной конфигурации плоскости. После выполнения генетических операторов их ре-

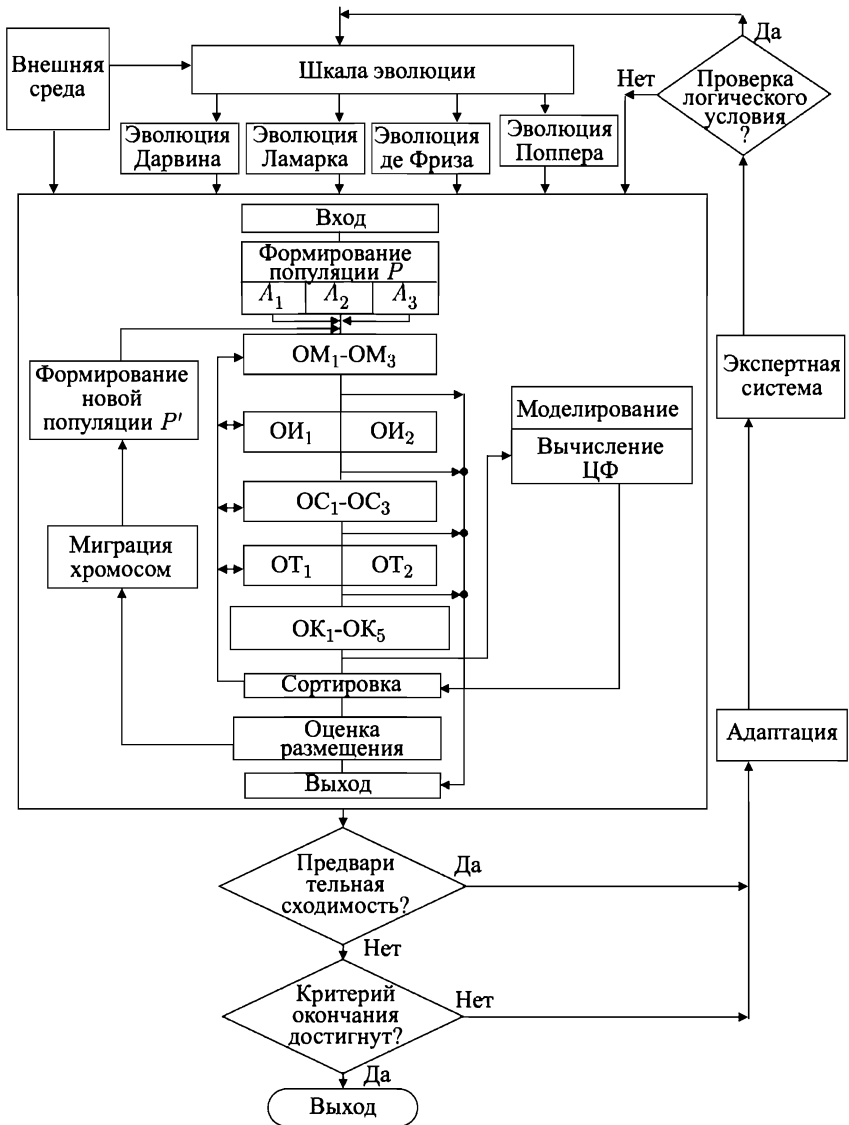


Рис. 6.13. Схема алгоритма размещения на основе эволюций.
ОС и ОТ — операторы сегрегации и транслокации соответственно

зультаты сравниваются с оценкой. При получении искоемых результатов алгоритм заканчивает свою работу. После реализации ОК для каждой хромосомы вычисляется ЦФ, и на ее основе сортируются варианты решений. Затем формируется новая популяция, и процесс продолжается аналогично. Возможен выход из каждого оператора в отдельности при получении локального оптимума.

Рассмотрим пример размещения вершин графа в одной линейке. Пусть задан граф, показанный на рис. 6.14 [159]. Данный граф необходимо разместить в линейке с шестью позициями с минимизацией суммарной длины ребер. При линейном размещении в случае небольшого количества элементов перспективными являются метод ветвей и границ, симплекс метод и метод отжига. С ростом числа вершин графов эффективное применение находят методы ЭМ и их различные модификации.

Пусть для графа $G = (X, U)$, $|X| = 6$, $|U| = 10$, показанного на рис. 6.14, случайным образом получена следующая популяция решений (размещение вершин графа в линейке):

$$p_1: 1 \ 2 \ 3 \ 4 \ 5 \ 6 \quad p_3: 2 \ 4 \ 6 \ 1 \ 3 \ 5 \quad p_5: 4 \ 6 \ 2 \ 5 \ 1 \ 3$$

$$p_2: 1 \ 3 \ 5 \ 2 \ 4 \ 6 \quad p_4: 3 \ 5 \ 1 \ 6 \ 4 \ 2 \quad p_6: 5 \ 1 \ 4 \ 3 \ 6 \ 2$$

Следуя [160], определим оценку минимальной суммарной длины для данного графа. Она определяется при расположении ребер графов по кратчайшим расстояниям без учета изоморфизма. При этом кратные ребра упорядочиваются по длине и располагаются в ближайшие позиции по убыванию длины. В нашем случае для графа $G = (X, U)$ (рис. 6.14) наилучшая оценка будет равна $L(G\Delta)_{\min} = 13$. Теперь вычислим ЦФ для каждой хромосомы в популяции. Тогда суммарная длина связей для альтернативных решений $P_1 - P_6$ определится так:

$$L(G)_1 = 1 \cdot 4 + 2 \cdot 2 + 3 \cdot 2 + 4 \cdot 1 + 5 \cdot 1 = 23;$$

$$L(G)_2 = 28; L(G)_3 = 22; L(G)_4 = 19; L(G)_5 = 24; L(G)_6 = 25.$$

Произведем сортировку хромосом согласно их ЦФ: $p_4; p_3; p_1; p_5; p_6; p_2$. Начиная с p_4 , применим ОМ. Пусть точка мутации случайным образом определится между 5 и 6 элементами. Тогда получим

$$p_4: 3 \ 5 \ 1 \ 6 \ 4 \mid 2 \quad L(G)_4 = 23,$$

$$p'_4: 3 \ 5 \ 1 \ 6 \ 2 \ 4 \quad L(G)'_4 = 20.$$

Далее предлагаются два основных решения. В первом выполняется мутация всех p_i из P и строится новая популяция P' . Во втором популяция изменяется после каждого ОМ. Например, вставим p'_4 в P' , исключив p_2 , тогда $P' = \{p_4, p'_4, p_3, p_5, p_1, p_6\}$. Покажем теперь применение случайной

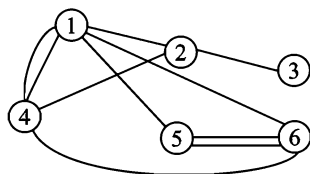


Рис. 6.14. Граф G

версии оператора инверсии. Возьмем p_4 , в котором случайным образом точка инверсии попадает между 1 и 2 элементами. Тогда

$$p_4: 3 \mid 5 \ 1 \ 6 \ 4 \ 2 \quad L(G)_4 = 23,$$

$$p'_4: 3 \mid 2 \ 4 \ 6 \ 1 \ 5 \quad L(G)'_4 = 16.$$

Величина $L(G)$ приближается к глобальному минимуму. На этом процесс размещения может быть закончен. На рис. 6.15 показано это размещение.

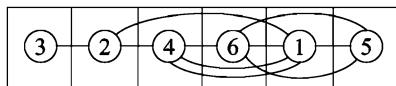


Рис. 6.15. Размещение графа G ($L(G) = 16$)

Отметим, что в алгоритме на основе блока эволюционной адаптации возможны обратные связи для управления поиском после каждого генетического оператора. Так, например, для лучшей хромосомы p'_4 выполним направленный оператор мутации.

$$p'_4: 3 \ 2 \ 4 \ 6 \mid 1 \ 5 \quad L(G)'_4 = 16,$$

$$p''_4: 3 \ 2 \ 4 \ 1 \mid 6 \ 5 \quad L(G)''_4 = 13 = L(G) - \text{мин.}$$

На рис. 6.16 показано размещение графа G (рис. 6.14) с глобальным минимумом $L(G)$ в линейке. Для того, чтобы получить размещение, во всех линейках можно применять описанные выше ГА разбиения. Для этого нужно разбить граф на заданное число линейек, с минимизацией количества ребер между линейками, а затем произвести оптимизацию размещения внутри каждой линейки, используя методы генетического поиска.

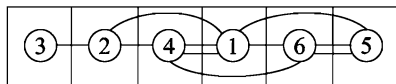


Рис. 6.16. Оптимальное размещение графа G в линейке

Определим ориентировочную трудоемкость разработанного линейного алгоритма:

$$T \approx [N_p t_p + N_p (t_{OM} + t_{OI} + t_{OC} + t_{OT} + t_{OK})] N_G,$$

здесь t_p — трудоемкость построения одной хромосомы в популяции; t_{OM} — трудоемкость оператора мутации; t_{OI} — трудоемкость оператора инверсии; t_{OC} — трудоемкость оператора сегрегации; t_{OT} — трудоемкость оператора транслокации; N_G — число генераций, N_p — размер популяции.

Временная сложность алгоритма линейного размещения меняется от $O(n)$ до $O(n^2)$. Вероятность получения оптимального результата можно определить так:

$$P_i(\text{ОПТ}) = P_i(N_p + N_n) N_G,$$

где P_i — вероятность получения оптимального результата при генерации одной хромосомы в популяции; N_n — число полученных потомков.

Второй подход к размещению вершин графа на плоскости основан на использовании классических ГА и принципов метагенетической оптимизации. Это позволяет повысить качество решений при увеличении затрат времени. Опишем применение описанной выше архитектуры микро-, макро- и метагенетической оптимизации для размещения вершин графа в решетке. Структурная схема алгоритма с относительным балансом показана на рис. 6.17.

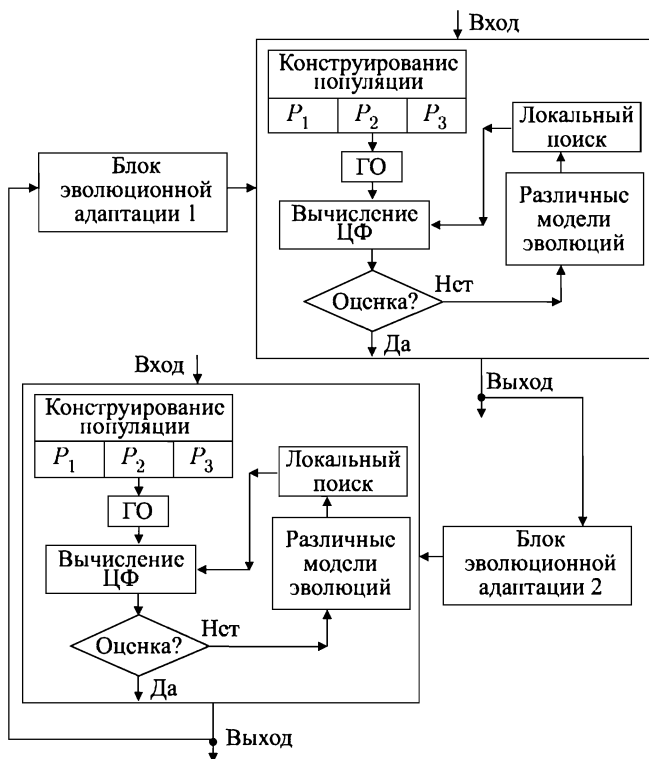


Рис. 6.17. Структурная схема алгоритма с относительным балансом

Здесь популяция создается тремя способами: направленно P_1 , случайно-направленно P_2 и случайно P_3 . Минимальный размер популяции $n/2$, где $n = |X|$ — число вершин графа. Минимальное число генераций $N_G = 10n$. Схема алгоритма (рис. 6.17) состоит из двух строительных блоков. Она построена таким образом, что в процессе работы устанавливается относительный баланс, причем на вход одного строительного блока подается набор случайно полученных хромосом, а на вход второго — заданный набор хромосом. Далее моделируем каждую хромосому путем выполнения ГА с ОК, ОМ, операторами сегрегации, транслокации, инверсии и другими генетическими операторами. Затем, определяя ЦФ, находим лучшее размещение с заданными параметрами. Выполняем работу алгоритма посредством вы-

бора параметров и использования блока эволюционной адаптации. Из множества родителей и потомков выбираются решения с лучшими ИЦФ, которые копируются в новую популяцию. При этом ее размер остается постоянным.

В заключение производится выбор наилучшего множества параметров из конечной популяции. Размер популяции обычно выбирается, исходя из опыта ЖПР и имеющихся вычислительных ресурсов. Теоретические исследования [163] рекомендуют размер популяции $N_p \approx n/2$, где n — число вершин графа. Определения числа генераций, наряду с размером популяций, являются открытыми проблемами в ГА. Отметим, что глобальный минимум может быть получен как на первой генерации, так и не найден на последней. Применение стандартных ГА и ПГА может не давать реальных размещений, поэтому необходимо использовать различные модифицированные генетические операторы и новые архитектуры реализации генетического поиска, приведенные выше.

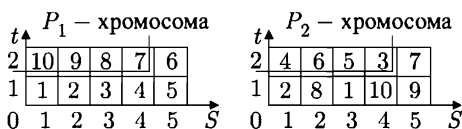


Рис. 6.18. Пример двух сегментов ячеек

Рассмотрим пример размещения графа на плоскости с минимизацией $L(G)$ (6.7). Пусть задана плоскость, на которую нанесена решетка в декартовой системе координат с осями s, t . Размеры решетки 2×5 . Разместим в этой решетке граф $G = (X, U)$, $|X| = 10$. На рис. 6.18 показаны два родительских сегмента (P_1, P_2) размещения вершин графа.

p_1 : 1 2 3 4 5 6 | 7 8 9 10 (вершины графа),
 s — 1 2 3 4 5 5 | 4 3 2 1 (координаты ячеек),
 t — 1 1 1 1 1 2 | 2 2 2 2 (координаты ячеек);

 p_2 : 2 8 1 10 9 7 | 3 5 6 4 (вершины графа),
 s — 1 2 3 4 5 5 | 4 3 2 1 (координаты ячеек),
 t — 1 1 1 1 1 2 | 2 2 2 2 (координаты ячеек).

Здесь p_1 и p_2 — две случайно выбранных хромосомы из популяции P , которые моделируют альтернативные результаты размещения вершин графа в решетке. Применим к p_1 и p_2 одноточечный ОК. Тогда

p'_1 : 1 2 3 4 5 6 | 3 5 6 4.

В p'_1 имеются повторяющиеся вершины (3, 5, 6, 4), поэтому использование стандартного одноточечного ОК нецелесообразно. Простая модификация любого стандартного ОК позволит получать реальные размещения. Используя модифицированный ОК, выбираем в p_1 левый от точки ОК сегмент хромосомы и копируем его в p'_1 . В p_2 хромосоме выбираем правый от точки ОК сегмент и производим его анализ на предмет отсутствия повторяющихся генов. Все повторяющиеся гены исключаются из p'_1 , а вместо

них добавляются отсутствующие элементы из левого сегмента p_2 . Тогда в рассматриваемом примере получим:

$$\begin{array}{l} p'_1: \quad 1 \quad 2 \quad 3 \quad 4 \quad 5 \quad 6 \quad 8 \quad 10 \quad 9 \quad 7 \\ s \text{ —} \quad 1 \quad 2 \quad 3 \quad 4 \quad 5 \quad 5 \quad 4 \quad 3 \quad 2 \quad 1 \\ t \text{ —} \quad 1 \quad 1 \quad 1 \quad 1 \quad 1 \quad 2 \quad 2 \quad 2 \quad 2 \quad 2. \end{array}$$

На рис. 6.19 показано возможное оптимальное размещение ячеек хромосомы $p'_1 : (1, 2, 3, 4, 5, 6, 8, 10, 9, 7)$ после применения модифицированного ОК. В данной схеме возможно применение любых генетических операторов и их модификаций. Наилучшие результаты дают жадные стратегии, циклический ОК, алгоритмы на основе фракталов и др.

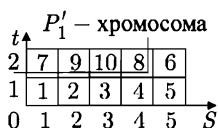


Рис. 6.19. Размещение после выполнения модифицированного ОК

Важным вопросом в метагенетическом алгоритме размещения является понятие шкал генетических операторов. Обозначим размер популяции P через N_p , число генераций — N_G , число размещаемых вариантов для копирования — N_{off} . Тогда шкалы ОК и ОМ можно определить следующим образом:

$$R_{\text{ок}} = \frac{N_{\text{off}}}{N_p}.$$

Здесь ОК управляется посредством шкалы ОК ($R_{\text{ок}}$); $R_{\text{ок}}$ определяется как отношение числа потомков, полученных в каждой генерации ГА, к размеру популяции. Отметим, что $R_{\text{ок}}$ косвенно оценивает отношение числа точек разрыва (применение ОК) в областях, где ЦФ имеет лучшее значение, к числу точек разрыва в других местах популяций. Увеличение значения $R_{\text{ок}}$ ($R_{\text{ок}} > 50$) позволяет анализировать большее пространство и повышает шанс нахождения оптимума. В то же время большое значение $R_{\text{ок}}$ требует огромных ресурсов вычислительного времени. Количество генераций в оптимизационной задаче можно находить по формуле

$$N_G = N_{Gn} \frac{N_p}{N_{\text{off}}}.$$

Здесь N_{Gn} — количество генераций ГА, задаваемых пользователем.

Шкала мутации $R_{\text{ом}}$ определяется как процент общего числа генов в популяции, которые в ней меняются местами (мутируют). Так, при размещении n вершин графа в решетке при размере популяции N_p

$$R_{\text{ом}} = \frac{n \cdot N_p \cdot R_{\text{ок}}}{2}.$$

В этом выражении nN_p — число генов в популяции P . Величина $R_{\text{ом}}$ характеризует парный обмен генов при мутации. Шкалы остальных генетических операторов определяются аналогично.

Для размещения вершин графа в решетке можно использовать построение минимального кластера и агрегацию фракталов.

На рис. 6.19, а показан граф $G = (X, U)$, $|X| = 9$, расположенный в решетке 3×3 с $L(G) = 31$.

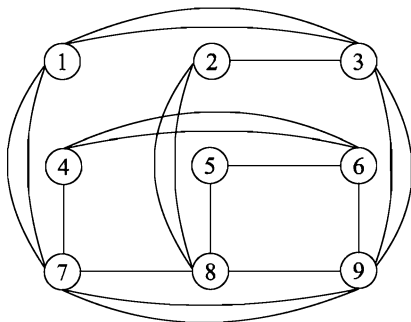


Рис. 6.19. а. Граф $G = (X, U)$, $|X| = 9$

Здесь в качестве ЦФ выбрана $L(G)$. Необходимо из заданного хаоса (случайного расположения вершин) получить порядок с $L(G) \rightarrow \min$. Построим фрактальное множество. Для этого каждую строку решетки представим в виде одной вершины нового графа G_1 с $|X_1| = 3$. На рис. 6.19, б показан результат такого преобразования. Применим алгоритм агрегации фракталов. Кроме того, здесь также эффективно используются модифицированные ОМ, оператор инверсии и оператор сегрегации, разработанные авторами. На рис. 6.19, в приведен оптимальный результат расположения графа рис. 6.19, б в линейке.

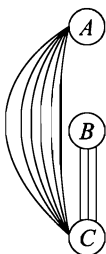


Рис. 6.19. б. Граф G_1 до преобразования



Рис. 6.19. в. Граф G_1 после преобразования

После перехода от фрактального графа G_1 к графу G получим расположение графа G в решетке, показанное на рис. 6.19, г. При этом $L(G) = 25$. Аналогичные преобразования для графа G (рис. 6.19, г) выполним при построении фракталов по столбцам (G_2 с $|X_2| = 3$). Эти преобразования показаны на рис. 6.19, д. и 6.19, е. Окончательное размещение графа G (рис. 6.19, а) с $L(G) = \min = 20$ приведено на рис. 6.19, ж. Заметим, что такие преобразования в графе, расположенном в решетке, можно выполнять не только по строкам и столбцам решетки, но и для области любой конфигурации.

В заключение отметим, что сложность описанных алгоритмов размещения является полиномиальной величиной. Для одной генерации временная сложность алгоритма лежит в пределах $O(n^2) \div O(n^3)$. Заметим, что при размещении графов использование методов ЭМ позволяет получать набор локально-оптимальных решений. При этом с большой вероятностью среди этих решений может быть найдено оптимальное. Из известных статистических данных следует, что в общем случае время решения практически линейно зависит от количества генераций и временная сложность последовательного и итерационного алгоритмов подтверждает теоретические предпосылки и приблизительно равна $O(n^2)$.

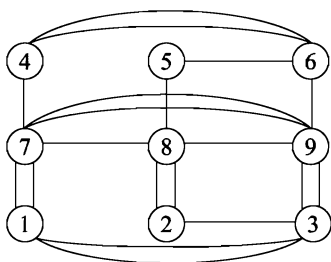


Рис. 6.19. г. Граф G после первого преобразования.

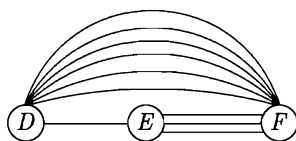


Рис. 6.19. д. Граф G_2 до преобразования

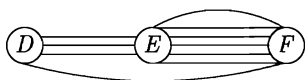


Рис. 6.19. е. Граф G_2 после преобразования

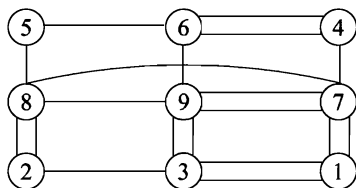


Рис. 6.19. ж. Окончательное размещение графа G

Рассмотренные алгоритмы размещения вершин графа в линейке и на плоскости на основе модифицированных генетических операторов (кроссинговера, мутации, инверсии, транслокации и сегрегации) позволяют за счет применения новых архитектур генетического поиска и эффективно управления им на основе иерархического подхода получить набор оптимальных решений. Для дальнейших исследований можно предложить ряд новых архитектур ЭМ с обратными связями, автоматами адаптации, синергетическими и гомеостатическими принципами управления поиском решений, позволяющими выбирать оптимальные эвристики с учетом сложности оптимизационной задачи, организовывать итерационные циклы поиска. Это может дать возможность распараллеливать процесс оптимизации, эффективно управлять поиском, получать оптимальные и квазиоптимальные решения за время, сопоставимое с временем реализации итерационных алгоритмов.

6.4. Генетические алгоритмы построения деревьев Штейнера

Заключительным этапом конструкторского синтеза интегральных схем является комплексная трудоемкая задача, определяющая качество всего процесса — трассировка соединений [159, 160]. Задачу трассировки представляют в виде укрупненных четырех этапов: определение перечня соединений; размещение по слоям; определение порядка соединений; трассировка проводников.

Рассмотрим алгоритмы решения задачи первого этапа. Основными алгоритмами здесь являются алгоритмы построения кратчайших связывающих деревьев Прима и Штейнера. Например, на рис. 6.20 показаны пять контактов цепи, которые должны быть соединены.

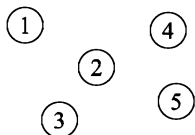


Рис. 6.20. Пять контактов цепи

Обычно ставится задача соединения заданных точек кратчайшим образом (сумма расстояний между точками должна быть минимальной или стремиться к ней). На рис. 6.21 показано одно из возможных кратчайших связывающих деревьев, построенных на основе алгоритма

Прима [159]. Условная суммарная длина $L(T)$ кратчайших связывающих деревьев рассчитывается по формуле (6.5). Для примера рис. 6.21 получим $L(T) = 23$.

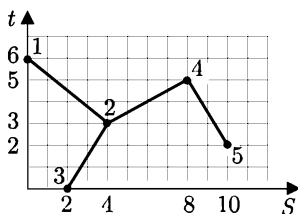


Рис. 6.21. Кратчайшее связывающее дерево $L(T) = 7 + 5 + 6 + 5 = 23$

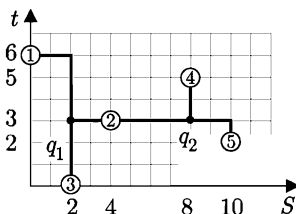


Рис. 6.22. Дерево Штейнера, 2 точки Штейнера, $L(T) = 19$

Для минимизации суммарной длины соединений в кратчайшем связывающем дереве предложено при соединении множества вершин (точек, контактов цепей) использовать дополнительные точки (вершины). Возникающую при этом задачу называют задачей Штейнера [118]. Она формулируется следующим образом. Для заданных n точек плоскости построить соединяющее их кратчайшее связывающее дерево с $n' \geq n$ точками (вершинами), общая суммарная длина соединений которого минимальна. Построенное таким образом дерево называется деревом Штейнера, дополнительные вершины — точками Штейнера. На рис. 6.22 показано одно из возможных деревьев Штейнера для пяти соединяемых контактов (рис. 6.20). Суммарная длина соединений дерева Штейнера (рис. 6.22) $L(T) = 19$ и введено две точки Штейнера (q_1, q_2). Задача Штейнера, как известно, является NP-полной. Для $n = 3$ известно точное решение задачи. Для $n = 4$ существует ряд эффективных алгоритмов. При $n \geq 5$ требуется огромное число

вычислений, поэтому продолжается разработка различных эвристических алгоритмов решения данной задачи.

В [118] сформулирован ряд лемм и теорем о построении деревьев Штейнера. Так, при соединении трех контактов единственная точка Штейнера находится внутри прямоугольника, связывающего эти точки, причем координаты единственной точкой Штейнера определяются как S_{cp} и t_{cp} — значения n_i при $i = 1, 2, 3$ и

$$\sum_{i=1}^3 d(q, n_i) = \frac{1}{2} P[R(n_1, n_2, n_3)],$$

где $P[R]$ — длина периметра прямоугольника, построенного на трех точках. Так, для дерева Штейнера (рис. 6.22) q_1 и q_2 будут иметь координаты, определяемые следующим образом:

$$\begin{array}{llll} S_1 = 0, & S_2 = 4, & S_3 = 2, & S_{cp\ q_1} = 2; \\ t_1 = 6, & t_2 = 3, & t_3 = 0, & t_{cp\ q_1} = 3; \\ S_2 = 4, & S_4 = 8, & S_5 = 10, & S_{cp\ q_2} = 8; \\ t_2 = 3, & t_4 = 5, & t_5 = 2, & t_{cp\ q_2} = 3. \end{array}$$

Длина дерева Штейнера (рис. 6.22) определяется

$$L(T) = \frac{1}{2} P_1(R(1, 2, 3)) + \frac{1}{2} P_2(R(2, 4, 5)) = 19.$$

Предложим простой эвристический алгоритм, временная сложность которого $O(n)$. Идея построения алгоритма заключается в следующем. Выбираются первые три точки из n точек, на которых должно быть построено дерево Штейнера. Согласно [118] строится оптимальный фрагмент дерева Штейнера для трех точек. Затем выбираются следующие по порядку три точки, одна из которых принадлежит построенному фрагменту. На этих точках строится второй оптимальный фрагмент дерева Штейнера. Далее процесс повторяется аналогично, пока не будет построено дерево Штейнера. Отметим, что каждый фрагмент дерева Штейнера оптимальный, но полное дерево Штейнера в общем случае может оказаться не оптимальным. При наличии достаточных вычислительных ресурсов или распараллеливания алгоритма можно строить оптимальные фрагменты по три вершины и, выполнив перебор, можно получить оптимальное или квазиоптимальное дерево Штейнера.

Предлагается упрощенная эвристическая процедура построения дерева Штейнера, основанная на так называемых «столбах» Штейнера. Из любой точки, которую требуется соединить деревом Штейнера, проводится вертикальный столб Штейнера, а затем из остальных точек проводятся перпендикулярные отрезки на этот столб. Например, на рис. 6.23 показано дерево Штейнера, построенное таким образом. Здесь две точки Штейнера

и $L(T) = 22$. Вертикальный столб можно проводить в любом месте плоскости (желательно в прямоугольнике, ограничивающем заданные точки). Аналогичным образом строится горизонтальный столб Штейнера и дерево Штейнера с использованием описанной выше идеи. На рис. 6.24 показано такое дерево Штейнера с двумя точками Штейнера и $L(T) = 19$.

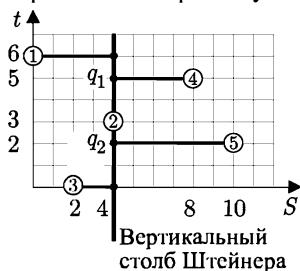


Рис. 6.23. Дерево Штейнера 2 точки Штейнера, $L(T) = 22$

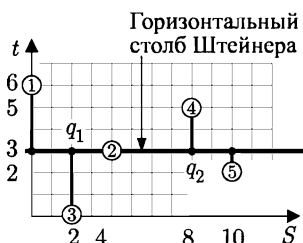


Рис. 6.24. Дерево Штейнера 2 точки Штейнера, $L(T) = 19$

В связи с возможностью быстрого получения набора различных деревьев Штейнера, по нашему мнению, представляется возможным использовать схемы генетического поиска, описанные выше, для построения прямоугольных деревьев Штейнера.

Сначала, используя предложенные эвристические алгоритмы, строится популяция $P^I = \{p_1^I, p_2^I, \dots, p_k^I\}$, каждый элемент популяции представляет собой закодированное дерево Штейнера. Далее производится сортировка популяции и селекция родительских пар. Затем возможно применение любых из описанных выше генетических операторов на основе мета-, микро- или макроэволюции.

Рассмотрим пример. Пусть необходимо построить дерево Штейнера на пяти точках (рис. 6.20). Используя горизонтальные столбы Штейнера, построим четыре дерева Штейнера (рис. 6.25–6.28). На рис. 6.25 столб проходит через точку 1; на рис. 6.26 — через 4; на рис. 6.27 — через 5; на рис. 6.28 — через 3. Получили популяцию $P^I = \{p_1^I, p_2^I, p_3^I, p_4^I\}$. Кодирование каждой хромосомы в популяции будем проводить следующим образом (отметим, что П — означает движение прямо, Н — направо, В — влево):

$$p_1^I: 1 \text{ П Т}_1 \text{ Н З П Т}_2 \text{ Н 2 П Т}_3 \text{ Н 4 П Н 5} \quad (1\text{-столб}) - 1$$

$$p_2^I: 1 \text{ Н П Т}_1 \text{ Н З П Т}_2 \text{ Н 2 П 4 П Н 5} \quad (2\text{-столб}) - 4$$

$$p_3^I: 1 \text{ Н П Т}_1 \text{ Н З П Т}_2 \text{ В 2 П Т}_3 \text{ В 4 П 5} \quad (3\text{-столб}) - 5$$

$$p_4^I: 1 \text{ Н П 3 П Т}_1 \text{ В 2 П Т}_2 \text{ В 4 П В 5} \quad (4\text{-столб}) - 3$$

Согласно вычисленной суммарной длине выбранных деревьев Штейнера, соответствующих хромосомам $p_1^I, p_2^I, p_3^I, p_4^I$, произведем сортировку деревьев Штейнера. Тогда в результате получим $p_3^I, p_2^I, p_1^I, p_4^I$. Построим новую популяцию $P^{II} = \{p_1^{II}, p_2^{II}, p_3^{II}, p_4^{II}\}$, используя вертикальные столбы Штейнера. На рис. 6.23, 6.29–6.31 показаны четыре дерева Штейнера.

Произведя их сортировку, получим: p_2^{II} , p_3^{II} , p_4^I , p_1^{II} . Выберем в качестве родительских пар элитные элементы из первой и второй популяций:

$$\begin{array}{l} p_3^I : \quad | 1 \text{ Н П Т}_1 \text{ Н З П} | \text{Т}_2 \text{ В 2 П Т}_3 | \text{В 4} | \text{П 5} \\ \hline p_2^I : \quad | 1 \text{ П Н Т}_1 \text{ П 4 Н} | \text{2 Н Т}_2 \text{ П 5} | \text{Н Л 5} \\ \hline p_{3,2}' : \quad | 1 \text{ Н П Т}_1 \text{ Н З П 2 Н Т}_2 \text{ П 5 В 4} \end{array}$$

На примере выбранных родительских пар p_3^I и p_2^{II} покажем работу модифицированного оператора сегрегации. Он может выполняться на одной, двух и так далее числе хромосом. Когда оператор сегрегации выполняется на одной хромосоме, тогда точки сегрегации позволяют определить строительные блоки (части хромосомы) для формирования потомка. В этом случае как бы происходит перераспределение генетического материала внутри одной хромосомы. При реализации оператора сегрегации на двух и более хромосомах точки оператора сегрегации, выбранные направленно или случайно направленно, позволяют найти строительные блоки и скопировать их в новую хромосому или в некоторое множество хромосом (потомков).

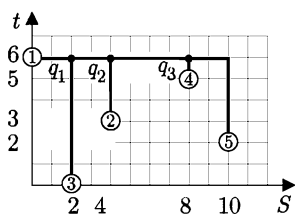


Рис. 6.25. Дерево Штейнера
 $L(T) = 24$, 3 точки Штейнера

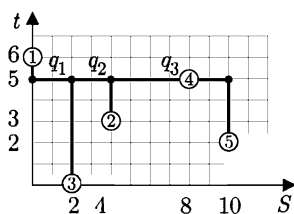


Рис. 6.26. Дерево Штейнера
 $L(T) = 21$, 3 точки Штейнера

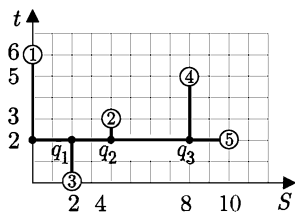


Рис. 6.27. Дерево Штейнера
 $L(T) = 20$, 3 точки Штейнера

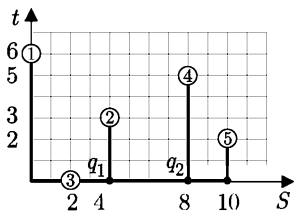


Рис. 6.28. Дерево Штейнера
 $L(T) = 26$, 2 точки Штейнера

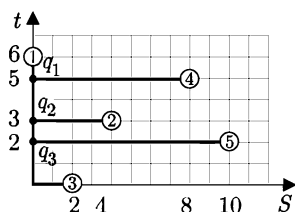


Рис. 6.29. Дерево Штейнера
 $L(T) = 30$, 3 точки Штейнера

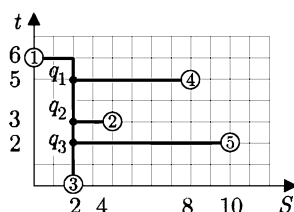


Рис. 6.30. Дерево Штейнера
 $L(T) = 24$, 2 точки Штейнера.

На рис. 6.32 показано дерево с двумя точками Штейнера и $L(T) = 20$. После применения модифицированного оператора сегрегации здесь первый строительный блок позволил построить фрагмент, соединяющий точки 1 и 3. Второй строительный блок — точки 2, 5, и третий строительный блок позволил подсоединить точку 4. Отметим, что мы специально не включили в популяцию P^1 лучшее решение с горизонтальным столбом, проходящим через точку 2. При включении этого элемента в популяцию было бы получено другое дерево Штейнера с $L(T) = 19$. С увеличением числа точек время построения дерева Штейнера зависит от количества элементов популяции и числа генераций.

Определим ориентировочную трудоемкость алгоритма:

$$T \approx t_k + [N_n t_n + (N_p + N_n) t_{oc} + N_p t_c] N_G,$$

где t_k — трудоемкость кодирования дерева Штейнера, t_n — трудоемкость получения одного потомка, N_n — количество потомков, t_c — трудоемкость оператора селекции, t_{oc} — трудоемкость оператора сегрегации. Отметим, что трудоемкость получения одного потомка меняется в пределах $O(n)$ — $O(n^2)$ и временная сложность алгоритма для разработанных эвристических алгоритмов дерева Штейнера $\approx O(n^2)$.

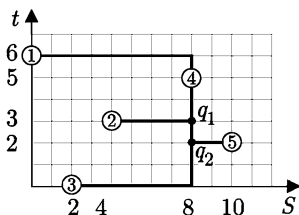


Рис. 6.31. Дерево Штейнера $L(T) = 26$. 2 точки Штейнера

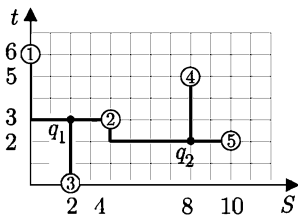


Рис. 6.32. Дерево Штейнера $L(T) = 20$, 2 точки Штейнера

Отметим, что основной сложностью применения ГА для построения прямолинейных деревьев Штейнера является оптимальное кодирование хромосом в популяции. Кроме того, остаются все проблемы селекции, предварительной сходимости алгоритма и выбора эффективных генетических операторов.

6.5. Трассировка соединений

Существует большое число работ по канальной трассировке топологии СБИС [174–179].

Рассмотрим прокладку соединений на основе канальных алгоритмов трассировки. Канальные алгоритмы базируются на представлении о каналах и магистралях. Магистралью называют отрезок прямой, по которой может проходить соединение в преимущественном направлении. Канал — это область прямоугольной формы, на одной или нескольких сторонах которой расположены контакты с системой однонаправленных магистралей. Каждая цепь, т. е. соединение эквипотенциальных контактов, представлена

как одиночный горизонтальный сегмент с несколькими вертикальными сегментами, которые соединяют горизонтальный сегмент с контактами цепи. Горизонтальные сегменты располагаются в одном слое, вертикальные — в другом. Соединения между горизонтальными и вертикальными сегментами делаются через переходные отверстия.

Основная задача канальной трассировки — выбор наименьшей ширины канала, достаточной для размещения в нем всех соединений и назначения соединений на магистрали. Кроме того, необходимо минимизировать суммарную длину соединений (цепей), число переходных отверстий и т. п.

Задача канальной трассировки в классической постановке основана на трассировке двухстороннего канала, по верхней и нижней сторонам которого проходят линейки контактов. Изломы, т. е. переходы горизонтального участка с одной магистрали на другую, не допускаются.

Канал описывается двумя последовательностями Top и Bottom, в которых размещены верхняя и нижняя линейки канала, соответственно. Размер обеих последовательностей равен C — числу колонок в канале. Множество цепей определяется как $\text{Net} = \{N_1, \dots, N_m\}$, где m — число цепей.

Рассмотрим следующий пример: Top = {1, 0, 3, 1, 4, 2, 3, 2} и Bottom = {6, 4, 6, 6, 3, 0, 5, 5}, где $C = 8$, $n = 6$, $\text{Net} = \{1, 2, 3, 4, 5, 6\}$, элемент 0 в Top или в Bottom обозначает свободный контакт. На рис. 6.33 дан эскиз канала с разведенными цепями.

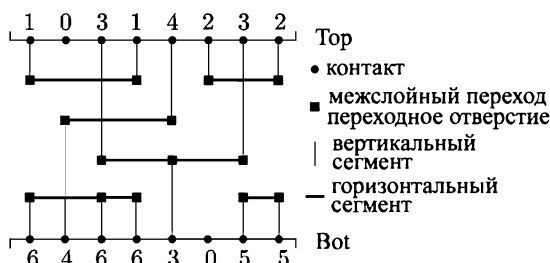


Рис. 6.33. Эскиз канала с разведенными цепями

При канальной трассировке не допускаются наложения вертикальных и горизонтальных сегментов цепей. Для решения этой задачи вводятся графы вертикальных и горизонтальных ограничений. Вертикальные ограничения описываются ориентированным графом вертикальных ограничений (рис. 6.34, а) $G_V = (E_{\text{Net}}, E_V)$, где E_{Net} — множество вершин, соответствующих множеству цепей Net, E_V — множество направленных ребер. Ребро $(n, m) \in E_V$ существует тогда и только тогда, когда цепь n должна быть расположена выше цепи m для предотвращения наложений вертикальных сегментов цепей. Например, на рис. 6.34 в графе вертикальных ограничений существует путь из вершины 1 в вершину 6, это означает, что цепь 1 должна быть размещена выше цепи 6 для того, чтобы не было наложений вертикальных сегментов на 1 и 6 контактах (см. рис. 6.33). Для того, чтобы задача решалась в рамках классической постановки, граф G_V должен быть ациклическим, в противном случае задача может быть решена

только с введением изломов, а это противоречит условиям классической постановки задачи канальной трассировки.

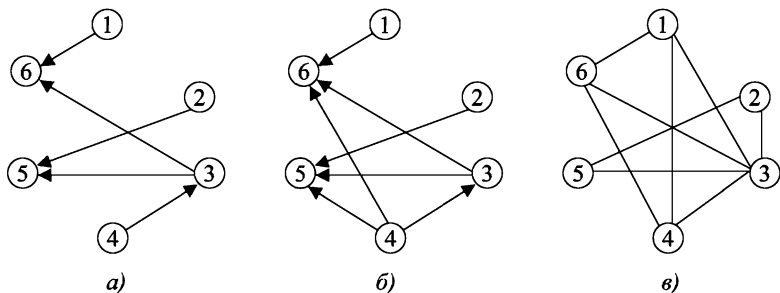


Рис. 6.34. Графы ограничений

Будем использовать расширенный граф вертикальных ограничений (рис. 6.34, б) $G_{RV} = (E_{Net}, E_{RV})$, где Net — множество вершин, соответствующих множеству цепей Net , E_{RV} — множество направленных ребер. Ребро $(n, m) \in E_{RV}$ существует тогда и только тогда, когда в графе вертикальных ограничений существует путь из вершины n в вершину m . Например, на рис. 6.34 в графе вертикальных ограничений есть путь из вершины 4 в вершину 5 через вершину 3. Следовательно, в расширенном графе вертикальных ограничений существует ребро (4,5).

Горизонтальные ограничения представлены неориентированным графом горизонтальных ограничений (рис. 6.34, в) $G_H = (E_{Net}, E_H)$, где E_{Net} — множество цепей, E_H — множество ребер. Ребро $(n, m) \in E_H$ существует тогда и только тогда, когда магистрали для n и m разные для исключения наложения горизонтальных сегментов n и m . Например, на рис. 6.33 в графе горизонтальных ограничений существует ребро (1,4); это означает, что цепь 1 не может размещаться на одной магистрали с цепью 4 (см. рис. 6.33).

Будем считать, что в ГА трассировки фенотип — это топология расположения цепей на кристалле, генотип — генетическая схема кодирования топологии, хромосома — кодированное представление одного варианта топологии, ген — элемент хромосомы, задающий некоторый фрагмент топологии.

Для ГА трассировки принята следующая схема:

1. Определение размера популяции N_p , числа поколений N_G , вероятности скрещивания $P(OK)$ и вероятности мутации $P(OM)$.
2. Задание случайным образом начальной популяции P^0 размером N_p .
3. $t = 0$.
4. Выбор случайным образом M пар хромосом из популяции P^t и применение операции скрещивания к каждой паре с заданной вероятностью $P(OK)$.
5. Применение операции мутации к каждой хромосоме популяции P^t с заданной вероятностью $P(OM)$.
6. Отбор. Отбор M хромосом с наилучшим значением целевой функции

из получившейся популяции P^t в новую популяцию P^{t+1} .

7. $t = t + 1$.

8. Если $t < N_G$, то переход к шагу 4.

9. Вывод хромосомы с наилучшим значением целевой функции.

10. Конец работы алгоритма.

Здесь используется представление хромосомы, основанное на топологическом описании канала, т. е. хромосома описывает взаимное расположение цепей. Для этого каждой паре цепей (m, n) , где $m \neq n$, ставится в соответствие ген, который может принимать три состояния: 0 — если цепь m должна располагаться выше n ; 1 — если цепь m должна располагаться ниже n ; и * — если их взаимное расположение не имеет значения или определяется из взаимного расположения остальных цепей (это происходит, если цепи не имеют горизонтальных ограничений). Для примера, изображенного на рис. 6.33, хромосома будет иметь следующий вид:

Таблица 6.3

Цепь m	1	1	1	1	1	2	2	2	2	3	3	3	4	4	5
Цепь n	2	3	4	5	6	3	4	5	6	4	5	6	5	6	6
Ген	*	0	0	*	0	0	*	0	*	1	0	0	*	0	*

где в строке таблицы 6.3, обозначенной как «Ген», записано значение гена в хромосоме, задающей расположение, показанное на рис. 6.33. Очевидно, что длина хромосомы при таком кодировании будет равна

$$L = \sum_{i=0}^{N-1} i,$$

где N — число цепей.

Как видно, длина хромосомы получается достаточно большой (для примера на рис. 6.33 она равна 15), что не приемлемо. Поэтому для уменьшения длины хромосомы используется анализ расширенного графа вертикальных ограничений и графа горизонтальных ограничений. Это получается за счет того, что взаимное расположения некоторых пар цепей уже зафиксировано в расширенном графе вертикальных ограничений, и изменение этого расположения приводит к образованию нарушений в канале, т. е. к наложению вертикальных сегментов цепей, что ведет к образованию нереальной хромосомы (т. е. такой, из которой не может быть получено решение, удовлетворяющее заданным требованиям).

Второй прием, позволяющий уменьшить длину хромосомы, основывается на том, что если цепи не имеют горизонтальных ограничений, то их взаимное положение либо не важно, либо определяется из соотношений с остальными цепями. Такие соотношения отмечены знаком «*». Следовательно, длина хромосомы будет равна $L = NGO - NRVO$, где NGO — число горизонтальных ограничений, $NRVO$ — число вертикальных ограничений в расширенном графе вертикальных ограничений. В примере для пар цепей (1,6), (2,5), (3,4), (3,5), (3,6), (4,6) взаимное расположение жестко задано расширенным графом вертикальных ограничений, а взаимное

расположение цепей в парах (1,2), (1,5), (2,4), (2,6), (4,5), (5,6) не имеет значения, поскольку цепи в этих парах не конфликтуют по горизонтали. Поэтому в примере хромосома будет выглядеть следующим образом:

Таблица 6.4

Цепь m	1	1	2
Цепь n	3	4	3
Ген	0	0	0

Длина этой хромосомы будет равна 3. Для получения из хромосомы эскиза канала с разведенными цепями используется граф топологии. Граф топологии — это ориентированный граф $G_T = (\text{Net}, E_T)$, где Net — множество цепей, E_T — множество направленных ребер, описывающих взаимное расположение цепей в канале. Ребро $(m, n) \in E_T$ существует тогда и только тогда, когда цепь m должна быть расположена в канале выше цепи n , т. е. на магистрали с меньшим номером. Граф топологии строится из хромосомы по следующему алгоритму:

1. Преобразуем расширенный граф вертикальных ограничений G_{RV} в граф топологии G_T .
 2. $i = 1$.
 3. Если ребро (m_i, n_i) не принадлежит G_T , то это ребро добавляется к G_T .
 4. Проверяем для всех вершин k существование пути из вершины m_i к вершине k при условии, что $k = 1, \dots, N$, $k \neq m_i$ и $k \neq n_i$ и не существует ребра (m_i, k) в G_T . Существование пути из вершины m_i к вершине k . Если такой путь существует, то добавляем ребро (m_i, k) к G_T .
 5. Если $i \leq L$, то увеличиваем i на 1 и переходим к шагу 3.
 6. Считаем построение графа топологии завершенным.
- Канал восстанавливается из хромосомы следующим образом:

1. Строим по хромосоме граф топологии G_T .
2. $i = 0$.
3. Находим вершины, у которых есть только исходящие дуги. Размещаем их на магистрали с номером i или на магистрали с меньшим номером, если это возможно, и удаляем эти вершины из графа.
4. $i = i + 1$.
5. Если граф топологии G_T не пустой, возвращаемся к шагу 3.
6. Не нарушая взаимного расположения, смещаем цепи по магистралям так, чтобы минимизировать длину вертикальных сегментов цепей.
7. Возвращаем полученный образец размещения цепей в канале шириной i .

Например, пусть задана хромосома $A=000$. На рисунке 6.33 показан полученный образец трассировки, который является оптимальным решением для примера. Если $A = 010$ (отличие от предыдущего решения заключается в том, что цепь 4 располагается выше цепи 1), то эскиз канала с разведенными цепями для такой хромосомы показан на рисунке 6.35.

В нашем случае задача заключается в минимизации целевой функции. Принимая во внимание характеристики задачи канальной трассировки, ЦФ определяется следующим образом:

$$F(A) = (\text{UsedTrack}(A) + 2) \cdot C + \text{TotalVertSeg}(A),$$

где C — число контактов, программная функция $\text{UsedTrack}(A)$ определяет число магистралей, занимаемых каналом, полученным при восстановлении из хромосомы A , а программная функция $\text{TotalVertSeg}(A)$ определяет длину вертикальных сегментов цепей в полученном решении. Длина вертикальных сегментов цепи определяется как расстояние между контактами и переходными отверстиями, которые соединяют вертикальные и горизонтальные сегменты. Например, для канала на рисунке 6.33 число занятых магистралей равно 4, длина вертикальных сегментов равна 22. Таким образом, целевая функция хромосомы будет равна $F(A) = (4 + 2) \cdot 8 + 22 = 70$, а для канала, показанного на рисунке 6.35, $F(A) = 72$.

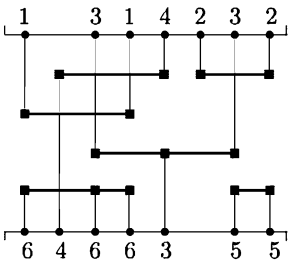


Рис. 6.35. Эскиз канала с разведенными цепями (первое альтернативное решение)

Данная методика определения целевой функции направлена, в первую очередь, на минимизацию ширины канала и, во вторую, — на минимизацию суммарной длины соединений.

Рассмотрим пример реализации описанного оператора кроссинговера:

1. $i = 0$.
2. Выбрать хромосому с лучшим значением ЦФ как первого родителя.
3. Выбрать произвольно второго родителя.
4. Сгенерировать случайное число rnd в интервале $[0,1]$.
5. Если $rnd < P(\text{ОК})$, то применить ОК к выбранным хромосомам и добавить получившихся потомков к популяции.
6. $i = i + 1$.
7. Если $i < M$, то перейти к шагу 2.
8. Конец работы алгоритма.

В рассматриваемом алгоритме после анализа хромосом выбран стандартный двухточечный ОК. Первая и вторая точка разреза в двухточечном ОК выбираются случайно. Часть первого родителя перед первой и после второй точки разреза копируется в аналогичные места в потомке. Место между первой и второй точкой разреза во втором родителе копируется в первого потомка. Второй потомок строится аналогично. Рассмотрим пример реализации двухточечного ОК:

p_1	:	0		1		0	Значение ЦФ:72
p_2	:	1		0		1	Значение ЦФ:77
p_3	:	0		0		0	Значение ЦФ:70
p_4	:	1		1		1	Значение ЦФ:77

В данном случае получен потомок p_3 с лучшим значением ЦФ, чем значение ЦФ родителей. Топология трассировки, соответствующая родителю p_1 , показана на рисунке 6.35, и родителю p_2 — на рисунке 6.36. Топология трассировки, соответствующая потомку p_3 , показана на рисунке 6.33, а потомок p_4 после декодирования хромосомы аналогичен родителю p_2 . Далее выполняется ОМ. Он произвольно изменяет один ген выбранной хромосомы при помощи случайного изменения значения гена с вероятностью $P(OM)$, равной норме мутации. Алгоритм применения ОМ в ГА выглядит следующим образом:

1. $i = 0$.
2. Сгенерировать случайное число rnd в интервале $[0, 1]$.
3. Если $rnd < P(OM)$, то применить ОМ к i -й хромосоме популяции M .
4. $i = i + 1$.
5. Если $i < M$, то перейти к шагу 2.
6. Конец работы алгоритма.

Следующий пример показывает реализацию этого механизма: хромосома (до применения ОМ): 0 (1) 0 — значение ЦФ: 72; хромосома (после применения ОМ): 0 (0) 0 — значение ЦФ: 70. Здесь выбранная точка мутации показана в скобках. После применения ОМ в данном случае получается хромосома с лучшим значением ЦФ. На рисунке 6.35 показана топология канала, соответствующего хромосоме до применения ОМ, а на рисунке 6.33 — после применения ОМ.

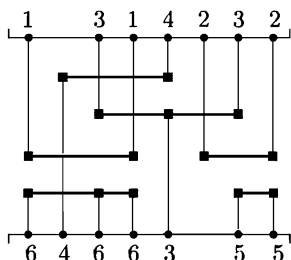


Рис. 6.36. Эскиз канала с введенными цепями (второе альтернативное решение)

Теоретическая временная сложность алгоритма составляет $N_G(N_p P(OK) 2) P(OM) \times O(N^2)$, где N_G — число поколений, N_p — размер популяции, $P(OK)$ — вероятность кроссинговера, $P(OM)$ — вероятность мутации, $O(N^2)$ — временная сложность декодирования хромосомы, N — число цепей.

Рассмотрим модифицированный генетический алгоритм двухслойной канальной трассировки цепей различной ширины. В работах [178–180] описаны алгоритмы канальной трассировки цепей различной ширины. Новые перспективные технологии проектирования топологии СБИС требуют трассировки цепей различной ширины и учета большого количества технологических ограничений. К ним относятся требования трассировки критических цепей (цепей частоты, питания) большего размера, чем у других цепей.

Алгоритм основан на проблемно-специфическом представлении схемы и проблемно-ориентированных генетических операторах и опирается на архитектуру поиска решений на основе миграции и искусственной селекции. В данном алгоритме каждая цепь, т.е. соединение эквипотенциальных контактов, представляется как ряд горизонтальных и вертикальных отрезков, имеющих различную ширину. Цепь может иметь несколько го-

Алгоритм основан на проблемно-специфическом представлении схемы и проблемно-ориентированных генетических операторах и опирается на архитектуру поиска решений на основе миграции и искусственной селекции. В данном алгоритме каждая цепь, т.е. соединение эквипотенциальных контактов, представляется как ряд горизонтальных и вертикальных отрезков, имеющих различную ширину. Цепь может иметь несколько го-

горизонтальных и вертикальных сегментов. Выводы, принадлежащие одной цепи, должны быть соединены в соответствии с налагаемыми ограничениями. Эта постановка не является классической, когда каждая цепь имеет только один горизонтальный сегмент и несколько вертикальных сегментов, соединяющих горизонтальный сегмент с контактами цепи.

Для проведения трассировки доступны два слоя. Возможны два варианта назначения слоев:

- в одном слое располагаются только вертикальные сегменты цепи, в другом — горизонтальные (VH-топология);
- в обоих слоях возможно любое ортогональное направление (как горизонтальное, так и вертикальное).

Соединения между горизонтальными и вертикальными отрезками делаются через переходные отверстия. Им является квадрат со стороной, равной ширине цепи, проходящей через это переходное отверстие (рис. 6.37). Не допускаются пересечения различных цепей друг с другом в одном слое.

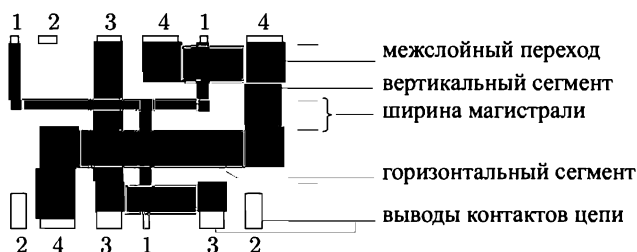


Рис. 6.37. Топология канала

Хромосома в данном алгоритме состоит из двух матриц (для верхнего и нижнего слоев), как показано на рис. 6.38, где $g_{i,j}$ и $d_{i,j}$ — гены хромосомы, $i = 1 \div m$, $j = 1 \div n$. Число m соответствует числу магистралей в данном канале, n — числу выводов в канале. Каждый ген хромосомы представляет в топологии канала прямоугольник со сторонами $Vxrom[0][i]$ и $Vxrom[1][j]$ (горизонтальные и вертикальные максимумы).

$$R_1 = \begin{bmatrix} g_{11} & g_{12} & g_{13} & \cdots & g_{1n} \\ g_{21} & g_{22} & g_{23} & \cdots & g_{2n} \\ g_{31} & g_{32} & g_{33} & \cdots & g_{3n} \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ g_{m1} & g_{m2} & g_{m3} & \cdots & g_{mn} \end{bmatrix} \quad R_2 = \begin{bmatrix} d_{11} & d_{12} & d_{13} & \cdots & d_{1n} \\ d_{21} & d_{22} & d_{23} & \cdots & d_{2n} \\ d_{31} & d_{32} & d_{33} & \cdots & d_{3n} \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ d_{m1} & d_{m2} & d_{m3} & \cdots & d_{mn} \end{bmatrix}$$

Рис. 6.38. Структура хромосомы

Возможны три значения гена в хромосоме:

- $g_{i,j} > 0$ — в данном прямоугольнике проходит цепь с номером $g_{i,j}$;
- $g_{i,j} = 0$ — данный прямоугольник пустой;
- $g_{i,j} < 0$ — в данном прямоугольнике находится межслойный переход, цепи с номером $|g_{i,j}|$

Первая строка матрицы R_1 соответствует верхним контактам канала, последняя строка матрицы R_1 соответствует нижнему набору контактов.

Рассмотрим на примере декодирование хромосомы (рис. 6.39).

$$R_1 = \begin{bmatrix} 0 & 0 & 0 & -4 & 4 & -4 \\ -1 & 1 & 1 & -1 & -1 & 0 \\ 0 & -4 & 4 & 4 & 4 & -4 \\ 0 & 0 & -3 & 3 & -3 & 0 \end{bmatrix} \quad R_2 = \begin{bmatrix} 1 & 2 & 3 & -4 & 1 & -4 \\ -1 & 0 & 3 & -1 & -1 & 4 \\ 0 & -4 & 3 & 1 & 0 & -4 \\ 2 & 4 & -3 & 1 & -3 & 2 \end{bmatrix}$$

Рис. 6.39. Пример хромосомы

Декодируем матрицы хромосомы R_1 и R_2 . Получаем топологию канала в двух слоях (рис. 6.40).

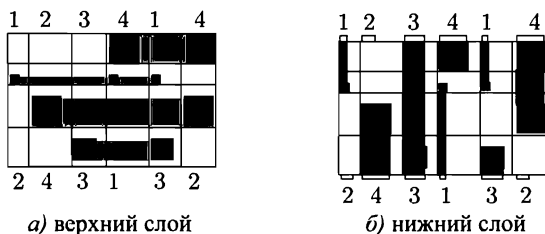


Рис. 6.40. Топология верхнего и нижнего слоев канала

а) верхний слой б) нижний слой

ГА канальной трассировки может быть описан следующим кортежем: $\langle P', D, Q \rangle$, где P' — является множеством всех решений для данной задачи, D — ограничения, накладываемые на множество P' для получения допустимых решений, и Q — целевая функция, с помощью которой можно определить наилучшее (оптимальное) решение. Для задачи канальной трассировки этот вектор можно интерпретировать следующим способом: P' — множество всех решений (хромосом) из всех популяций, причем эти хромосомы могут также представлять собой недопустимые решения или решения, являющиеся не 100-процентной трассировкой канала.

Пусть P^t — некоторая популяция на шаге t , тогда $P' = \cup P^t$, $t = 1 \div N_G$; $P^t = \{p_j\}$, $j = 1 \div N_p$, где N_G — количество итераций алгоритма, N_p — количество хромосом в популяции P^t . Ограничения D : цепи не могут пересекаться, горизонтальные сегменты цепей проводятся в одном слое, вертикальные в другом, т. е. накладываются ограничения $D(h_{i,j})$ на множество $P'(h_{i,j})$ такие, что в множестве допустимых решений $S \subseteq P'$ не могут существовать такие хромосомы p_n , гены которых $g_{i,j}$ имеют номера двух цепей одновременно (т. е. пересекаются), или значения нескольких горизонтальных (вертикальных) генов $\dots g_{i,j-1} \cdot g_{i,j} \cdot g_{i,j+1} \dots$ при $i = \text{const}$ ($\dots g_{i-1,j} \cdot g_{i,j} \cdot g_{i+1,j} \dots$ при $j = \text{const}$) имеют одинаковый номер цепи в нижнем (верхнем) слое (т. е. ограничение на недопустимость проведения горизонтальных участков цепей в нижнем слое, а вертикальных участков — в верхнем).

Целевая функция Q вычисляется в зависимости от суммарной ширины цепей в канале, количества межслойных переходов и общей длины цепей в канале.

Габаритные размеры канала определяются:

$$Q_1^1 = \sum_j^n \text{Brxrom}[0][j], \quad Q_1^2 = \sum_j^m \text{Brxrom}[1][j];$$

число межслойных переходов вычисляется:

$$Q_2 = \sum_j^n \sum_i^m \text{BrViv}[|g_{ji}|], \quad \text{если } g_{ji} < 0,$$

а суммарная длина цепей имеет вид:

$$Q_3 = \sum_j^n \sum_i^m \text{BrViv}[|g_{ji}|], \quad \text{если } g_{ji} > 0,$$

здесь n — количество столбцов в канале (т. е. ширина канала), m — количество строк в канале (т. е. высота канала), BrViv — массив ширины цепей, $g_{i,j}$ — текущий ген в хромосоме.

Общая ЦФ вычисляется суммированием

$$Q = k_1 \cdot Q_1 + k_2 \cdot Q_2 + k_3 \cdot Q_3,$$

где $k_1 - k_3$ — коэффициенты при локальных критериях, с помощью которых можно регулировать влияние того или иного критерия на качество решения. Они принимают значения в интервале $(0,1)$.

Процесс оптимизация сводится к минимизации критерия Q , т. е. $Q(P') \rightarrow \min Q(h_{\text{опт}}) = \min Q(h_{ij})$, где $h_{ij} \subset S$.

Для того, чтобы определить ширину и высоту прямоугольного участка топологии канала соответствующего гена, используются два массива горизонтальных и вертикальных максимумов. Они вычисляются для каждой конкретной хромосомы и используются в дальнейшем для определения ширины и высоты части проводника цепи (рис. 6.41). В массивах хранятся целые положительные числа, которые определяют максимальный размер вертикали или горизонтали в хромосоме (канале). Для того, чтобы вычислить конкретные размеры ячейки, т. е. гена, необходимо знать номер столбца и строки и затем, с помощью соответствующих значений из массива, определяем ширину и высоту ячейки.

Зная размеры каждой ячейки и структуру хромосомы, можно построить геометрическую топологию

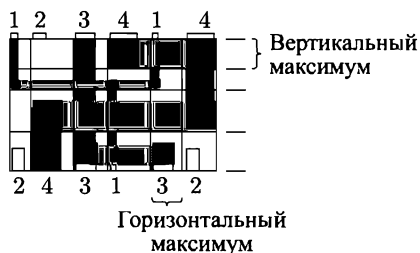


Рис. 6.41. Решение, полученное из хромосомы, с горизонтальными и вертикальными максимумами

канала. Хромосома служит для отображения канала без учета ширины цепей. Таким образом, задав ширину, одинаковую для всех цепей, можно получить частный случай — канал без учета ширины цепей. Декодируя хромосомы, вместе с расшифровкой каждого гена в обоих слоях получим канал с различной шириной цепей (рис. 6.41).

Алгоритм формирования массивов горизонтальных и вертикальных максимумов (для массива вертикальных максимумов) представлен на рис. 6.42.

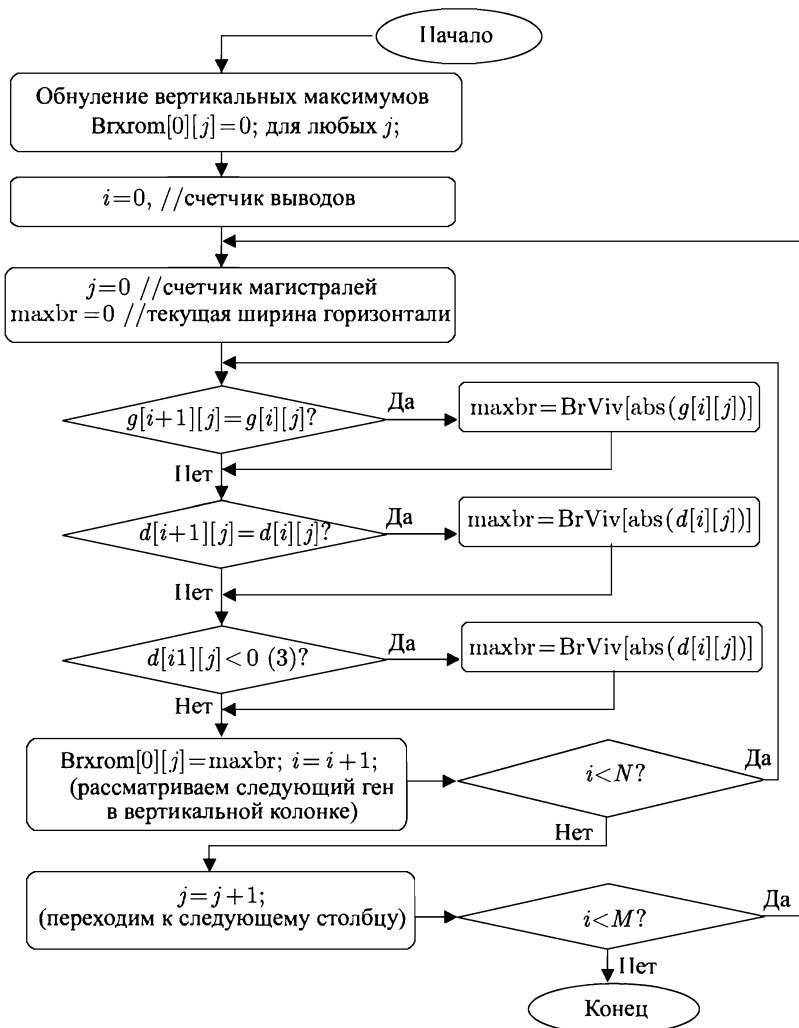


Рис. 6.42. Структурная схема формирования горизонтальных и вертикальных максимумов

Поясним структурную схему алгоритма:

1. Обнуляется массив вертикальных максимумов $Vrxrom[0]$ (для горизонтальных максимумов — $Vrxrom[1][\]$).
2. Начинается цикл по столбцам, т.е. рассматривается сначала первый столбец, затем второй и так далее, для каждого столбца присваивается максимальная ширина $maxbr$, которая определяется на шаге 3.
3. Цикл по магистралям. Просматривается весь столбец и находится цепь с самой большой шириной (из массива ширины цепей $BrViv[\]$ в нижнем слое (1) или верхнем слое (2), причем если это будет межслойный переход, его тоже надо учитывать (3)).
4. Переход к пункту 2 для рассмотрения следующего столбца до тех пор, пока не будут рассмотрены все столбцы и не сформирован массив вертикальных максимумов $Vrxrom[0]$.

Рассмотрим модифицированный оператор кроссинговера, ориентированный на задачу трассировки. Два родителя выбираются независимо друг от друга. Пусть, например, p_α и p_β — копии родителей (рис. 6.43); p_γ — полученный в результате применения ОК потомок.

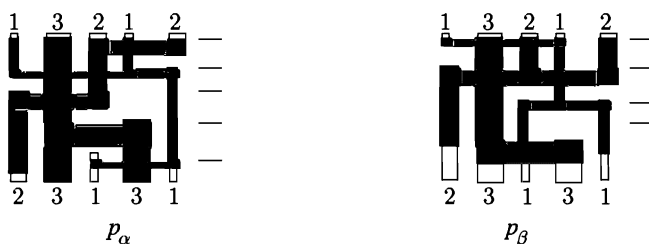


Рис. 6.43. Хромосомы–родители

Сначала произвольно выбирается точка кроссинговера или, другими словами, продольное сечение x_c между двумя какими-либо выводами, причем $1 \leq x_c \leq x_{ind}$, где x_{ind} — число возможных сечений канала. Затем в p_γ из индивида p_α наследуется трассировочная структура, удовлетворяющая следующим двум условиям:

- расположена на участке (x_α, y_α, z) при $1 \leq x_\alpha \leq x_c$, $1 \leq y_\alpha \leq y_{ind}$ (y_{ind} — число магистралей), $1 \leq z \leq 2$ (z — номер слоя);
- не пересекает сечение x_c .

Аналогичным образом наследуется трассировочная структура из индивида p_β . Связи родителей p_α и p_β , пересеченные сечением x_c , просматриваются, пока их точки Штейнера или выводы не будут достигнуты (рис. 6.44). Оставшиеся участки цепей наследуются в p_γ (рис. 6.45). Если трассировочная структура p_α (или p_β), которая должна быть передана в p_γ , содержит магистрали $y_{ind\alpha}$ ($y_{ind\beta}$), не занятые горизонтальными отрезками или точками Штейнера, то они удаляются. Начальное число магистралей в p_γ есть $y_{ind\gamma} = \max(y_{ind\alpha}, y_{ind\beta})$. Часть индивида-родителя, содержащая

меньше магистралей, чем p_γ , расширяется добавлением магистрали в произвольную позицию перед преобразованием трассировочной структуры p_γ .

Трассировка оставшихся несоединенных в p_γ цепей осуществляется следующим образом: пусть N_α — множество всех точек Штейнера и выводов p_α (концевые точки сегмента); соответственно N_β — множество аналогичных точек p_β . Если N_α содержит более, чем один вывод цепи, то эти точки (если это возможно) соединяются друг с другом в произвольном порядке на основе стратегии случайной трассировки, рассмотренной выше. Соединенные точки цепи из списка N_α удаляются, за исключением одной произвольно выбранной. Аналогичная «внутренняя» трассировка производится и с элементами N_β . Далее точки из N_α выбираются произвольно и сравниваются с точками из N_β . Если точка той же цепи найдена в N_β , то они соединяются произвольной случайной трассой. Процесс продолжается, пока все цепи канала не будут соединены.

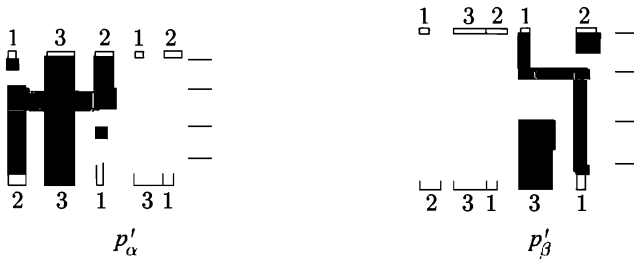


Рис. 6.44. Преобразованные хромосомы

Часто возникает ситуация, когда произвольная трассировка между двумя точками не приводит к связям после проведения i -числа итераций заданных заранее, тогда канал расширяется в произвольной позиции y_{add} при условии $1 \leq y_{\text{add}} \leq y_{\text{ind}_\gamma}$. Если j расширений канала (j — начальное число магистралей p_γ) не обеспечивают соединение, то потомок p_γ удаляется и процесс продолжается снова с выбора местонахождения продольного сечения родительских хромосом.

В приведенном примере одно из возможных решений представлено на рис. 6.45 (потребовалось одно расширение канала).

Рис. 6.45. Полученный потомок после ОК

Отметим, что алгоритм основан на проблемно-специфическом представлении схемы и использует специально разработанные проблемно-ориентированные генетические операторы. Для избежания попадания в локальные оптимумы включена компонента управления генетическим поиском. Она отслеживает неблагоприятные изменения в популяции, тем самым препятствуя ее вырождению. Алгоритм имеет квадратичную зависимость $O(n^2)$ времени решения от числа выводов (цепей) канала.

6.6. Решение задач о коммивояжере методами моделирования эволюций

Задача о коммивояжере в английской интерпретации TSP является классической NP-полной задачей [90, 161, 182]. Она заключается в нахождении кратчайшего гамильтонова цикла в графе. Задача о коммивояжере нашла широкое применение в инженерных приложениях. Без каких-либо изменений в постановке, она используется для проектирования разводки коммуникаций, разработки архитектуры вычислительных сетей и др. Кроме того, она имеет теоретическую ценность, являясь асимптотической оценкой исследования различных эвристических алгоритмов, которые могут быть затем применены для решения более сложных задач комбинаторной оптимизации, например, квадратичной задачи о назначениях, частным случаем которой является задача о коммивояжере.

Рассмотрим постановку задачи. В неформальном виде задача о коммивояжере трактуется следующим образом: коммивояжеру необходимо

посетить M городов, не заезжая в один и тот же город дважды, и вернуться в исходный пункт по маршруту с минимальной стоимостью. Более строго имеем: дан граф $G = (X, U)$, где $|X| = n$ — множество вершин (городов), $|U| = m$ — множество ребер (возможные пути между городами), показанный на рис. 6.46. Дана матрица чисел $R(i, j)$, где $i, j \in 1, 2, \dots, n$, представляющих собой стоимость переезда из вершины x_i в x_j .

Требуется найти перестановку из элементов множества X , такую что $\{CF$:

$$\text{Fitness}(\varphi) \equiv R(\varphi(1), \varphi(n)) + \sum \{R((i), \varphi(i+1))\} \rightarrow \min.$$

Если граф не является полным, то дополнительными ограничениями являются:

$$\langle \varphi(i), \varphi(i+1) \rangle \in U \quad \forall i \in 1, 2, \dots, n-1 \quad \text{и} \quad \langle \varphi(1), \varphi(n) \rangle \in U.$$

Данные ограничения можно учесть, например, проставив большие числа в матрицу R в ячейках, соответствующих отсутствующим ребрам в графе. Поэтому в дальнейшем для решения задачи о коммивояжере будем рассматривать полные графы с весами на ребрах.

Запишем матрицу смежности графа $G = (X, U)$, где $|X| = 7$, $|U| = 21$ (рис. 6.46). Для пути $\varphi(1) = \langle 1, 2, 3, 4, 5, 6, 7 \rangle$ (хромосома p_1) имеем ЦФ: $\text{Fitness}(\varphi(1)) = 1+1+1+1+1+1+1 = 7$, а для пути $\varphi(2) = \langle 7, 2, 5, 4, 3, 6, 1 \rangle$ (хромосома p_2), имеем ЦФ: $\text{Fitness}(\varphi(2)) = 3+6+1+1+5+3+1 = 20$. Следовательно, маршрут $\varphi(1)$ предпочтительнее, чем $\varphi(2)$, с точки зрения

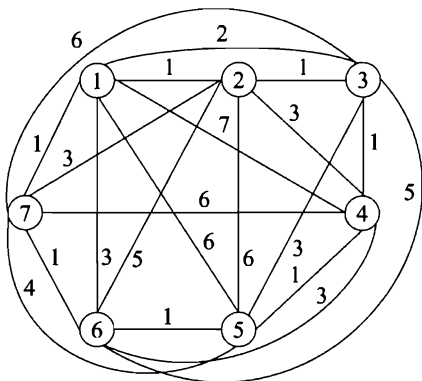


Рис. 6.46. Граф $G = (X, U)$

заданной ЦФ. Отметим, что значение ЦФ $\text{Fitness}(\varphi)$ не зависит в частном случае от выбора вершины — начала маршрута. Например, $\text{Fitness}(\varphi(3)) = \text{Fitness}(\varphi(1)) = 7$, где $\varphi(3) = \langle 2, 3, 4, 5, 6, 7, 1 \rangle$ (хромосома p_3), являясь инвариантом для всех возможных циклических сдвигов и, следовательно, существуют n равнозначных маршрутов:

$$R = \begin{vmatrix} & 1 & 2 & 3 & 4 & 5 & 6 & 7 \\ 1 & 0 & 1 & 2 & 7 & 6 & 3 & 1 \\ 2 & 1 & 0 & 1 & 3 & 6 & 5 & 3 \\ 3 & 2 & 1 & 0 & 1 & 3 & 5 & 6 \\ 4 & 7 & 3 & 1 & 0 & 1 & 3 & 6 \\ 5 & 6 & 6 & 3 & 1 & 0 & 1 & 4 \\ 6 & 3 & 5 & 5 & 3 & 1 & 0 & 1 \\ 7 & 1 & 3 & 6 & 6 & 4 & 1 & 0 \end{vmatrix}.$$

Фиксация вершины-старта может быть использована для сужения пространства поиска, которое в этом случае равно $(n - 1)!$. Поэтому в общем случае задача о коммивояжере чувствительна к выбору начальных условий. В этой связи для создания структуры алгоритма эффективно используются методы ЭМ, описанные в предыдущих разделах.

NP-полнота задачи о коммивояжере подразумевает, что поиск ведется в пространстве, растущем от n экспоненциально. Соответственно, время работы алгоритмов, основанных на полном переборе или на построении дерева решений (методы ветвей и границ, золотого сечения и др.), растет экспоненциально в зависимости от числа вершин графа. Эти алгоритмы способны решать задачу для малого числа вершин ($n \approx 20$) за полиномиальное время. Поэтому для решения сложных задач о коммивояжере используются различные эвристики.

Достоинством эвристических алгоритмов является их способность находить оптимальные или близкие к ним решения при $n \geq 100$ за полиномиальное время вычислений. Недостатком является то, что найденное наилучшее решение в общем случае не является оптимальным. В практических задачах для графов большой размерности ($n \geq 1000$) упомянутый недостаток эвристических алгоритмов не является существенным, так как находимые таким путем решения обычно отличаются от оптимальных лишь на доли процента. Соответственно, двумя основными критериями при решении задачи о коммивояжере, оценивающими эвристический алгоритм, являются скорость сходимости и близость получаемого решения к оптимальному (если оно известно) или же к лучшим стандартным решениям (бенчмаркам), полученным с помощью других эвристик (если оптимум не известен).

Кратко рассмотрим классификацию алгоритмов, применяемых для решения задачи о коммивояжере. Эвристические алгоритмы могут быть условно разбиты на два класса — класс, работающий только с одним решением (например, моделирование отжига) и класс, работающий с множеством (популяцией) решений (например, ГА и ЭМ).

Первый класс алгоритмов обычно имеет более высокую скорость сходимости, но в общем случае неспособен к выходу из многих локальных оптимумов. Второй класс эвристических алгоритмов осуществляет параллельный поиск в нескольких точках пространства решений. Алгоритмы этого класса имеют значительно больше шансов найти оптимальное решение, но параллельный поиск обычно имеет временную сложность большего порядка. Наиболее известными представителями этого класса являются ГА с жадными стратегиями и с последовательным случайным или направленным преобразованием специальных матриц [182, 183]. Жадные стратегии применяются иерархически: на уровне алгоритма, на уровне блока, на уровне генетического оператора.

Опишем эволюционные стратегии, которые являются комбинацией ГА, эволюционного программирования, эволюций Ч. Дарвина, Ж. Ламарка, де Фриза и К. Поппера, а также методов локального поиска.

Рассмотрим характерные для задачи о коммивояжере генетические операторы. Существуют два основных генетических представления возможного решения: кодирование решения в виде списка вершин (последовательность посещаемых городов) или кодирование решения как списка ребер, образующих гамильтонов цикл (список использованных путей из города в город).

Первый способ кодирования более прост. При таком кодировании в позиции i -й хромосомы находится i -й посещенный город (i -я вершина графа). Для графа G (рис. 6.46) приведем два возможных решения при таком методе кодирования:

1	2	3	4	5	6	7		1	2	3	4	5	6	7
1	4	2	3	5	7	6		6	4	2	3	1	5	7

Здесь в первой строке записаны номера вершин графа, а во второй строке — случайная подстановка, определяющая одно из возможных решений. Одной из характерных особенностей задачи о коммивояжере, как, впрочем, и большого ряда других задач, является ограничение на возможные комбинации значений в хромосоме. В этом случае имеем: один и тот же город не может быть посещен дважды и все города должны быть посещены. Таким образом, если применить ПГА с обычными одноточечным или двухточечным ОК, то в общем случае полученные решения (потомки) будут, скорее всего, нереальными. Например, для графа G (рис. 6.46) возьмем из популяции $P = \{p_1, p_2, p_3, p_4, p_5\}$ две хромосомы (решения задачи о коммивояжере) p_4, p_5 :

p_4 : 1 4 2 3 5 7 6
 p_5 : 6 4 2 3 1 5 7.

Применим простой ОК. Пусть точка ОК находится между третьим и четвертым геном в хромосоме. Тогда получим следующих двух потомков:

p_6 : 1 4 2 | 3 1 5 7
 p_7 : 6 4 2 | 3 5 7 6.

Очевидно, что решения p_6 , p_7 не являются допустимыми, так как в p_6 повторяется ген 1 и отсутствует ген 6, а в p_7 повторяется ген 6 и отсутствует ген 1, т. е. в решении задачи о коммивояжере содержатся города, посещаемые дважды и не посещаемые ни разу.

Интерес, представляет матричное кодирование задачи о коммивояжере. Например, для хромосомы p_4 матрица запишется:

$$R(p_4) = \begin{array}{c|ccccccc} & 1 & 2 & 3 & 4 & 5 & 6 & 7 \\ \hline 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 2 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 3 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 4 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 5 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 6 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 7 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{array}.$$

Как видно из матрицы, единицами кодируются переходы между генами в хромосоме. Для хромосомы p_5 матрица задачи о коммивояжере примет вид:

$$R(p_5) = \begin{array}{c|ccccccc} & 1 & 2 & 3 & 4 & 5 & 6 & 7 \\ \hline 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 2 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 3 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 4 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 5 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 6 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 7 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{array}.$$

Тогда популяция хромосом будет состоять из набора аналогичных матриц. Такое кодирование в виду большого числа нулей избыточное, хотя является наглядным и удобным для преобразований. Отметим, что применение стандартных генетических операторов для такой кодировки является неэффективным в виду большого количества получаемых несуществующих потомков. Предложим вместо матриц использовать списки связности. Это позволит получить набор строительных блоков. На данном наборе на основе оператора сегрегации и оператора инверсии можно эффективно получать новых допустимых потомков. Например, на основе $R(p_4)$ запишем набор строительных блоков: 1–4; 2–3; 3–5; 4–2; 5–7; 7–6; 6–1. Применив операторы сегрегации и инверсии, получим новую допустимую хромосому p_8 : 4 1 7 6 5 3 2 с ЦФ: $\text{Fitness}(\varphi(8)) = 20$.

Основное преимущество представления задачи о коммивояжере в виде матрицы смежности то, что оно дает возможность анализировать шаблоны, которые применимы для n -размерной оптимизации ГА. Шаблоны, определенные в терминах единственной определенной позиции, соответствуют естественным строительным блокам, т. е. элементам. Например, шаблон (# # # # # 7 #) есть множество всех изменений, в котором элемент 6,

соответствующий гену 7, имеет место, а остальные позиции шаблона могут быть определены случайным образом. Обобщим основные моменты анализа шаблонов ГА в отсутствии оператора репродукции и ОМ. Генерация для всех представленных шаблонов в популяции определится по формуле:

$$M(H, t + 1) \approx M(H, t)(u(H, t)/u(P, t)),$$

где $M(H, t)$ — количество представителей шаблонов H за время t ; $u(H, t)$ — производительность шаблонов H за время t ; $u(P, t)$ — средняя производительность популяции за время t . Элементы любой части шаблона конкурируют против других элементов этой части. Это приводит к сокращению размерности пространства поиска и построению больших строительных блоков высокой производительности.

Для задачи о коммивояжере существуют специально разработанные ОК, позволяющие гарантированно получать из родителей допустимых потомков. Основными из них являются: циклический ОК, универсальный ОК, частично соответствующий ОК и жадный ОК, которые описаны выше.

Жадные стратегии применяются, когда искомый объект строится по частям (в нашем случае из строительных блоков). Жадная стратегия делает на каждом шаге «локально оптимальный» выбор. Алгоритмы на основе жадных стратегий работают быстрее, чем алгоритмы динамического программирования. Однако жадная стратегия не всегда дает оптимальное решение. Задача обладает свойством оптимальности для подзадачи, если оптимальное решение задачи содержит оптимальные решения ее подзадач.

Существуют две особенности жадной стратегии: принцип жадного выбора, оптимальность для подзадач. Говорят, что к оптимизационной задаче применим принцип жадного выбора, если последовательность локально оптимальных жадных выборов дает глобальное решение. На каждом шаге жадная стратегия делает наилучший выбор, а потом уже пытается сделать наилучший выбор среди оставшихся решений, какие бы они ни были.

Приведем алгоритм построения модифицированного жадного ОК для задачи о коммивояжере:

1. Для каждой пары хромосом случайным образом выберем точку жадного ОК и в качестве номера стартовой вершины графа возьмем номер отмеченного гена в хромосоме.
2. Сравним частичную стоимость путей, ведущих из текущих вершин в хромосомах-родителях и выберем из них кратчайший.
3. Если выбранная таким образом вершина графа уже была включена в частичный путь, то взять случайную вершину из числа еще не отмеченных. Присвоить полученной таким образом вершине значение текущей.
4. При преждевременном образовании циклов выбирается другой кратчайший путь.
5. Повторяем пункты 2 и 3 пока не будет построен гамильтонов цикл с минимальной суммарной стоимостью ребер.
6. Конец работы алгоритма.

Решение-потомок в этом алгоритме формируется как последовательность вершин графа в том порядке, в котором они становились текущими.

В данном жадном ОК использовано знание о задаче, что выбор кратчайшего пути обычно предпочтителен. Однако подобное простое предположение не всегда приводит к оптимуму, поскольку иногда требуется пренебречь локальным выигрышем ради достижения глобальной цели. Жадные ОК, основанные на знаниях о задаче, улучшают скорость сходимости, однако часто препятствуют выходу из локальных минимумов. Стандартный жадный ОК легко устраняет все плохие изменения, получаемые ОМ. Такие решения приводят к неэффективному использованию популяции. Существует несколько путей избежать этой ситуации: изменить архитектуру ГА или же не помещать такие решения в популяцию. Могут быть предприняты следующие меры:

Эвристика 1 — это предпочтение более «невыгодного» маршрута более выгодному с определенной вероятностью;

Эвристика 2 — взятие около 50% генетического материала от каждого из родителей.

Когда путь оказывается тупиковым, производится случайный выбор вершины или же выбор ближайшей вершины из числа не рассмотренных. Он может быть задан ЛПР, блоком эволюционной адаптации или ЭС.

Опишем видоизмененный модифицированный жадный ОК:

1. Случайным образом выбрать стартовую вершину в первом родителе.
2. Текущий родитель — первый родитель.
3. Выбрать, какая из соседних вершин (соседними являются вершины, расположенные в хромосоме справа и слева от рассматриваемой) в анализируемом родителе ближе к текущей вершине. Ближайшая из соседних вершин, не вошедшая в путь, становится новой текущей вершиной. Если только одна из двух соседних вершин не вошла в путь, то она становится текущей. Если обе соседние вершины являются посещенными (тупиковая ситуация), то выбрать ближайшую из числа еще не посещенных. Тогда в качестве текущего родителя выбирается другой родитель.
4. Повторять пункт 3 пока все вершины не будут посещены. Потомок формируется как последовательность посещаемых вершин.
5. Конец работы алгоритма.

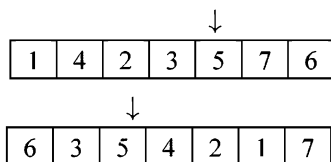
Покажем пример работы алгоритма для графа G (рис. 6.46). Пусть популяция состоит из двух хромосом (p_9, p_{10}):

p_9 : 1 4 2 3 5 7 6

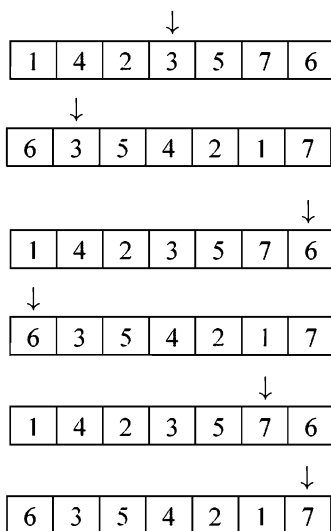
p_{10} : 6 3 5 4 2 1 7

Для p_9 ЦФ: $\text{Fitness}(\varphi(9)) = 22$, а для p_{10} ЦФ: $\text{Fitness}(\varphi(10)) = 15$. Далее стрелка над одной из вершин в каждом из родителей обозначает, что данная вершина является текущей. Жадный ОК с изменяемой вероятностью принятия плохих решений требует настройки этой вероятности, если это константа, или же определения специальной функции для ее ав-

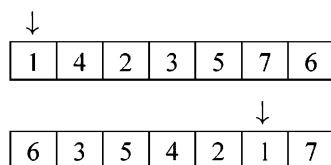
томатического изменения во время работы. В данном примере использован 50-процентный жадный ОК. На первом этапе вершина 5 выбрана стартовой и является текущей для обоих родителей. Согласно алгоритму следующей вершиной в путь выбирается вершина 3, соседняя 5, имеющая кратчайший путь (5–3).



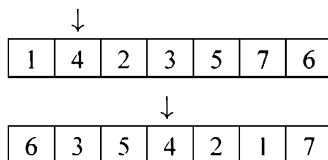
На следующем шаге в путь включается вершина 6, так как путь (5–3–6) имеет лучшее значение ЦФ, чем путь (5–3–7), после применения 50-процентного жадного ОК. Далее в путь включается вершина 7 и частичный путь задачи о коммивояжере имеет вид (5–3–6–7).



Согласно алгоритму для включения в путь выбирается вершина 1. Это происходит из-за того, что в правой половине первого родителя рассмотрены все вершины, а для второго родителя вершина 1 — соседняя с вершиной 7, и этот путь — кратчайший.



Далее в путь включается вершина 4, соседняя вершине 1 в первом текущем родителе. Тогда имеем путь (5–3–6–7–1–4).



Осталась не рассмотренной вершина 2, которая автоматически включается в путь задачи о коммивояжере. Тогда имеем хромосому потомок p_{11} : 5 3 6 7 1 4 2 с ЦФ: $\text{Fitness}(\varphi(11)) = 26$. В связи с тем, что потомок p_{11} имеет значение ЦФ худшее, чем у родителей p_9 , p_{10} , он в популяцию не включается. Снова в текущем родителе p_9 выбираем текущую вершину 7 и после нескольких шагов жадного ОК получаем хромосому-потомок p_{12} : 7 6 5 4 3 2 1 с ЦФ: $\text{Fitness}(\varphi(12)) = 7$, что соответствует глобальному оптимуму. Такие жадные ОК не всегда позволяют получать глобальные оптимумы, поэтому необходимо использовать множество разработанных генетических операторов и нестандартные архитектуры генетического поиска.

На рис. 6.47 приведена структурная схема алгоритма для решения задачи о коммивояжере. Данная схема аналогична схеме алгоритма решения задачи размещения. В отличие от нее предлагается следующее. Введен блок локального поиска, который позволяет получать локальные экстремумы. Все генетические операторы ориентированы на использование знаний о решаемой задаче, причем изменен порядок использования генетических операторов. В задаче о коммивояжере решающую роль играют ОК и его модификации, поэтому ОК в данной схеме примеряется первым.

Использование остальных генетических операторов может варьироваться в зависимости от конкретных особенностей задачи. Блок редукции составляет размер популяции постоянным для каждой генерации.

В схеме рис. 6.47 возможно уменьшение «жадности» ОК, что разнообразит популяцию, если есть добавление нового генетического материала с высокими ЦФ, производимого случайными факторами или же дополнительными эвристиками. После достижения локального оптимума в популяции наблюдается большое число одинаковых маршрутов. В этой связи может наступить преждевременная сходимость. Эта проблема тесно связана с потерей разнообразия в популяции. Одним из источников однообразия является появление «супер-хромосом», которые в течении многих поколений доминируют в ней. Имеется ряд путей частичного решения этой проблемы, например, изменение алгоритма селекции или же модификация генетического оператора рекомбинации.

Например, факт нахождения в локальном (глобальном) экстремуме может быть установлен, когда K первых хромосом в популяции имеют одинаковую длину маршрута. Это K называется элитным числом. Предполага-

Модифицированные генетические операторы — это полная или частичная рандомизация решения. Такие изменения в популяции могут разрушать хромосомы с высоким значением ЦФ. Однако в двух предложенных эвристиках лучшее решение в популяции остается неизменным.

Отметим, что такие генетические операторы и эвристики работают не все время, а только в ситуациях, когда однообразие в популяции достигает высокого уровня. Совместная эволюция Ч. Дарвина, Ж. Ламарка, де Фриза и К. Поппера регламентирует и управляет таким процессом генетического поиска.

Отметим, что в описанном алгоритме был использован набор методов селекции. В основном цикле ГА промежуточная популяция решений не создавалась, получаемый потомок замещал наихудшую хромосому в популяции. Следует отметить, что модифицированные генетические операторы инверсии и ОМ оказывают большое влияние на получение эффективных результатов. Отметим, что отличительной особенностью рассмотренного ГА является способность хорошо работать на популяциях с малым числом хромосом, что уменьшает сложность и временную сложность алгоритма. Подобное улучшение связано с высокой степенью использования миграции популяции в виде регулярных адаптаций, выполняемых рассмотренными выше генетическими операторами. Полученные результаты в генетическом решении задачи о коммивояжере позволяют говорить об эффективности метода и целесообразности использования аналогичных подходов в решении других задач оптимизации.

6.7. Задачи раскраски, построения клик и независимых множеств графов

Рассмотрим задачу раскраски графа. Раскраской вершин графа $G = (X, U)$ [89, 123, 124, 159] называется разбиение множества вершин X на L непересекающихся классов (подмножеств):

$$X_1, X_2, \dots, X_L; \quad X = \bigcup_{i=1}^L X_i; \quad X_i \cap X_j = \emptyset; \quad i, j = 1, 2, \dots, L,$$

таких, что внутри каждого подмножества X_i не должно содержаться смежных вершин. Если каждому подмножеству X_i поставить в соответствие определенный цвет, то вершины внутри этого подмножества можно окрасить в один цвет, а вершины другого подмножества X_j — в другой цвет и так далее до раскраски всех подмножеств. В этом случае граф называется L -раскрашиваемым.

Основной стратегией для задач раскраски графа являются последовательные и жадные эвристики, которые дают с первой попытки результаты с локальным оптимумом.

Последовательная эвристика заключается в следующем. Сначала составляется упорядоченный в порядке убывания степеней вершин список. Первая вершина удаляется из списка и окрашивается в цвет 1. Просматриваются

ривая список, в цвет 1 раскрашиваются и удаляются из него все вершины, несмежные с первой выбранной и между собой. Далее выбирается вторая вершина из списка, она окрашивается в цвет 2 и удаляется. Процесс продолжается аналогично, пока не будут окрашены все вершины.

Жадный алгоритм — это оптимизационный алгоритм, который основан на жадной стратегии. Он просматривает серию альтернатив, выбирая лучшее решение. Запишем основные жадные эвристики для раскраски [184].

Эвристика 1. «Первые наибольшие». Вершины графа упорядочиваются по убыванию локальных степеней. Затем по порядку каждая вершина после анализа связности раскрашивается в определенный цвет.

Эвристика 2. «Наименьшие последние». Построение порядка из хаоса производится следующим образом. Выбирается вершина графа с наименьшей локальной степенью и удаляется из графа вместе с инцидентными ребрами. В оставшемся графе повторяется та же процедура. Вершины графа раскрашиваются в возможный для этой процедуры цвет в порядке, обратном порядку удаления. Затем вершины красятся в определенный цвет в порядке, обратном процессу удаления вершин.

Эвристика 3. «Разбиения по смежности». В начале алгоритма выбирается вершина графа максимальной степени и раскрашивается в цвет 1. Затем все нераскрашенные вершины разбиваются на два подмножества: X_1 — смежные с раскрашенными и X_2 — несмежные с раскрашенными. Выбор очередной вершины для раскраски в возможный цвет осуществляется по наибольшей степени в X_1 . Далее процесс продолжается аналогично до заполнения цвета 1. После этого вершины, раскрашенные цветом 1, удаляются и процедура продолжается до полной раскраски графа.

Эвристика 4. «Раскраска ранее упорядоченных вершин». Для вершины x_i выбирается цвет k , если и только если среди вершин, смежных с x_i -вершиной, существуют уже раскрашенные вершины цветов $c = 1, 2, \dots, k-1, k+1, k+2, \dots$ и нет вершин, помеченных цветом k .

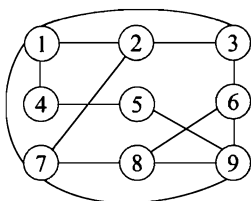
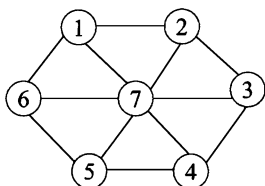
Эвристика 5. «Специальная раскраска в предписанные цвета». Предварительно для каждой вершины задан набор цветов, в которые она может быть раскрашена. Упорядочение вершин производится следующим образом. Определяются вершины с минимальным числом $N = \rho(x_i) + z_i/C$, где $\rho(x_i)$ — локальная степень вершины $x_i \in X$ графа $G = (X, U)$, z_i — число запрещенных цветов для данной вершины, C — общее число цветов. Вершина x_i с минимальным значением N удаляется из графа вместе с ребрами. Для оставшегося графа повторяется такая же процедура и так далее. Затем вершины красятся в возможный цвет в порядке, обратном порядку удаления [185].

Запишем простой алгоритм на основе одной из рассмотренных жадных эвристик:

1. Упорядочить вершины по убыванию локальных степеней.
2. Каждой вершине упорядоченного списка назначать первый имеющийся цвет, если это возможно.
3. Шаг 2 продолжать до полной раскраски графа.
4. Конец работы алгоритма.

Алгоритм на основе такой эвристики является жадным, так как он раскрашивает граф последовательно вершину за вершиной, присваивая каждой вершине цвет, если это возможно, не учитывая глобальных последствий такой раскраски. Жадный алгоритм задачи раскраски графа очень быстр (временная сложность алгоритма $O(n)$), так как он рассматривает лишь один из возможных вариантов раскраски каждой вершины.

Рассмотрим, например, задачу раскраски графа (рис. 6.48) [89]. Эвристика из хаоса вершин графа создает следующий порядок: $\langle 9\ 7\ 1\ 8\ 6\ 3\ 2\ 5\ 4 \rangle$. Сначала раскрашивается вершина 9, алгоритм оставляет вершину 7 не раскрашенной, так как она соединена с 9, а вершина 9 уже раскрашена. Эвристика не раскрашивает вершину 8 по той же причине. Далее раскрашивается вершина 1 и не раскрашиваются остальные. Следовательно, $X_1 = \{9, 1\}$. На втором шаге получим $X_2 = \{7, 6, 5\}$. Далее, $X_3 = \{8, 3, 4\}$ и $X_4 = \{2\}$. Получим, что граф G (рис. 6.48) раскрашен 4 цветами.

Рис. 6.48. Граф G Рис. 6.49. Колесо W_7

Рассмотрим второй граф, изображенный на рис. 6.49. Это колесо W_7 содержит 7 вершин. Первые шесть вершин расположены в виде кольца, а седьмая вершина расположена в центре кольца и соединена со всеми остальными вершинами. Жадная эвристика из хаоса вершин графа создает следующий порядок: $\langle 7\ 6\ 5\ 4\ 3\ 2\ 1 \rangle$. Алгоритм раскрасит вершины следующим образом: $X_1 = \{7\}$; $X_2 = \{6, 4, 2\}$ и $X_3 = \{5, 3, 1\}$. Получим, что колесо W_7 (рис. 6.49) раскрашено 3 цветами. Отметим, если в графе, аналогично колесу, имеется одна вершина, смежная всем остальным, то она раскрашивается одним цветом и удаляется из упорядоченного списка.

Так как жадная и последовательная эвристики раскраски графа достаточно быстрые, то предлагается совместить их с ГА с учетом концепций ЭМ. Стратегия совмещения предполагает, что используется кодирование нашего алгоритма для гибридного ГА. Кодировка, используемая в жадной эвристике, основана на упорядочивании вершин графа. Она упорядочивает вершины по уменьшению значений локальных степеней вершин и затем декодирует эту последовательность, назначая каждой вершине по порядку первый реальный цвет, где реальность основана на использовании цветов при раскраске предыдущих вершин. Такой способ кодировки при решении проблемы раскраски графа упорядочивает вершины графа определенным способом, а затем декодирует их в соответствии с жадной эвристикой.

Первое, что необходимо изменить в ГА — это замена бинарной ЦФ на ЦФ раскраски вершин. Согласно ЦФ выбираем первый ген (вершину) в хромосоме и присваиваем ей первый реальный цвет, если это возможно.

Каждая альтернатива учитывает предыдущие использованные цвета. Перестановка списка элементов получается в результате упорядочивания элементов в списке (тривиальная перестановка просто оставляет предыдущий порядок). Следующим шагом в ГА является замена технологии представления на технологию представления измененного списка. Это означает, что каждая хромосома является перестановкой списка вершин графа. Жадная эвристика сортирует список вершин и передает результат в эвристику декодирования.

На следующем шаге осуществляется замена инициализации на случайную или основанную на знаниях о решаемой задаче перестановку. В ГА будем генерировать случайным и заданным образом переставленный список вершин. Жадная эвристика генерирует только одну хромосому, но эта хромосома имеет шанс оказаться лучше, чем средняя. Такая процедура гарантирует, что с помощью ГА мы как минимум не ухудшим результат жадной эвристики, так как используется элитная и другие процедуры селекции для создания начальной популяции. В результате изменений порядка элементов в упорядоченном списке получим измененный список вершин графа. Упорядоченный список и список с изменениями и будут в дальнейшем использоваться. Применяя к полученным хромосомам генетические операторы, определяем ЦФ. В рассматриваемом алгоритме применяются элитная селекция и генетические операторы, описанные выше. Процесс совмещения связан с изменениями репродукционного модуля, где используются генетические операторы, работающие с упорядоченным представлением.

В [89] описан однородный универсальный ОК, приспособленный для области упорядоченных генетических операторов. Рассмотрим модификацию данного оператора. Этот оператор сохраняет основную часть первого родителя, а также добавляет информацию о втором. Информация, которая здесь закодирована, не имеет фиксированного значения, связанного с его позицией в хромосоме. Модифицированный универсальный ОК позволяет второму родителю «сказать» первому родителю, какие элементы следует упорядочить по-другому. Сетевой эффект модифицированного универсального ОК состоит в совмещении относительного порядка вершин двух родителей с хромосомами двух потомков.

Запишем алгоритм реализации этого ОК. Пусть заданы альтернативные решения, как родитель-1 и родитель-2, тогда потомок-1 получается следующим образом:

1. Генерируется бинарная строка-шаблон, количество элементов в которой совпадает с длиной родителей.
2. Гены из родителя-1, соответствующие единицам в строке шаблона, копируются на следующий шаг.
3. Создается список генов из родителя-1, соответствующих 0 в бинарной строке.
4. Последовательно просматривая элементы родителя-2, помещают в пустые позиции хромосомы еще не вошедшие гены.
5. Объединяя эти элементы без повторов, получаем потомка-1.
6. Конец работы алгоритма.

Построение потомка-2 производится аналогичным образом.

Рассмотрим пример для графа, приведенного на рисунке 6.48. Пусть альтернативные решения закодированы следующим образом:

p_1 :	1	2	3	4	5	6	7	8	9
p_2 :	9	7	5	3	1	8	6	4	2
БС:	0	1	0	0	0	1	0	0	0

Результат после шага два:

p_1 :	—	2	—	—	—	6	—	—	—
p_2 :	9	—	5	3	1	—	6	4	2

Окончательный результат:

p_3 :	9	2	5	3	1	6	7	4	8
p_4 :	9	8	5	3	1	7	6	4	2

Здесь БС — бинарная строка, показывающая смежность соответствующих вершин в p_1 и p_2 . То есть, 1 в БС показывает, что вершины 5 и 9 (рис. 6.48) связаны между собой. Если смежных вершин в заданных родителях нет, тогда бинарная строка заполняется 0 и 1 случайным образом. Выполнив жадное декодирование, получим, что хромосома p_3 соответствует следующей задаче раскраски графа (рис. 6.51): $X_1 = \{9, 2, 4\}$, $X_2 = \{5, 3, 7\}$, $X_3 = \{1, 6\}$ и $X_4 = \{8\}$. Для p_4 получим задачу раскраски графа: $X_1 = \{9, 3, 4\}$, $X_2 = \{8, 2, 1\}$, $X_3 = \{7, 6\}$ и $X_4 = \{5\}$. Здесь часть структуры каждого родителя зафиксирована одним родителем (как указано в битовой строке). Остальные переупорядочены так, что элементы остаются в том же порядке, в котором они были в другом родителе. Заметим, что основная проблема здесь — декодирование ОК. Эта операция имеет временную сложность алгоритма $O(n^2)$, где n — размер хромосомы.

В [89] описан оператор частичной мутации. Он выбирает часть родительской хромосомы, и, произведя перестановки внутри нее, переносит эту часть в потомка, а остальную оставляет без изменений. Такой тип мутации не имеет параметров, хотя для длинных хромосом полезно ограничивать длину секции. Все генетические операторы такого вида имеют сложное декодирование. Опишем жадную стратегию с одновременным декодированием хромосом. Пусть для графа G (рис. 6.48) задана популяция $P = \{p_1, p_2, p_3, p_4\}$:

p_1 :	1	2	3	4	5	6	7	8	9
p_2 :	9	8	7	6	5	4	3	2	1
p_3 :	1	3	5	7	9	2	4	6	8
p_4 :	9	7	5	3	1	8	6	4	2

В хромосоме p_1 выбирается ген, соответствующий вершине с максимальной степенью. Если таких вершин несколько, то они анализируются последовательно в произвольном порядке. В первой хромосоме выбираем

элемент 9. Согласно жадному ОК следующей выбирается вершина 1. Дальнейшие шаги жадного ОК на всей популяции приводят к тупиковым ситуациям. Следовательно, определен первый цвет $X_1 = \{9, 1\}$. Элементы 9, 1 исключаются из всех хромосом популяции P . Далее в p_1 выбирается элемент 7. Согласно жадному ОК получим второй цвет $X_2 = \{7, 6, 4\}$. Продолжая, аналогично определим, что $X_3 = \{8, 2, 5\}$ и $X_4 = \{3\}$. Следовательно, граф (рис. 6.48) раскрашивается четырьмя цветами. Данная методика позволяет сокращать размер хромосом (альтернативных решений) на каждом шаге и соответственно время раскраски графа.

Кроме генетических операторов необходимо также определить тип селекции. Для задачи раскраски графа наиболее применима модифицированная элитная селекция, которая реализуется по следующей схеме [17, 89]. Все хромосомы t -го поколения упорядочиваются в порядке убывания значений степеней вершин (приспособленности к внешней среде). Выживают только первые i из упорядоченных хромосом, т. е. те, для которых ранг $r(p'_k)$, равный k -й позиции в упорядоченной последовательности хромосом, меньше или равен i , то есть, $r(p'_k) \leq i$.

Структура гибридного ГА, используемого для задачи раскраски графа, представлена на рис. 6.50.

В данной схеме используются идеи совместной эволюции, установление баланса, адаптации к внешней среде, активное взаимодействие с внешней средой, иерархическое управление генетическим поиском, построение порядка из хаоса, локальный поиск решений и применение всех модифицированных генетических операторов на основе жадной стратегии и поисковых методов. Временная сложность алгоритма такого класса лежит в пределах $O(n^2) \div O(n^3)$, где n — длина хромосомы (альтернативного решения).

Задача раскраски графа тесно связана с построением независимых или внутренне устойчивых подмножеств, а также определением клик графа. Если две любые вершины подмножества X' графа $G = (X, U)$, $X' \subseteq X$, не смежны, то оно называется внутренне устойчивым. Подмножество $\Psi_i \subseteq X$ графа $G = (X, U)$ называется максимальным внутренне устойчивым подмножеством или независимым, если добавление к нему любой вершины $x_i \in X$ делает его не внутренне устойчивым. Подмножество Ψ_i будет независимым, если

$$\forall x_i \in \Psi_i \quad (G_i \cap \Psi_i = \emptyset).$$

Здесь G_i — подмножество вершин, смежных x_i .

Независимые подмножества различаются по числу входящих в них элементов. В произвольном графе G можно выделить семейство всех независимых подмножеств вида $\Psi_i = \{\Psi_1, \Psi_2, \dots, \Psi_s\}$ или его части [159]. Жадные стратегии, используемые для задачи раскраски графа, также эффективно применяются для построения семейства независимых подмножеств. Например, построим семейство независимых подмножеств для графа G , представленного на рисунке 6.51. Здесь предварительно произведем упорядочивание вершин по возрастанию, начиная с наименьшей локальной степени. Работу модифицированного жадного ОК покажем на примере.

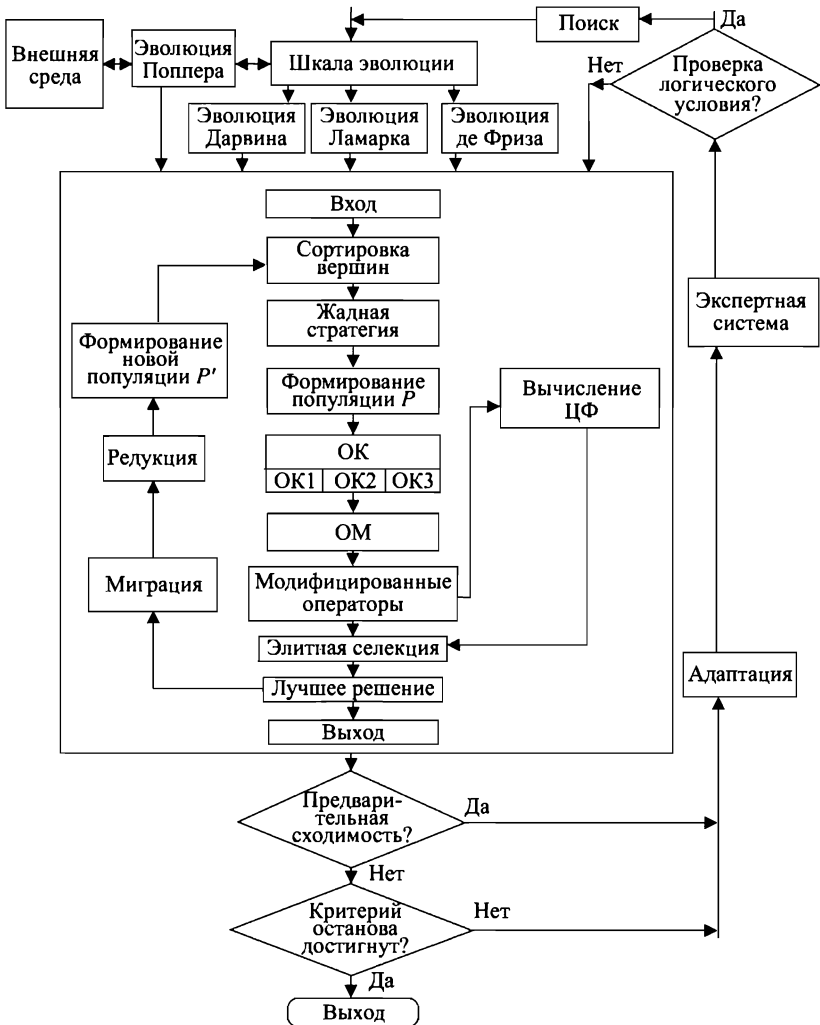


Рис. 6.50. Структура гибридного ГА

Пусть для графа G (рис. 6.51) задана популяция $P = \{p_1, p_2, p_3, p_4, p_5\}$:

p_1 :	1	2	3	4	5	6	7	8	9
p_2 :	9	8	7	6	5	4	3	2	1
p_3 :	6	1	5	4	7	9	3	8	2
p_4 :	4	9	3	7	8	1	7	6	2
p_5 :	5	6	7	8	9	2	3	4	1

Работа алгоритма начинается с выбора в упорядоченной хромосоме p_5 первого элемента — 5. Далее выбираем вершину 6, так как она соседняя вер-

шине 5 в хромосоме p_5 и не смежна ей. Аналогично выбираем 7 — вершину в этой хромосоме. После этого, анализируя популяцию P , в хромосоме p_3 выбираем вершину 9. Дальнейший анализ популяции P показывает, что построено первое независимое подмножество $\Psi_1 = \{5, 6, 7, 9\}$.

Существует большое число стратегий построения независимых подмножеств. Одна из них состоит в построении независимых строительных блоков на две, три, четыре и так далее вершин с дальнейшим объединением этих строительных блоков в независимое подмножество. Другая стратегия (в ширину) предусматривает нахождение всех независимых подмножеств, в которые входит первая рассматриваемая вершина (в нашем случае вершина 5) с дальнейшим анализом других вершин на предмет создания новых независимых подмножеств. Третья стратегия (в глубину) предусматривает последовательный анализ всех вершин в упорядоченном списке (хромосома p_5) до получения семейства независимых подмножеств. Четвертая стратегия является комбинацией первых трех. Применяя четвертую стратегию для графа G (рис. 6.51), получим набор независимых подмножеств $\Psi_2 = \{6, 7, 8\}$, $\Psi_3 = \{7, 8, 2\}$, $\Psi_4 = \{8, 1, 3\}$, $\Psi_5 = \{9, 2, 4, 5\}$, $\Psi_6 = \{1, 2, 5\}$, $\Psi_7 = \{2, 5, 7, 9\}$, $\Psi_8 = \{3, 5, 7, 9\}$, $\Psi_9 = \{4, 5, 6, 9\}$. Очевидно, что данный набор не полный. Для получения полного семейства независимых подмножеств необходимо продолжить реализацию данных стратегий.

Описанная методика эффективно используется для задачи раскраски графа. Например, пусть Ψ_5 задает первый цвет $X_1 = \{9, 2, 4, 5\}$, Ψ_2 задает второй цвет $X_2 = \{6, 7, 8\}$, а Ψ_3 задает третий цвет $X_3 = \{1, 3\}$. Следовательно, граф G (рис. 6.51) раскрашен тремя красками. Временная сложность алгоритма построения семейства независимых подмножеств лежит в пределах $O(n^2) \div O(n^4)$.

Заметим, что обратной для построения независимого подмножества является задача построения клик графа [123, 159]. Подмножество из максимального числа смежных между собой вершин в графе называется кликой. В графе можно выделить некоторое семейство клик: $K = \{k_1, k_2, \dots, k_s\}$. Очевидно, что стратегии, описанные выше, могут быть применены для определения семейства клик. Покажем на примере графа G (рис. 6.52) упрощенную схему работы ГА для определения клик графа.

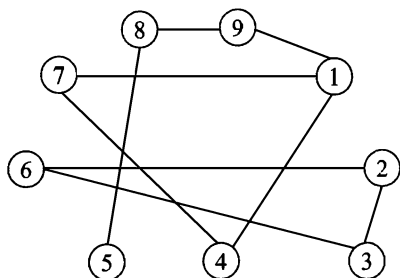


Рис. 6.51. Граф G для определения независимых подмножеств

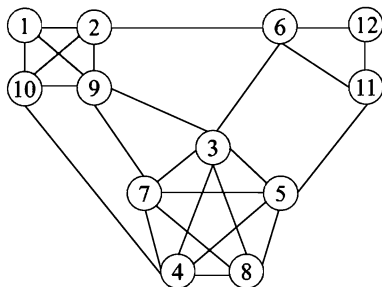


Рис. 6.52. Граф G для определения клик

Пусть для графа G (рис. 6.52) задана популяция $P = \{p_1, p_2, p_3, p_4, p_5\}$:

p_1 :	1	2	3	4	5	6	7	8	9	10	11	12
p_2 :	12	11	10	9	8	7	6	5	4	3	2	1
p_3 :	2	4	6	8	10	12	1	3	5	7	9	11
p_4 :	1	5	9	2	6	10	3	7	8	4	11	12
p_5 :	1	10	12	6	2	9	11	4	8	7	3	5

Применим следующую схему поиска, основанную на ГА и жадной стратегии:

1. 1.
2. 1, 2 — образовано полное подмножество на две вершины.
3. 1, 2, 3 ♦(тупик).
4. 1, 3 ♦.
5. 1, 5 ♦.
6. 1, 10, 11 ♦.
7. 1, 10, 9, 11 ♦.
8. 1, 10, 9, 8 ♦.
9. 1, 10, 9, 2, 3 ♦.
10. 1, 10, 9, 2, 4 ♦.
11. 1, 10, 9, 2, 6 ♦.

Построена первая клика $k_1 = \{1, 10, 9, 2\}$. Элементы, соответствующие вершинам k_1 , удаляются из всех хромосом популяции. Получаем новую популяцию с хромосомами меньшей длины:

p_6 :	3	4	5	6	7	8	11	12
p_7 :	12	11	8	7	6	5	4	3
p_8 :	4	6	8	12	3	5	7	11
p_9 :	5	6	3	7	8	4	11	12
p_{10} :	12	6	11	4	8	7	3	5

Продолжая далее:

1. 3.
2. 3, 4.
3. 3, 4, 5.
4. 3, 4, 5, 6 ♦.
5. 3, 4, 5, 12 ♦.
6. 3, 4, 5, 7, 8, 11 ♦.
7. 3, 4, 5, 7, 8, 6 ♦.
8. 3, 4, 5, 7, 8, 12 ♦.

Построена вторая клика $k_2 = \{3, 4, 5, 7, 8\}$. Далее, продолжая аналогично, построим третью клику $k_3 = \{6, 11, 12\}$. Отметим, что за 21 элементарный шаг на основе жадной модифицированной стратегии генетического поиска получены основные три клики графа. Временная сложность алгоритма в лучшем случае $O(n)$, в самом худшем случае $O(n!)$. В среднем для реальных графов имеем $O(n^2) \div O(n^3)$.

6.8. Определение планарности графов на основе генетического поиска

Проблема определения планарности и получения плоской укладки графа относится к числу известных задач теории графов.

Граф $G = (X, U)$ называют *плоским*, если существует такое изображение на плоскости его вершин и ребер, что каждая вершина $x_i \in X$ изображается на плоскости отдельной точкой и каждое ребро $u_j \in U$ изображается простой кривой, имеющей концевые точки (x_i, x_k) , причем эти кривые пересекаются только в общих концевых точках, т. е. в вершинах. Здесь и далее под термином граф понимается связный неориентированный граф [123].

Граф называется *планарным*, если он изоморфен плоскому, т. е. существует возможность получения плоской укладки такого графа. Область плоскости, ограниченная ребрами плоского графа и не содержащая внутри себя ни вершин, ни ребер, называется *гранью*. Известная формула Эйлера связывает число вершин и ребер плоского графа с числом его граней: $n - m + f = 2$, где n — число вершин, m — число ребер графа, f — число граней графа [123]. Исходя из указанной формулы, был сформулирован ряд следствий:

С л е д с т в и е 1. В любом простом планарном графе существует вершина, степень которой не больше пяти.

С л е д с т в и е 2. Каждый планарный граф G с $n \geq 4$ вершинами имеет по крайней мере четыре вершины со степенями, не превышающими 5.

С л е д с т в и е 3. Если G — связный простой планарный граф с $n \geq 3$ вершинами и m ребрами, то $m \leq 3n - 6$.

Приведенные следствия определяют зависимость планарности графа от числа его вершин и ребер и задают границы интервала по числу ребер, при попадании в который необходимо проводить дополнительные исследования, чтобы получить достоверный ответ на вопрос, планарный ли исследуемый граф.

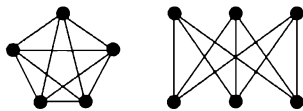
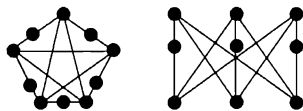
Все графы можно условно разбить на три непересекающихся подмножества: G'_1 , G'_2 и G'_3 . В первое подмножество G'_1 входят заведомо планарные графы, для которых справедливо неравенство $m \leq n + 2$. Второе подмножество G'_2 образуют заведомо непланарные графы, для которых справедливо неравенство $m > 3(n - 2)$. А третье подмножество G'_3 составляют графы, число ребер которых находится в пределах интервала: $(n + 2) < m \leq 3(n - 2)$. Именно для этих графов требуется проведение дополнительных исследований, поскольку каждый из них может быть как планарным, так и не планарным.

С л е д с т в и е 4. Графы K_5 и $K_{3,3}$ не являются планарными.

Здесь K_5 — это полный граф на пять вершин, а $K_{3,3}$ — двудольный однородный граф с локальной степенью вершин, равной трем (рис. 6.53).

Критерий Понтрягина–Куратовского. Граф планарен тогда и только тогда, когда он не содержит подграфа, гомеоморфного K_5 или $K_{3,3}$.

Данный критерий является наиболее распространенным. Смысл его можно пояснить на примере. Пусть задан граф $G = (X, U)$, не имеющий петель и кратных ребер. *Подразбиением* ребра $u = (x_i, x_j)$ называется замена данного ребра двумя ребрами $u_1 = (x_i, x_k)$ и $u_2 = (x_k, x_j)$ с введением промежуточной вершины x_k . Два графа называются *гомеоморфными*, если они имеют изоморфные подразделения [123]. Примеры графов, гомеоморфных графам K_5 и $K_{3,3}$, показаны на рисунке 6.54.

Рис. 6.53. Графы K_5 и $K_{3,3}$ Рис. 6.54. Графы, гомеоморфные графам K_5 и $K_{3,3}$

Существуют и другие критерии планарности.

Критерий Харари–Татта. Граф планарный тогда и только тогда, когда у него нет подграфов, стягиваемых к K_5 и к $K_{3,3}$.

Критерий Уитни. Граф G планарный тогда и только тогда, когда он имеет комбинаторно двойственный граф G^* .

Уитни связал планарность графов с существованием двойственных графов. Для построения *геометрически двойственного* графа G^* каждая грань исходного графа G , включая внешнюю, преобразуется в вершину графа G^* .

При этом вершины графа G^* будут смежными в случае, если были смежными (т. е. имели общее ребро) соответствующие грани графа G .

Критерий Мак-Лейна. Граф G планарный тогда и только тогда, когда каждый его блок, содержащий, по крайней мере, три вершины, обладает таким базисом циклов $Z_1, Z_2, \dots, Z_q$ и таким дополнительным циклом Z_0 , что любое ребро блока принадлежит точно двум из этих $q + 1$ циклов. Этот критерий основан на рассмотрении циклической структуры графа. Как следует из формулы Эйлера, все внутренние грани двусвязного плоского графа G образуют базис циклов $Z_1, Z_2, \dots, Z_q$, где q — циклический ранг графа G . Кроме того, имеется также Z_0 — внешний простой цикл графа G . Тогда любое ребро такого графа G будет принадлежать точно двум из $q + 1$ циклов Z_i .

Практическое использование перечисленных критериев затруднено из-за необходимости полного перебора для содержательного рассмотрения графа. Поэтому были разработаны эвристические методы определения планарности. Условно их можно разделить на три класса: циклические, матричные и комбинированные (рис. 6.55). Рассмотрим кратко идею каждого из перечисленных подходов.

Циклический метод определения планарности заключается в выделении в заданном графе произвольных циклов, в результате чего граф разбивается на подграфы, каждый из которых проверяют на планарность, уменьшая таким образом размерность задачи. Планарность исходного графа определяется исходя из планарности подграфов, полученных в результате выделения циклов. Основным недостатком этих методов заключается в том, что

задача определения необходимых циклов сложна и требует значительных затрат времени.

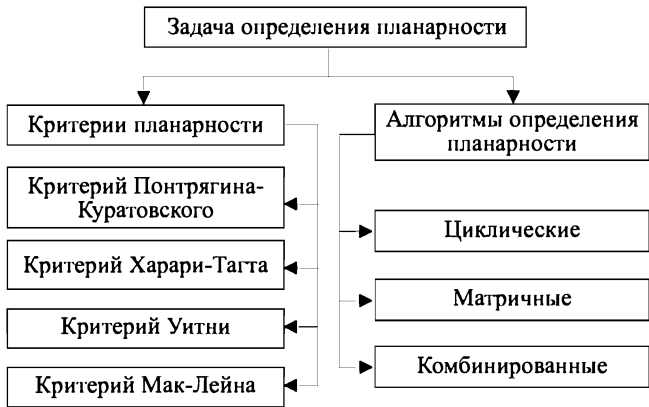


Рис. 6.55. Классификация методов определения планарности

Матричные методы определения планарности используют некоторые специальные матрицы графа, например, матрицу линейно независимых циклов. При этом отправной точкой является критерий Мак-Лейна, отмечающий, что граф планарный, если в матрице всех его возможных циклов можно выделить подматрицу, содержащую $m - n + 2$ строк и m столбцов, причем каждый столбец содержит строго две единицы. Однако такой вариант подразумевает большие затраты времени, особенно если исследуемый граф оказывается непланарным.

Работу матричного алгоритма определения планарности проиллюстрируем на примере. Пусть задан граф G , изображенный на рисунке 6.56. Граф имеет $n = 6$, $m = 9$, а число строк, которое необходимо выделить из матрицы циклов B_{am} для установления планарности графа, будет равно: $m - n + 2 = 9 - 6 + 2 = 5$. В матрице B_{am} строки — номера циклов, а столбцы — номера ребер графа. В процессе работы алгоритма генерируется множество всех возможных циклов длины k :

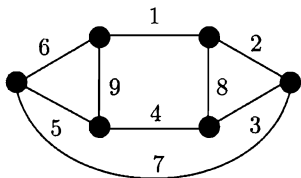


Рис. 6.56. Исходный граф G

$C_k = \{C_1, C_2, \dots, C_a\}$, где a — число циклов длины k . Каждый элемент множества C_k представляет собой цикл длины k , состоящий из номеров ребер, составляющих данный цикл. Сформируем реберно-ориентированную матрицу B_{am} , вектор множества ребер M (рис. 6.57) и вектор комбинаций циклов C_k . В векторе C_k выбирается первая комбинация $C_1 = [11111000000000]$ и находится алгебраическое произведение векторов: $C_1 M = 3 + 3 + 4 + 4 + 4 = 18$. На следующем шаге полученное значение $C_1 M = p$ сравнивается с числом $2m$. В случае, если $p < 2m$, алгоритм генерирует новую комбинацию циклов и процесс повторяется. Если $p > 2m$, то исследуемый граф непланарный. В случае же, когда $p = 2m$, как,

например, в рассматриваемом примере, алгоритм продолжает работу. Если в результате удалось сформировать подматрицу B_c (рис. 6.58), отвечающую критерию Мак-Лейна, то граф планарный, в противном случае после генерации новой комбинации циклов процесс повторяется.

$$B_{am} = \begin{bmatrix} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 \\ 0 & 1 & 1 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 1 & 1 & 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 & 1 & 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 & 0 & 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 & 1 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 & 1 & 1 & 1 \end{bmatrix}; \quad M = \begin{bmatrix} 3 \\ 3 \\ 4 \\ 4 \\ 4 \\ 5 \\ 5 \\ 5 \\ 5 \\ 5 \\ 5 \\ 6 \\ 6 \\ 6 \end{bmatrix}.$$

Рис. 6.57. Реберно-ориентированная матрица и вектор множества ребер

$$B_e = \begin{bmatrix} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 \\ 0 & 1 & 1 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 1 & 1 & 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 1 \end{bmatrix}.$$

Рис. 6.58. Подматрица B_c

Основная проблема рассмотренного алгоритма — это генерация и хранение необходимых для работы алгоритма массивов данных (матрицы инцидентности, циклов, реберно-ориентированная матрица и так далее).

К числу достоинств алгоритма можно отнести возможность гарантированного определения планарности.

Циклические алгоритмы, начинают свою работу с определения в исследуемом графе $G = (X, U)$ некоторого простого, т. е. не имеющего повторяющихся вершин, цикла произвольной длины. Предварительно проводится исследование заданного графа на предмет удаления висячих вершин и инцидентных им ребер, а также стягивания вершин, имеющих локальную степень равную двум. Кроме того, необходимо проверить, является ли исходный граф связным, а также определить, не является ли рассматриваемый граф заведомо планарным, либо заведомо непланарным. После выполнения всех операций производится факторизация связных компонент, т. е. стягивание всех вершин и ребер, принадлежащих компоненте, в одну вершину. Затем каждая образовавшаяся компонента, включая цикл, проверяется на возможность построения плоской укладки. В графе находится произвольный цикл, после удаления которого граф распадается на отдельные фрагменты.

Затем производится попытка расположить каждый полученный фрагмент на плоскость во внешней либо внутренней областях первоначального цикла. Далее укладки фрагментов, по возможности, объединяют, получая плоскую укладку всего графа. Алгоритм основан на идеях поиска в глубину и имеет временную сложность $O(n \log n)$.

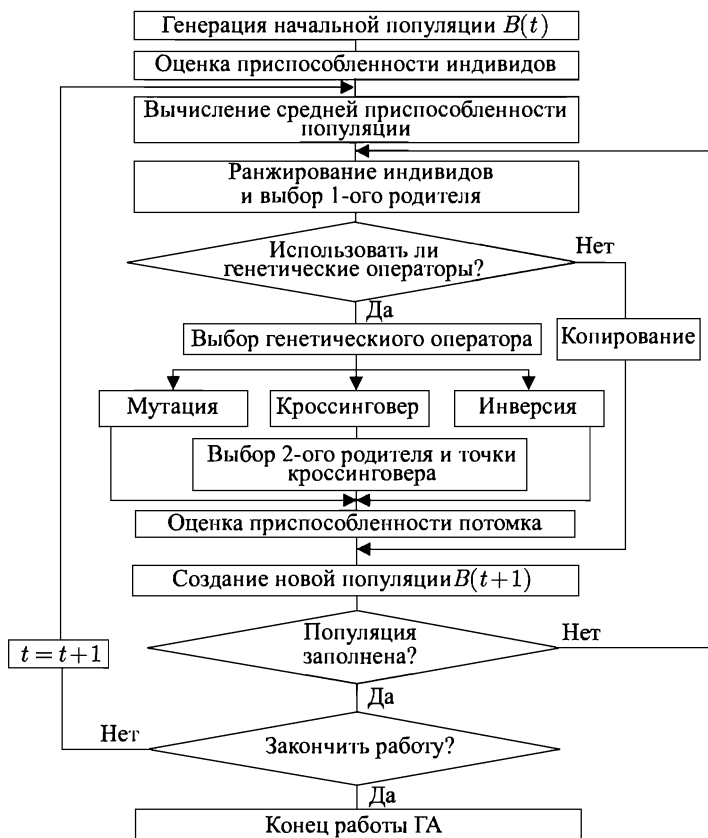


Рис. 6.59. Модифицированная схема работы ГА

Рассмотрим механизм использования наследственной информации на примере задачи минимизации пересечений. Для этого в заданном графе выделяем произвольным образом некоторый цикл, разбивая таким образом всю плоскость на две полуплоскости. Причем все ребра, инцидентные вершинам цикла, располагаются либо внутри, либо снаружи цикла. Таким образом получаем начальную «родительскую» пару хромосом, каждая из которых несет информацию о расположении ребер. При этом целевой функцией будет число пересечений ребер в данной полуплоскости.

При решении задачи минимизации пересечений возможен вариант, когда из одной полуплоскости в другую перемешивают не отдельные ребра,

а последовательность ребер, расположенных в строго определенном порядке. В этом случае хромосома представляет собой закодированную в определенной последовательности цепочку ребер, и каждый ее разряд есть звено цепи. Следовательно, применение стандартных операторов может привести к разрыву цепи и получению нереального решения.

Для применения универсального ОК к подобной хромосоме необходимо предварительно сформировать маску оператора таким образом, чтобы при обмене ребрами во время выполнения ОК изменение внутренней структуры цепочек ребер не приводило к образованию разрывов.

Рассмотрим порядок выполнения ОМ для определения планарности графа. Например, в ходе решения задачи получена векторная хромосома, каждый ген которой представляет собой закодированную последовательность ребер. Для выполнения операции мутации случайным образом выбираем ген:

{1, 2, 3}	{4 5, 6}	{7, 8, 9}	{10, 11, 12}

где

{7,8,9}

 — ген, выбранный для мутации. А затем к выбранному гену применяем мутацию:

7	8	9	ген до мутации
7	9	8	ген после мутации

где || — точка мутации. После выполнения операции простой мутации получаем хромосому следующего вида:

{1,2,3}	{4,5,6}	{7,9,8}	{10,11,12}

После выполнения ОМ последовательность ребер в гене 3 изменена, что может привести к разрыву цепи и получению нереального решения.

Покажем порядок выбора точки мутации с помощью чисел Фибоначчи. Пусть имеется векторная хромосома, представляющая некоторое число цепочек ребер. Для выполнения операции мутации случайным образом выбираем ген:

{1,2,3,4,5}	{6,7,8,9,10,11}	{12,13,14,15,16,17,18,19,20}	{21,22,23,24}

А затем к выбранному гену применяем ОМ, причем разряды для мутации выбираются в соответствии с числами Фибоначчи, т. е. 1, 2, 3, 5, 8, ..., а обмен числовых значений между разрядами происходит по кругу со сдвигом вправо:

12	13	14	15	16	17	18	19	20	ген до мутации
19	12	13	15	14	17	18	16	20	ген после мутации

После выполнения ОМ получаем хромосому следующего вида:

{1,2,3,4,5}	{6,7,8,9,10,11}	{19,12,13,15,14,17,18,16,20}	{21,22,23,24}

Рассматриваемый алгоритм имеет блочную структуру, где каждый блок является отдельным фрагментом, входные данные для которого предоставляет предыдущий блок (за исключением начального блока). Укрупненная (блочная) структурная схема алгоритма представлена на рисунке 6.60. Опишем функции каждого блока.

Первым блоком является ввод исходных данных. Он служит для преобразования заданного произвольным образом (матричным, графическим или иным образом) исходного графа к форме исходных данных, используемых алгоритмом.

Блок генерации циклов производит генерацию всех циклов заданной длины, выполняя это действие столько раз, сколько требует от него блок анализа. Иначе говоря, он является своего рода поставщиком «сырья» для блока генетических операторов, в котором происходит процесс планаризации исследуемого графа, т.е. плоской укладки графа на плоскость. Блок укладки выполняет плоскую укладку графа, либо его максимально планарной части.

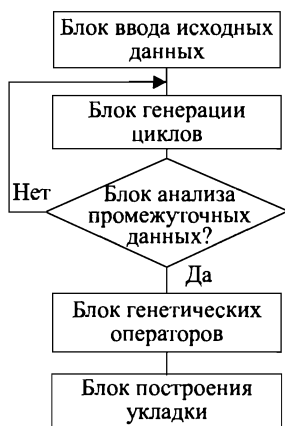


Рис. 6.60. Укрупненная структурная схема алгоритма плоской укладки

Исходный граф G может быть задан матрицей либо списком смежности. Матричное представление является избыточным. Для алгоритма определения планарности используется модифицированная форма списка смежности LS, где каждой вершине ставятся в соответствие номера ребер, инцидентных рассматриваемой вершине, например, запись $1(1,2)$ означает, что ребро инцидентно вершинам x_1 и x_2 и имеет номер 1, причем в список смежности некоторой вершины x_i не включаются номера вершин от x_1 до x_{i-1} включительно.

Например, для сгенерированного случайным образом графа G , изображенного на рисунке 6.61, исходные данные для начала работы алгоритма запишутся:

$$\begin{aligned}
 Lx_1 &= \{1(1,2), 2(1,3), 3(1,5), 4(1,6), 5(1,8)\}; & Lx_2 &= \{6(2,3), 7(2,8)\}; \\
 Lx_3 &= \{8(3,6), 9(3,8)\}; & Lx_4 &= \{10(4,6), 11(4,7), 12(4,8)\}; \\
 Lx_5 &= \{13(5,6), 14(5,7), 15(5,8)\}; & Lx_6 &= \{16(6,7)\}.
 \end{aligned}$$

Здесь, например, запись $Lx_6 = \{16(6,7)\}$ означает, что вершины 6 и 7 графа G связаны ребром 16. Такая форма представления исходных данных не содержит избыточной информации, и позволяет увеличить эффективность алгоритма генерации циклов.

Генерация циклов осуществляется на основе поиска в глубину. Число списков смежности будет меньше либо равно числу вершин графа, а число элементов в списке Lx_i находится в интервале: $0 < |Lx_i| < \rho(x_i)$. Процесс

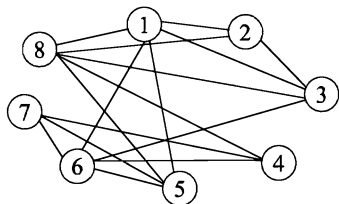


Рис. 6.61. Граф G , заданный случайным образом

генерации циклов k -й длины начинается со списка Lx_1 , в котором выбирается вершина x_i , стоящая на первом месте, после чего выполняется переход к списку Lx_i . В случае, если на любом из шагов обнаруживается невозможность дальнейшего продолжения (замыкания) цепи, алгоритм возвращается на один шаг назад и просматривает другие возможности для выбора пути. Таким образом форми-

руется множество циклов длины k , представляющих собой цепочки вида $x_1 \dots x_i \dots x_k \dots x_1$. После того, как просмотрены все возможности и сформированы все циклы, использующие вершину x_1 в качестве начальной, список Lx_1 удаляется из рассмотрения, после чего процесс повторяется для следующего списка.

В процессе работы алгоритма генерируется множество всех возможных циклов длины k : $C_k = \{C_1, C_2, \dots, C_a\}$, где a — число циклов длины k . Каждый элемент множества C_k представляет собой цикл длины k , состоящий из номеров ребер, составляющих данный цикл. Так, например, для графа, изображенного на рисунке 6.61, в процессе работы блока генерации циклов были получены следующее множество циклов длины три:

$$\begin{aligned} C_1 &= (1 - 6 - 2); & C_2 &= (1 - 7 - 5); & C_3 &= (2 - 8 - 4); \\ C_4 &= (2 - 9 - 5); & C_5 &= (3 - 13 - 4); & C_6 &= (3 - 15 - 5); \\ C_7 &= (6 - 9 - 7); & C_8 &= (10 - 16 - 11); & C_9 &= (13 - 16 - 14). \end{aligned}$$

Здесь, например, в цикле C_3 , 2 это ребро (1, 3), 8 — ребро (3, 6), 4 — ребро (1, 6). На основе этих списков может быть сформирована матрица циклов $B = ||b_{ij}||_{c \times m}$, где c — число циклов, а m — число ребер графа, причем элемент матрицы $b_{ij} = 1$ в случае, если ребро m_j принадлежит циклу c_i ($m_j \in c_i$), и $b_{ij} = 0$ в противном случае. Пример матрицы циклов длины 3 показан на рисунке 6.62.

Полученные данные являются исходными для работы ГА. Алгоритм анализа и принятия решений можно записать в следующем виде:

1. Подсчет числа циклов $N_{c_k} = |C_k|$ и числа повторов N_{m_j} каждого ребра m_j , где $j = 1, \dots, M$.

2. Если выполняется условие: $N_{c_k} \geq m - n + 2$ и $N_{m_j} \geq 2 \forall j \in M$, то переход к пункту 3, если нет, то возврат к блоку генерации циклов и начало генерации циклов длины $k + 1$.

3. Из множества циклов C_k выбираются первые $m - n + 2$ цикла и подсчитывается общее число ребер N_m , принадлежащих этим циклам: $N_m = \sum_{k=3}^K k \cdot |C_k|$, где k — длина цикла, $|C_k|$ — мощность множества циклов длины k , K — длина самого длинного из всех сгенерированных циклов,

вычисляемая по формуле $K = k + s$, где s — число шагов в блоке генерации циклов.

4. Полученное значение N_m сравнивается с удвоенным числом ребер исходного графа G . Если $N_m > 2m$, то рассматриваемый граф является непланарным. После чего необходимо принять решение. Если нужно выделить максимальную планарную часть графа и получить для нее плоскую укладку, то переход к пункту 6, в противном случае к 7.

5. Необходимо сформировать такую комбинацию циклов, чтобы выполнялось условие: $N_m = 2m$. Если полученное значение $N_m < 2m$, то необходимое условие достигается путем замены некоторого числа циклов меньшей длины (например длины k) на циклы большей длины (циклы длины $k + 1$). Если циклы длины $k + 1$ отсутствуют, то нужно вернуться к блоку генерации циклов и начать генерацию циклов длины $k + 1$, в противном случае переходим к пункту 6.

6. В случае, когда $N_c = 2m$, либо после преобразований, описанных в пункте 5, либо в случае, описанном в пункте 4, работа блока анализа завершена, полученные данные (множество циклов и предполагаемая схема комбинации) фиксируются и передаются для работы ГА.

7. Конец работы алгоритма.

Для графа, представленного на рисунке 6.61, процесс выбора решения будет следующим.

Согласно пункту 1 алгоритма:

1.1. Сформировано девять циклов длины три ($N_{c3} = 9$).

1.2. Необходимое число циклов $p = m - n + 2 = 16 - 8 + 2 = 10$. Поскольку $N_{c3} < p$, то необходимо вернуться к блоку генерации циклов и начать генерацию циклов длины 4.

Согласно пункту 2 алгоритма:

2.1. В результате работы блока генерации циклов сформировано 16 циклов длины четыре:

$$\begin{aligned} C_{10} &= (1-6-8-4); & C_{11} &= (1-6-9-5); & C_{12} &= (1-7-9-2); \\ C_{13} &= (1-7-15-3); & C_{14} &= (2-6-7-5); & C_{15} &= (2-8-13-3); \\ C_{16} &= (2-9-15-3); & C_{17} &= (3-14-16-4); & C_{18} &= (4-8-9-5); \\ C_{19} &= (4-10-12-5); & C_{20} &= (4-13-15-5); & C_{21} &= (8-10-12-9); \\ C_{22} &= (8-13-15-9); & C_{23} &= (10-13-14-11); & C_{24} &= (10-13-15-12); \\ C_{25} &= (11-14-15-12). \end{aligned}$$

2.2. Общее число циклов стало равным двадцати пяти ($N_c = 9 + 16 = 25$), т.е. $N_c > p$. Число повторений каждого ребра $N_{m_j} > 2$, поэтому переходим к пункту 3.

2.3. Поскольку $p = 10$, то для первых десяти циклов подсчитываем число ребер: $N_c = 3 \times 9 + 4 \times 1 = 27 + 4 = 31$.

2.4. Число ребер N_m не больше удвоенного числа ребер графа $2m$. ($2m = 2 \times 16 = 32$), поэтому переходим к пункту 5.

2.5. Полученное значение N_m меньше удвоенного числа ребер графа ($2m = 2 \times 16 = 32 \Rightarrow N_c < 2m$). Формируем предполагаемую схему

комбинации. Для этого необходимо заменить один из циклов длины три на цикл длины четыре.

2.6. После проведения замены условие $N_c = 2m$ выполнено и выясняется, что для решения поставленной задачи необходимо из имеющегося множества циклов выделить некоторый базис, включающий десять циклов, в том числе 8 циклов длины три и 2 цикла длины четыре.

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
C_1	1	1	0	0	0	1	0	0	0	0	0	0	0	0	0	0
C_2	1	0	0	0	1	0	1	0	0	0	0	0	0	0	0	0
C_3	0	1	0	1	0	0	0	1	0	0	0	0	0	0	0	0
C_4	0	1	0	0	1	0	0	0	1	0	0	0	0	0	0	0
C_5	0	0	1	1	0	0	0	0	0	0	0	0	1	0	0	0
C_6	0	0	1	0	1	0	0	0	0	0	0	0	0	0	1	0
C_7	0	0	0	0	0	1	1	0	1	0	0	0	0	0	0	0
C_8	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	1
C_9	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	1

Рис.6.62. Матрица циклов длины три

Блок генетических операторов является основным блоком алгоритма планаризации, выполняющим исследование исходного графа на предмет определения планарности, а также нахождение вариантов его плоской укладки. Задача определения планарности и построения плоской укладки графа рассматривается как задача построения некоторого базиса (здания), удовлетворяющего установленным критериям (нормативам). Строительство этого базиса, также как и строительство здания, проводится в несколько этапов:

1 этап. «Строительство фундамента» процесс формирования начального базового решения, которое должно стать основой для дальнейших преобразований.

2 этап. «Строительство каркаса» это процесс дополнения имеющегося базового решения новыми комбинациями циклов до достижения максимально возможной степени заполнения. В зависимости от достигнутого на втором этапе результата (оптимальное или квазиоптимальное заполнение) процесс строительства либо завершается, либо переходит к следующему этапу.

3 этап. «Индивидуальная достройка» — алгоритм пытается путем пошаговых изменений улучшить полученное ранее квазиоптимальное решение, т. е. найти глобальный оптимум. В случае, если это удастся, процесс строительства завершается, в противном случае алгоритм переходит к следующему, четвертому и последнему, этапу строительства.

4 этап. «Корректировка проекта». На этом этапе полученное квазиоптимальное решение в виду невозможности дальнейшего улучшения запоминается, после чего в данное решение («проект») вносятся изменения и строительство возобновляется со второго этапа.

В качестве строительных методов на первых трех этапах используются генетические операторы селекции, кроссинговера и отбора, адаптиро-

ванные к специфике поставленной задачи. Задача корректировки проекта выполняется с помощью операторов мутации и инверсии.

Сырьем для производства строительных блоков, используемых в процессе строительства, является множество циклов, сформированное блоком генерации циклов, причем строительные блоки могут быть размера два (число циклов в каждом из них) на первом и втором этапах работы генетического алгоритма, либо один. Схема функционирования блока генетических операторов показана на рисунке 6.63.

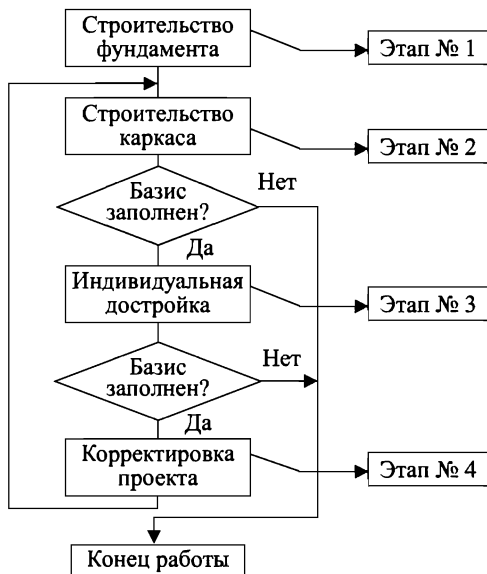


Рис. 6.63. Структурная схема ГА

Исходными данными для рассматриваемого ГА является множество циклов, из которого необходимо некоторым образом выделить базис циклов B_p , где $p = m - n + 2$. То есть, для представления базиса в виде хромосомы необходимо, чтобы в ней было не менее p разрядов (генов), каждый из которых соответствует номеру цикла. Таким образом, под хромосомой понимается комбинация из p циклов, являющаяся одновременно вариантом плоской укладки исходного графа.

В алгоритме используются специального вида хромосомы с числом генов, равным $(p + 3)$, причем каждая хромосома в популяции состоит из двух частей. Левая часть хромосомы имеет p разрядов, представляющих собой комбинацию задействованных в данном варианте решения циклов, а правая часть имеет три разряда, несущих информацию о том, какие ребра и сколько раз содержатся в выбранной комбинации (0, 1 или 2 раза соответственно для разрядов $p + 1$, $p + 2$ и $p + 3$). Кроме того, последние три разряда используются при количественной оценке качества полученного решения, для этого вводится в рассмотрение функция $\delta(m)$, которая стремится к ми-

нимуму для генов $p + 1$ и $p + 2$ ($f(H) \rightarrow 0$) и стремится к максимуму для гена $p + 3$ ($f(H) \rightarrow m$):

$$\delta(m) = (2 \times N_{m_p} + 3 + N_{m_p} + 2)/2m,$$

где $N_{m_p} + 3$ — число элементов в $p + 3$ разряде хромосомы, а $N_{m_p} + 2$ — соответственно число элементов в $p + 2$ разряде хромосомы.

Как следует из формулы, значения, которые может принимать функция оценки $\delta(m)$, находятся в интервале $[0, 1]$, причем качество решения будет тем выше, чем большее число ребер (и прежде всего ребер, встречающихся дважды) задействовано в оцениваемом решении. Левая часть хромосомы, в свою очередь, может содержать как значащую, так и незначащую части. Разряды значащей части заполняются номерами циклов, а незначащей — нулями. При этом чем выше качество решения, тем меньше длина незначащей части ($L_0 \rightarrow 0$). Для оптимального решения L_0 будет равна нулю. Таким образом, целевая функция хромосомы полученного решения $f(H)$ является аддитивной функцией двух переменных: $f(H) = (\delta(m); L_0)$, причем $f(H) \rightarrow \max$, когда $\delta(m) \rightarrow 2m$ и $L_0 \rightarrow 0$.

Методику расчета ЦФ хромосомы покажем на следующем примере. Пусть существует некая произвольная комбинация циклов C_1 , C_2 и C_3 , закодированная хромосомой следующего вида:

1	2	3	0	0	0	0	0	0	0	0	{3,9,10,11,12,13,14,15,16}	{4,5,6,7,8}	{1,2}

Здесь 1, 2, 3 — номера циклов, 3, 9–16 — номера неиспользованных ребер, 4–8 — номера ребер, задействованных один раз, а 1, 2 — номера ребер, использованных дважды.

Тогда значение ЦФ данной хромосомы равно $f(H) = (0, 2813; 7)$: $L_0 = 7$; $\delta(m) = (2 \times 2 + 5)/(2 \times 16) = 9/32 = 0,2813$. Решение, имеющее подобное значение ЦФ, соответствует промежуточному решению, поскольку значение ЦФ базового решения не может быть менее 0,5.

Для алгоритма используются два варианта стратегии создания начального базового решения, избираемые в зависимости от характера исходных данных и решения ЛПР.

Первый вариант заключается в том, что в имеющемся множестве циклов C выбираем циклы, в которых ребра повторяются точно два раза. Если это не удастся сделать, то область поиска сужается и просматривается не все множество, а только первые p циклов. Циклы, которым принадлежат найденные в результате такого поиска ребра, суммируются, образуя определенную комбинацию. Эта комбинация проверяется на предмет образования нереальных решений (т. е. наличия ребер, повторяющихся более двух раз). Если такие ребра обнаруживаются, то из комбинации удаляется минимально возможное число циклов, содержащих «лишние» ребра, а полученная комбинация преобразуется в хромосому. Таким образом получается начальное базовое решение.

Второй вариант стратегии создания начального множества решений реализуется по усмотрению ЛПР в случаях, когда не удалось создать начальное базовое решение (т. е. выделить начальный базис циклов, содержащих точно две единицы), либо когда процесс улучшения базового решения зашел

в тупик (локальный оптимум), и в течении нескольких поколений целевая функция базового решения не улучшается. При этом варианте начальное базовое решение формируется случайным образом, путем последовательного объединения циклов, имеющих хотя бы одну одинаковую вершину.

Рассмотрим подробнее первый вариант стратегии создания начального множества решений на примере матрицы циклов графа G , сгенерированного случайным образом (рис. 6.61).

В результате работы алгоритма сформировано множество циклов C_k длины 3 и 4, где $N_c = 25$. В каждом цикле этого множества ищем ребра, повторяющиеся два раза. Так как таких ребер нет ($\forall j \in M)(N_{m_j} > 2)$, то сужаем область поиска и просматриваем первые p элементов множества C . В результате поиска выясняется, что таких ребер четыре. Это ребра номер 3, 7, 13, 16. Теперь суммируем все циклы, включающие указанные ребра. Получаем комбинацию циклов: $\{C_2, C_5, C_6, C_7, C_8, C_9\}$, а затем проверяем, не создает ли полученная комбинация нереальных решений. Поскольку ни в одном столбце сумма единиц не превышает два, данная комбинация является начальным базовым решением и будет закодирована следующей хромосомой:

1н	2	5	6	7	8	9	0	0	0	0	0	0	{2,8,12}	{1,4,6,9,10,11,14,15}	{3,5,7,13,16}

Здесь 1н — первая хромосома начальной популяции.

ЦФ хромосомы 1н равна: $\delta(m) = (2 \times 5 + 8)/32 = 18/32 = 0,5625$; $L_0 = 4$; $f(1н) = (0,5625; 4)$. На данном этапе продолжается процесс заполнения сформированного ранее начального базового решения с помощью генетических операторов селекции, кроссинговера и отбора. Схема работы ГА на этом этапе выглядит следующим образом (рис. 6.64).

Шаг «А». Имеется хромосома, представляющая начальное базовое решение. Теперь необходимо сформировать множество промежуточных решений, из числа которых будет выбран второй родитель для участия в операции скрещивания. Множество промежуточных решений формируется путем попарного объединения циклов, причем объединение производится таким образом, чтобы образующиеся в результате парные ребра были идентичны ребрам, присутствующим в разрядах $p+1$, $p+2$ начального базового решения. Кроме того, номера циклов множества промежуточных решений не должны совпадать с номерами циклов, участвующих в базовом решении. Размер популяции промежуточных решений зависит от числа ребер в разрядах $p+1$, $p+2$ базового решения и от длины хромосомы L (т. е. размерности исследуемого графа). Для рассматриваемого примера множество промежуточных решений формируется случайным образом путем попарного объединения циклов, включающих ребра 2, 8, 12.

Размер начальной популяции примем равным трем (по одной хромосоме для каждого ребра):

2н	1	3	0	0	0	0	0	0	0	0	0	0	{3,5,7,9,10,11,12,13,14,15,16}	{1,4,6,8}	{2}
3н	15	18	0	0	0	0	0	0	0	0	0	0	{1,6,7,10,11,12,14,15}	{2,3,4,5,9,13,16}	{8}
4н	19	21	0	0	0	0	0	0	0	0	0	0	{1,2,3,6,7,13,14,15,16}	{4,5,8,9,11}	{10,12}

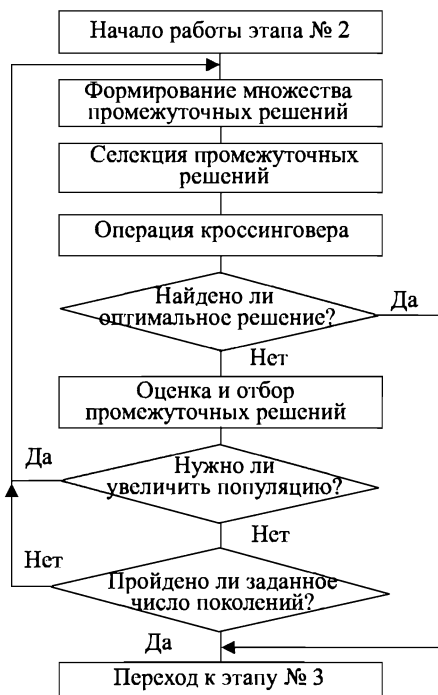


Рис. 6.64. Структурная схема ГА (этап №2)

пригодности промежуточных решений относительно базового решения $f'(II)$. Данный параметр является аддитивной функцией двух переменных $f'(II) = (\delta'(m); N'_c)$, где $\delta'(m)$ — относительное увеличение числа ребер; N'_c — относительное увеличение числа циклов. Количественное значение функции пригодности рассчитывается следующим образом:

а) определяется число совпадений N_{d_1} ребер из разряда $p+3$ оцениваемого промежуточного решения с ребрами из разряда $p+3$ первого родителя. Затем подсчитывается число совпадений N_{d_2} в разрядах $p+3$ промежуточного решения и $p+2$ первого родителя, а также число совпадений N_{d_3} в разрядах $p+2$ промежуточного решения и $p+3$ родителя. Сумма этих чисел характеризует возможное число ребер, которые необходимо будет удалить из базового решения после выполнения кроссинговера. Следовательно, в оптимальных промежуточных решениях значение суммы будет стремиться к нулю;

б) находится число совпадений N_{s_1} ребер из разряда $p+3$ оцениваемого промежуточного решения с ребрами из разряда $p+1$ первого родителя. Затем подсчитывается число совпадений N_{s_2} в разрядах $p+2$ промежуточного решения и $p+1$ первого родителя и, наконец, подсчитывается число совпадений N_{s_3} в разрядах $p+2$ промежуточного решения и $p+2$ родителя. В этом случае определяется число ребер, которые добавляются в базовое ре-

После того, как множество промежуточных решений сформировано, алгоритм переходит на следующий шаг.

Шаг «В». Здесь производится селекция множества промежуточных решений с целью выбора хромосомы. Для оценки каждого решения были разработаны следующие правила:

1. В генах $p+2$ и $p+3$ претендента, по возможности, не должно быть ребер, совпадающих с ребрами из разряда $p+3$ первого родителя.

2. В генах $p+1$ претендента и $p+3$ первого родителя желательно иметь максимальное количество совпадающих ребер.

3. В гене $p+2$ претендента должно быть максимальное число ребер, идентичных ребрам из разряда $p+2$ хромосомы первого родителя.

На основе правил создан комплексный параметр — функция

шение в результате операции кроссинговера, т. е. качество промежуточного решения тем выше, чем больше значение этой суммы;

в) подсчитывается значение $\delta'(m)$ по формуле: $\delta'(m) = 1 - [(2 \times N_{d_1} + N_{d_2} + N_{d_3}) / (2 \times N_{s_1} + N_{s_2} + N_{s_3})]$. Полученное значение зависит от соотношения возможного числа удаляемых и добавляемых ребер и стремится к единице для наилучших промежуточных решений. В случае получения отрицательного значения функции пригодности делается вывод о том, что данное решение непригодно для участия в дальнейших операциях;

г) сравниваются ребра в разрядах $p + 2$ и $p + 3$ промежуточного решения с ребрами в разряде $p + 3$ базового решения и, при наличии совпадений, выявляется минимальное число циклов N_{c-} , которые после выполнения ОК будут удалены из базовой хромосомы для устранения нереальных решений.

То есть, в оптимальном промежуточном решении значение N'_c будет стремиться к нулю;

д) вычисляется значение N'_c по формуле: $N'_c = 1 - (N_{c-} / N_{c+})$. Здесь N_{c+} — число добавляемых циклов, N_{c-} — удаляемых;

е) подсчитывается значение функции пригодности хромосомы относительно текущего базового решения $f'(H)$.

После этого из множества промежуточных решений выбирается элемент, имеющий наилучшую функцию пригодности относительно первого родителя, и назначается в пару к базовому решению для выполнения ОК.

Для рассматриваемого примера подсчитаем значения функций пригодности относительно базового решения (хромосома 1н):

$$\delta'(m)_{2н} = 1 - [(0 + 0) / (2 \times 1 + 4)] = 1; \quad N'_c(2н) = 1 - (0/2) = 1; \quad f'(2н) = (1; 1).$$

$$\delta'(m)_{3н} = 1 - [(0 + 4) / (2 \times 1 + 1 + 2)] = 1 - (4/5) = 1 - 0,8 = 0,2; \\ N'_c(3н) = 1 - (2/2) = 1 - 0 = 0; \quad f'(3н) = (0,2; 0);$$

$$\delta'(m)_{4н} = 1 - [(2 \times 1 + 1) / (2 \times 1 + 1 + 3)] = 1 - (3/6) = 1 - 0,5 = 0,5; \\ N'_c(4н) = 1 - (2/2) = 0; \quad f'(4н) = (0,5; 0).$$

После ранжирования полученных функций пригодности промежуточных решений в качестве второго родителя будет выбрана хромосома 2н, как имеющая наилучшую оценку пригодности.

Шаг «С». На этом шаге выполняется ОК двух родительских особей. Здесь используется двухточечный суммирующий ОК. Точки скрещивания в операторе выбираются следующим образом:

1. В первом родителе первая точка скрещивания t_1 находится за последним значащим (ненулевым) элементом h_i хромосомы, а позиция второй точки t_2 выбирается после элемента $h_i + k$, где k — число значащих элементов во второй родительской хромосоме.

2. Во втором родителе первая точка находится перед первым элементом хромосомы, а вторая точка определяется так же, как в случае с первым родителем.

В результате выполнения ОК происходит объединение значащих частей родительских хромосом. В случае, если в функции пригодности второго родителя $\delta'(m) \neq 0$, то из хромосомы потомка удаляются ребро (или ребра), которые повторяются в данной комбинации более двух раз. Параллельно с удалением ребер удаляются N'_c циклов, которым принадлежит данное ребро. Данный цикл (циклы) вместе с принадлежащими им ребрами заносятся в хромосому второго родителя. При этом вновь появившиеся в хромосоме циклы не могут быть удалены. В примере точки ОК для хромосомы 1н: $t_1 = 6, t_2 = 8$, и для хромосомы 2н: $t_1 = 0, t_2 = 2$.

1н	2	5	6	7	8	9	0	0	0	0	{2,8,12}	{1,4,6,9,10,11,14,15}	{3,5,7,13,16}
2н	1	3	0	0	0	0	0	0	0	0	{3,5,7,9,10,11,12,13,14,15,16}	{1,4,6,8}	{2}

После выполнения ОК получается новое базовое решение:

1п	2	5	6	7	8	9	1	3	0	0	{12}	{8,9,10,11,14,15}	{1,2,3,4,5,6,7,13,16}

Шаг «Д». Здесь популяция промежуточных решений переупорядочивается. Для этого производится пересчет функций пригодности каждого элемента популяции относительно нового базового решения. В случае, если после произведенного пересчета функции пригодности всех элементов имеют низкие оценки ($f'(p_i) = (\delta'(m) < 25\% \text{ и } N'_c < 0)$), то необходимо либо расширить размер популяции промежуточных значений, либо перейти к следующему этапу строительства. Решение принимается ЛПР на основании оценки размера популяции и числа пройденных поколений. Расширение размера популяции производится за счет добавления промежуточных решений, включающих пары ребер, идентичных ребрам из разряда $p+3$ базовой хромосомы. Если получено базовое решение, в котором число ребер в разряде $p+1$: $N_{m_{p+1}} = 0$, а в разряде $p+2$: $N_{m_{p+2}} \neq 0$, алгоритм также переходит к следующему этапу.

Поскольку размер популяции промежуточных решений в рассматриваемом примере изначально мал и число пройденных поколений ($N_G = 1$) невелико, увеличиваем популяцию промежуточных решений за счет добавления хромосом, содержащих ребро 12, так как в разряде $p+1$ базового решения осталось только одно это ребро:

5н	19	24	0	0	0	0	0	0	0	0	{1,2,3,6,7,8,9,11,14,16}	{4,5,13,15}	{10,12}
6н	21	25	0	0	0	0	0	0	0	0	{1,2,3,4,5,6,7,13,16}	{8,9,10,11,14,15}	{12}
7н	19	25	0	0	0	0	0	0	0	0	{1,2,3,6,7,8,9,13,16}	{4,5,10,11,14,15}	{12}
8н	21	24	0	0	0	0	0	0	0	0	{1,2,3,4,5,6,7,11,14,16}	{8,9,13,15}	{10,12}

Подсчитаем функции пригодности новых промежуточных решений:

$$\begin{aligned} \delta'(m)_{5н} &= 0,2; & N'_c(5н) &= 0; & f'(5н) &= (0,2; 0); \\ \delta'(m)_{6н} &= 1; & N'_c(6н) &= 1; & f'(6н) &= (1; 1); \end{aligned}$$

$$\delta'(m)_{7n} = 0,6667; \quad N'_c(7n) = 0; \quad f'(7n) = (0,6667; 0);$$

$$\delta'(m)_{8n} = 0,6; \quad N'_c(8n) = 0; \quad f'(8n) = (0,6; 0).$$

Лучшую оценку пригодности имеет хромосома 6n, которая будет выбрана в качестве второго родителя для ОК. Теперь возвращаемся к этапу «С» и выполняем ОК для хромосом 1n и 6n:

1п	2	5	6	7	8	9	1	3	0	0	{12}	{8,9,10,11,14,15}	{1,2,3,4,5,6,7,13,16}
6н	21	25	0	0	0	0	0	0	0	0	{1,2,3,4,5,6,7,13,16}	{8,9,10,11,14,15}	{12}
2п	2	5	6	7	8	9	1	3	21	25	{0}	{0}	{1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16}

В полученной хромосоме 2n все разряды с 1-го по p -й (т. е. с 1-го по 10-й) являются значащими, а разряды $p+1$ и $p+2$ — пустые. Следовательно, условие планарности графа выполнено. Таким образом, в результате выполнения ОК построено оптимальное базовое решение (хромосома 2п).

После декодирования хромосомы будет сформирован необходимый базис циклов B_p (рис. 6.65). Следовательно, данный граф может быть уложен на плоскости без пересечений.

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
C_1	1	1	0	0	0	1	0	0	0	0	0	0	0	0	0	0
C_2	1	0	0	0	1	0	1	0	0	0	0	0	0	0	0	0
C_3	0	1	0	1	0	0	0	1	0	0	0	0	0	0	0	0
C_5	0	0	1	1	0	0	0	0	0	0	0	0	1	0	0	0
C_6	0	0	1	0	1	0	0	0	0	0	0	0	0	0	1	0
C_7	0	0	0	0	0	1	1	0	1	0	0	0	0	0	0	0
C_8	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	1
C_9	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	1
C_{21}	0	0	0	0	0	0	0	1	1	1	0	1	0	0	0	0
C_{25}	0	0	0	0	0	0	0	0	0	0	1	1	0	1	1	0

Рис.6.65 Базис циклов B_p

Задача алгоритма на этапе 3 состоит в том, чтобы при имеющемся квазиоптимальном решении поиск оптимального решения вести, не ухудшая качества имеющегося. Для этого популяция промежуточных решений расщепляется таким образом, чтобы каждая хромосома новой популяции содержала в себе только один цикл, после чего выполняются действия, идентичные шагам «В»–«D» этапа 2. При построении оптимального решения алгоритм завершает работу, в противном случае переходит к этапу 4. Переход к этапу 4 осуществляется, если в ходе решения задачи получено квазиоптимальное решение, не улучшающееся на протяжении значительного числа поколений (попадание в локальный оптимум).

В алгоритме за основу был взят стандартный оператор — мутация обмена. Для проведения ОМ случайным образом выбираются два гена, один из которых является значащим элементом, а второй — незначащим. В ходе вы-

полнения ОМ элементы меняются местами, после чего значащий элемент обнуляется, все принадлежащие данному циклу ребра удаляются из генов $p+2$ и $p+3$ и, при необходимости, оставшиеся значащие элементы сдвигаются вправо, вытесняя нулевой элемент из значащей части. Например, для хромосомы 1п ОМ выполняется следующим образом. В хромосоме случайным образом выбираем два гена, один из которых значащий, а второй нулевой. Пусть это будут гены 3 и 10:

1п

2	5	6	7	8	9	1	3	0	0

 {12} {8,9,10,11,14,15} {1,2,3,4,5,6,7,13,16}

После этого производим обмен:

1п

2	5	0	7	8	9	1	3	0	6

 {12} {8,9,10,11,14,15} {1,2,3,4,5,6,7,13,16}

Из генов $p+2$ и $p+3$ удаляем ребра принадлежащие циклу 6.

1пм

2	5	0	7	8	9	1	3	0	0

 {12,15} {3,5,8,9,10,11,14} {1,2,4,6,7,13,16}

Сдвигаем значащие элементы влево, после чего ОМ заканчивает свою работу:

1пм

2	5	0	7	8	9	1	3	0	0

 {12,15} {3,5,8,9,10,11,14} {1,2,4,6,7,13,16}

Здесь 1пм — первое решение потомок после мутации.

Схема проведения оператора инверсии аналогична ОМ с той лишь разницей, что для обмена выбирается не один ген хромосомы, а несколько. Число обмениваемых генов задается случайным образом, однако оно не должно быть больше числа незначащих разрядов инвертируемой хромосомы. Вероятности проведения операций мутации $P(OM)$ и инверсии $P(OI)$ выбираются случайным образом на интервале $[0,1]$, причем их суммарная вероятность $P(SUM) = P(OM) + P(OI) = 1$.

Для построения плоской укладки по сформированному базису циклов используем принцип, изложенный в [200]. Согласно ему плоская укладка строится следующим образом: в базисе p последовательно выбираются цикл за циклом, причем на каждом шаге разрастание происходит за счет включения новых ребер во внешний цикл. Таким образом, на каждом шаге внешний цикл либо удлиняется если в новом цикле число новых, незадействованных ребер больше числа повторяющихся, либо укорачивается в противном случае. При этом повторяющиеся ребра автоматически выпадают из внешнего цикла и оказываются внутри.

На каждом $(i+1)$ -м шаге находим разность множеств $C'_{ВН_i} = C_{ВН_i} \setminus C_{i+1}$ и $C'_{i+1} = C_{i+1} \setminus C_{ВН_i}$, где $C_{ВН_i}$ — внешний цикл графа, после чего выполняем суммирование $C_{ВН_{i+1}} = C'_{ВН_i} + C'_{i+1}$. Работа алгоритма плоской укладки заканчивается после того, как все ребра графа будут пройдены, как минимум, один раз. Покажем работу алгоритма плоской укладки для графа G (рис. 6.61). Выбираем первый цикл в подматрице p . Это цикл C_1 . На первом шаге цикл C_1 совпадает с внешним циклом $C_{ВН_1}$. $C_{ВН_1} = \{1(1,2) - 6(2,3) - 2(3,1)\}$. Затем выбираем следующий цикл в базисе p . Это цикл C_2 . Находим разность множеств:

$$C'_{ВН_1} = C_{ВН_1} \setminus C_2 = \{6(2,3), 2(3,1)\}, \quad C'_2 = C_2 \setminus C_{ВН_1} = \{5(1,8), 7(8,2)\}.$$

После суммирования $C'_{\text{ВН}_1}$ и C'_2 получим:

$$C_{\text{ВН}_2} = \{5(1,8) - 7(8,2) - 6(2,3) - 2(3,1)\}.$$

Далее процесс продолжается аналогичным образом:

$$C_{\text{ВН}_3} = \{5(1,8) - 7(8,2) - 6(2,3) - 8(3,6) - 4(6,1)\};$$

$$C_{\text{ВН}_4} = \{5(1,8) - 7(8,2) - 6(2,3) - 8(3,6) - 13(6,5) - 3(5,1)\};$$

$$C_{\text{ВН}_5} = \{7(8,2) - 6(2,3) - 8(3,6) - 13(6,5) - 15(5,8)\};$$

$$C_{\text{ВН}_6} = \{9(8,3) - 8(3,6) - 13(6,5) - 15(5,8)\}.$$

На следующем шаге алгоритма должен быть выбран цикл C_8 , однако в нем нет ни одного ребра, принадлежащего внешнему циклу. Поэтому временно пропустим его и перейдем к циклу C_9 :

$$C_{\text{ВН}_9} = \{9(8,3) - 8(3,6) - 16(6,7) - 14(7,5) - 15(5,8)\}.$$

Теперь во внешнем цикле появилось ребро 16, соединяющее вершины графа (6,7) и возвращаемся к циклу C_8 :

$$C_{\text{ВН}_8} = \{9(8,3) - 8(3,6) - 10(6,4) - 11(4,7) - 14(7,5) - 15(5,8)\};$$

$$C_{\text{ВН}_9} = \{12(8,4) - 11(4,7) - 14(7,5) - 15(5,8)\}.$$

Алгоритм плоской укладки завершил работу, так как все ребра графа пройдены. В результате получена плоская укладка исходного графа G (рис. 6.66).

Рассмотрим теперь применение описанного алгоритма для решения задачи минимизации пересечений в непланарном графе на примере сгенерированного случайным образом графа G' (рис. 6.67). Списки смежности данного графа будут выглядеть следующим образом:

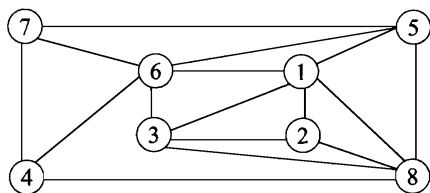


Рис. 6.66. Плоская укладка графа G

$$Lx_1 = \{1(1,2), 2(1,3), 3(1,4), 4(1,9)\}; \quad Lx_2 = \{5(2,4), 6(2,8)\};$$

$$Lx_3 = \{7(3,5), 8(3,6), 9(3,3)\}; \quad Lx_4 = \{10(4,6), 11(4,7), 12(4,9)\};$$

$$Lx_5 = \{13(5,6), 14(5,7)\}; \quad Lx_6 = \{15(6,8)\}; \quad Lx_7 = \{16(7,8), 17(7,9)\}.$$

После работы блока генерации циклов будет сформировано следующее множество циклов длины три:

$$C_1 = (1 - 5 - 3); \quad C_2 = (2 - 9 - 4); \quad C_3 = (3 - 12 - 4);$$

$$C_4 = (7 - 13 - 8); \quad C_5 = (11 - 17 - 12).$$

Поскольку $N_c < p$, то продолжаем генерацию циклов длины четыре:

$$C_6 = (1 - 5 - 12 - 4); C_7 = (2 - 8 - 10 - 3); C_8 = (2 - 9 - 12 - 3);$$

$$C_9 = (3 - 11 - 17 - 4); C_{10} = (5 - 10 - 15 - 6);$$

$$C_{11} = (5 - 11 - 16 - 6); C_{12} = (7 - 14 - 17 - 9);$$

$$C_{13} = (8 - 10 - 12 - 9); C_{14} = (10 - 13 - 14 - 11);$$

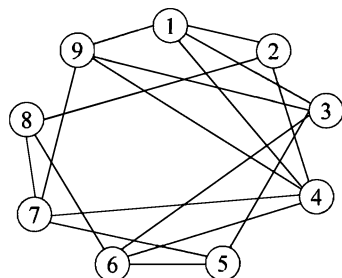
$$C_{15} = (10 - 15 - 16 - 11); C_{16} = (13 - 15 - 16 - 14).$$

В результате общее число сгенерированных циклов N_c равно 16. Пе-

реходим к блоку анализа (п. 2) и сравниваем: $N_c > p$; $N_{m_j} > 2$. Условие выполнено, переходим к пункту 3. Выбираем первые 10 циклов и подсчитываем число ребер: $5 \times 3 + 5 \times 4 = 35$; $2 \times m = 2 \times 17 = 34$; $35 > 34$. Следовательно, граф непланарный. Для минимизации числа пересечений ребер данного графа переходим к блоку генетических операторов.

Создание начального базового решения выполним по второму варианту стратегии. Для этого последовательно просматриваем имеющиеся списки и выбираем

Рис. 6.67. Граф G' , заданный случайным образом



попарно циклы, имеющие как минимум одно общее ребро, а затем суммируем их. В результате получится следующее множество начальных решений:

1н	1	6	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	{2,6-11,13-17}	{3,4,12}	{1,5}
2н	2	7	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	{1,5,6,7,11-17}	{3,4,8,9,10}	{2}
3н	4	12	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	{1-6,10-12,15,16}	{8,9,13,14,17}	{7}
4н	5	14	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	{1-9,15,16}	{10,12,13,14,17}	{11}

Просуммировав их, получим начальное базовое решение:

5н	1	6	2	7	4	12	5	14	0	0	0	0	0	0	0	0	0	0	0	{6,15,16}	{0}	{1-5,7-14,17}

Целевая функция полученного базового начального решения равна: $L_0 = 2$; $\delta(m) = (2 \times 14 + 0) / (2 \times 17) = 28/34 = 0,8235$; $f(H) = (0,8235; 2)$. Теперь необходимо сформировать популяцию промежуточных решений. Поскольку в начальном базовом решении отсутствуют 6 ребер, примем размер популяции равным 6:

6н	10	11	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	{1-4,7-9,12,13,14,17}	{10,11,15,16}	{5,6}
7н	15	16	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	{1-9,17}	{10,11,13,14}	{15,16}
8н	10	15	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	{1-4,7-9,12-14,17}	{5,6,11,16}	{10,15}

9н	11	16	0	0	0	0	0	0	0	0	0	{1-4,7-10,12,17}	{5,6,11,13,14,15}	{16}
10н	10	16	0	0	0	0	0	0	0	0	0	{1-4,7-9,11,12,17}	{5,6,10,13,14,16}	{15}
11н	11	15	0	0	0	0	0	0	0	0	0	{1-4,7-9,12-14,17}	{5,6,10,15}	{11,16}

Теперь вычисляем функции пригодности полученных хромосом:

$$\begin{aligned}
 \delta'(m)_{6н} &= 0; & N'_c(6н) &= -0,5; & f'(6н) &= (0; -0,5); \\
 \delta'(m)_{7н} &= 0; & N'_c(7н) &= 0,5; & f'(7н) &= (0; 0,5); \\
 \delta'(m)_{8н} &= 0; & N'_c(8н) &= -0,5; & f'(8н) &= (0; -0,5); \\
 \delta'(m)_{9н} &= 0; & N'_c(9н) &= 0; & f'(9н) &= (0; 0); \\
 \delta'(m)_{10н} &= 0; & N'_c(10н) &= 0; & f'(10н) &= (0; 0); \\
 \delta'(m)_{11н} &= 0; & N'_c(11н) &= -0,5; & f'(11н) &= (0; -0,5).
 \end{aligned}$$

Согласно полученным функциям пригодности хромосомы 6, 8, 11 удаляются из популяции. Из трех оставшихся хромосом наибольшее значение функции пригодности имеет хромосома 7 ($f'(7н) = (0; 0,5)$), которая и будет отобрана для проведения ОК:

5н	1	6	2	7	4	12	5	14	0	0	{6,15,16}	{0}	{1-5,7-14,17}
7н	15	16	0	0	0	0	0	0	0	0	{1-9,17}	{10,11,13,14}	{15,16}
1п	1	6	2	7	4	12	5	14	15	16	{6}	{0}	{1-5,7-10,11,13,14-17}

В полученной хромосоме (1п) есть четыре ребра (10, 11, 13, 14), повторяющиеся трижды. Для того, чтобы избежать появления нереального решения достаточно удалить один цикл 14. Тогда базовое решение примет вид:

1п	1	6	2	7	4	12	5	15	16	0	{6}	{0}	{1-5,7-17}

Определим ЦФ полученного базового решения: $L_0 = 1$; $\delta(m) = (2 \times \times 16)/(2 \cdot 17) = 28/38 = 0,9412$; $f(H) = (0,9412; 1)$. Как видно, значение ЦФ базового решения улучшилось. Полученное значение $f(H)$ позволяет считать, что найдено некоторое квазиоптимальное решение. Однако базис циклов сформирован не полностью. Продолжим процесс исследования. Для этого применяется ОМ. Для проведения ОМ случайным образом выбираем разряд 7 в хромосоме базового решения и обнуляем его:

1п	1	6	2	7	4	12	0	15	16	0	{6}	{0}	{1-5,7-17}

Затем из правой части хромосомы удаляем ребра, соответствующие циклу 5:

2п	1	6	2	7	4	12	15	16	0	0	{6}	{11,12,17}	{1-5,7-10,13-16}

Для полученной хромосомы (2п), используя стратегию доработки базового решения, сформируем множество промежуточных решений. Это мно-

жество решений, состоящих из одного цикла:

1 _{H2}	3	0	0	0	0	0	0	0	0	0	{1,2,5-11,13-17}	{3,4,12}	{0}
2 _{H2}	8	0	0	0	0	0	0	0	0	0	{1,5-8,10,11,13-17}	{2,3,9,12}	{0}
3 _{H2}	9	0	0	0	0	0	0	0	0	0	{1,2,5-10,12-16}	{3,4,11,17}	{0}
4 _{H2}	10	0	0	0	0	0	0	0	0	0	{1,-4,7-9,11-14,16,17}	{5,6,10,15}	{0}
5 _{H2}	11	0	0	0	0	0	0	0	0	0	{1,-4,7-10,12-15,17}	{5,6,11,16}	{0}
6 _{H2}	13	0	0	0	0	0	0	0	0	0	{1-7,11,13-17}	{8,9,10,12}	{0}

Следовательно, построенное базовое решение является оптимальным для данного графа и улучшено быть не может.

Построим теперь максимально планарную укладку данного графа. В результате работы ГА сформирован следующий базис циклов:

$$B_p = \{C_1, C_6, C_2, C_7, C_4, C_{12}, C_5, C_{15}, C_{16}\}.$$

Выбираем первый цикл базиса C_1 :

$$C_{BH_1} = (1(1,2) - 5(2,4) - 3(4,1)); \quad C_{BH_2} = (4(1,9) - 12(9,4) - 3(4,1));$$

$$C_{BH_3} = (2(1,3) - 9(3,9) - 12(9,4) - 3(4,1));$$

$$C_{BH_4} = (9(3,9) - 12(9,4) - 10(4,6) - 8(6,3));$$

$$C_{BH_5} = (9(3,9) - 12(9,4) - 10(4,6) - 13(6,5) - 7(5,3));$$

$$C_{BH_6} = (12(9,4) - 10(4,6) - 13(6,5) - 14(5,7) - 17(7,9));$$

$$C_{BH_7} = (10(4,6) - 13(6,5) - 14(5,7) - 11(7,4));$$

$$C_{BH_8} = (13(6,5) - 14(5,7) - 16(7,8) - 15(8,6)).$$

Алгоритм плоской укладки завершил работу. В результате работы получена укладка максимальной планарной части исходного графа G' (рис. 6.68).

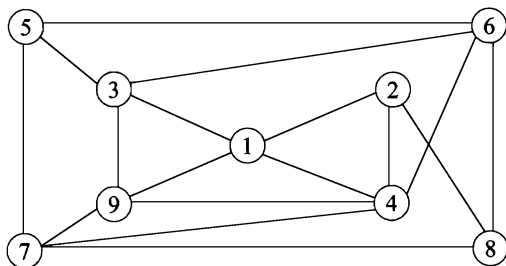


Рис. 6.68. Максимальная планарная укладка графа G'

Для алгоритма определения планарности в классической постановке временная и пространственная сложности определяются как функции от n, m . Временная сложность ГА складывается из временных сложностей отдель-

ных операторов. Временная сложность алгоритма селекции, в худшем случае, составляет $O(N_p + 2 \log 2N_p)$. Временные сложности ОК и ОМ составляют $O(L)$.

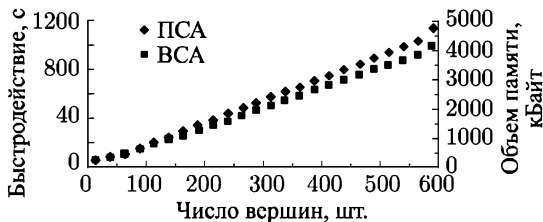


Рис. 6.69. Зависимость временной и пространственной сложности алгоритма от числа вершин

Здесь ПСА — пространственная сложность алгоритма, определяемая объемом оперативной памяти при работе алгоритма. В течение одного поколения выполняется один оператор селекции и один ОК. ОМ выполняются не чаще одного раза в 4–5 поколений. Необходимо учесть, что при начальной генерации популяции формирование одной хромосомы требует L операций для определения генов. После генерации начальной популяции необходимо выполнить ее сортировку, что требует $N_p \log 2N_p$ операций. То есть, временная сложность генетического алгоритма составит: $O((n + m) + 2m + [N_p \log 2N_p + (N_p + 2 \log 2N_p) + O(L) + O(L)/5]N_G)$, где N_G — число поколений, N_p — размер популяции. Следовательно, временная сложность алгоритма определения планарности пропорциональна $O(n + m)$. Зависимость временной и пространственной сложности от числа вершин графа приведена на рисунке 6.69.

6.9. Определение изоморфизма графов

Большое число комбинаторно-логических инженерных задач на графах требует установления изоморфизма или изоморфного вложения между заданными структурами [124, 159, 160]. Данная проблема, как и все остальные рассмотренные выше проблемы, является NP-полной. Поэтому разрабатываются различные эвристики для получения приемлемых практических результатов.

Задача распознавания изоморфизма графов состоит в следующем. Для заданных графов $G_1 = (X_1, U_1)$ и $G_2 = (X_2, U_2)$ требуется определить, существует ли взаимно однозначное отображение $\varphi: X_1 \rightarrow X_2$ такое, что $U_1 = (x, y) \in U$ тогда и только тогда, когда $(\varphi(x), \varphi(y)) \in U_2$ [159].

В настоящее время известны полиномиальные алгоритмы для следующих классов графов: графы ограниченной степени; графы с ограниченной кратностью собственных значений; k -разделимые графы; k -стягиваемые графы и так далее [188–192].

Особое внимание заслуживают сильнорегулярные графы. Граф $G = (X, \rho, n_{11}^1, n_{11}^0)$, $|X| = n$, называется n -вершинным регулярным графом степени ρ , если в нем любая пара смежных вершин имеет n_{11}^1 общих соседей, а любая пара несмежных вершин — n_{11}^0 . Известно [188], что сильнорегулярные графы образуют класс наиболее трудных задач для распознава-

ния изоморфизма графов. Для сильнорегулярных графов известен алгоритм с временной сложностью $O(\log n^2)$.

Существует ряд задач, которые полиномиально сводятся друг к другу и к задаче распознавания изоморфизма графов G_1 и G_2 [188]:

1. Распознавание изоморфизма графов и построение порождающего множества для группы автоморфизмов графа, где $G = G_1 \cup G_2$.
2. Распознавание изоморфизма графов и существование изоморфной подстановки (временная сложность алгоритма $\approx O(n^2)$).
3. Распознавание изоморфизма графов и число симметрии графа (временная сложность алгоритма $\approx O(n^2)$).
4. Распознавание изоморфизма графов и автоморфное разбиение множества вершин графа (временная сложность алгоритма $\approx O(n^2)$).
5. Распознавание изоморфизма графов и число изоморфизмов графов (временная сложность алгоритма $\approx O(n^2)$).
6. Распознавание изоморфизма графов и определение существования автоморфизма для пары фиксированных вершин графа G (временная сложность алгоритма $\approx O(n^2)$).

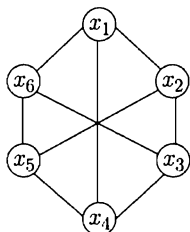
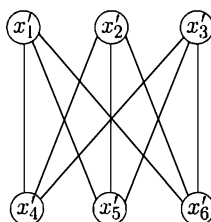
Известно, что задача изоморфного вложения графа также является NP-полной задачей [161]. Она имеет много сходства и в то же время существенно (по сложности) отличается от задачи распознавания изоморфизма графов. Например, для решения задачи изоморфизма подграфа G_1 подграфу G_2 графа G с использованием известных алгоритмов распознавания изоморфизма графов необходимо разработать процедуру выделения в графе G подмножества $X_1 \subset X$, равномошного с множеством вершин X_2 графа G_2 .

Данная процедура включает k_1 действий, где $k_1 = \binom{n}{n_2}$, $n = |X|$, $n_2 = |X_2|$. Следовательно k_1 раз надо применять и алгоритм распознавания изоморфизма графов. Поэтому при выделении каждого подграфа G_1 в графе G необходимо выполнять k_2 действий, где $k_2 = \binom{m_1}{m_2}$ (m_1 — количество ребер в подграфе G_1 , m_2 — в графе G_2 , $m_1 > m_2$). Следовательно, даже если есть полиномиальный алгоритм распознавания изоморфизма графов, с его помощью невозможно решить задачу изоморфного вложения за полиномиальное время. Она может быть решена за полиномиальное время, если G_1 — лес, а G_2 — дерево.

Наибольшую трудность представляет установление изоморфизма однородных графов, имеющих автоморфные подграфы. Для решения таких задач используются методы разбиения исследуемых графов на различные уровни. При этом временная сложность алгоритма снижается с $O(n!)$ до $O(k!)$, где n — число элементов в графе, а k — число элементов в наибольшем автоморфном подграфе, т. е. в таком подграфе, где не имеет значения выбор вершин для установления соответствия.

Основная идея таких алгоритмов заключается в следующем [18]. В графах, исследуемых на изоморфизм, выбираются две предполагаемо изоморфные вершины. Относительно них производится разбиение всех остав-

шихся вершин на два класса. В первый класс включаются вершины, смежные предполагаемо изоморфным, а во второй нет.

Рис. 6.70. а. Граф G Рис. 6.70. б. Граф G'

Например, на рис. 6.70, а, б показаны два графа G, G' . Пусть $G = (X, U)$, $G' = (X', U')$, $|X| = |X'| = 6$, $|U| = |U'| = 9$, локальные степени всех вершин графа равны трем. Необходимо установить изоморфизм графов G и G' . Другими словами, необходимо определить, существует ли отношение эквивалентности:

$X \Leftrightarrow X', U \Leftrightarrow U'$ такое, что
если ребро $(x_i, x_j) \Leftrightarrow$ ребру (x'_i, x'_j) ,
тогда $x_i \Leftrightarrow x'_i, x_j \Leftrightarrow x'_j, (x_i, x_j) \in X, (x'_i, x'_j) \in X'$.

Покажем на примере реализацию эвристики разбиения для установления изоморфизма однородных графов. Выберем одну вершину в графе G и одну в графе G' . И относительно них выполним указанное выше разбиение:

$$\begin{array}{lll} \{1\} & \{2, 4, 6\}_{1+} & \{3, 5\}_{1-} \quad (G) \\ \{1'\} & \{4', 5', 6'\}_{1'+} & \{2', 3'\}_{1'-} \quad (G') \end{array}$$

Знак $(1+)$ означает, что в данном разбиении вершины x_2, x_4, x_6 смежны вершине x_1 , а знак $(1-)$ означает, что вершины x_3, x_5 не смежны вершине x_1 графа G . Аналогичное разбиение выполнено для графа G' . Здесь считается, что вершины x_1 и x'_1 предполагаемо изоморфны (П-изоморфны). Далее, выбираются подмножества меньшей мощности и внутри них проверяется смежность вершин. В нашем примере вершины x_3, x_5 и x'_2, x'_3 не смежны. Поэтому процесс разбиения продолжается:

$$\begin{array}{llll} \{1\} & \{3\}_{1-} & \{\{2, 4, 6\}_{3+}\}_{1+} & \{\{5\}_{3-}\}_{1-} \\ \{1'\} & \{2'\}_{1'-} & \{\{4', 5', 6'\}_{2'+}\}_{1'+} & \{\{3'\}_{2'-}\}_{1'-} \end{array}$$

Продолжая процесс аналогично, получим, что граф G изоморфен графу G' . Подстановка вершин запишется

$$t = \left\{ \begin{array}{ccccc} 1 & 2 & 4 & 6 & 3 & 5 \\ 1' & 4' & 5' & 6' & 2' & 3' \end{array} \right\}$$

В рассматриваемом примере подмножества вершин $\{x_2, x_4, x_6\}$ и $\{x'_2, x'_5, x'_6\}$ являются автоморфными, т. е. каждая вершина в одном подгра-

фе может быть изоморфна любой вершине второго подграфа. В примере (рис. 6.70) временная сложность алгоритма распознавания изоморфизма графов $O(6!)$ была сведена к временной сложности алгоритма распознавания изоморфизма графов $O(3!)$. В алгоритмах такого типа необходим перебор между элементами автоморфных подграфов, причем, как следует из рассмотренного примера, такой перебор здесь выполняется последовательно для всех вершин, кроме последней в подграфе. В этой связи применение новых архитектур генетического поиска с адаптацией, совместные эволюции Ч. Дарвина, Ж. Ламарка, де Фриза и К. Поппера и локальный поиск позволяют повысить скорость установления изоморфизма графов.

Предлагается в рассматриваемом методе распознавания изоморфизма графов после получения разбиения в подмножествах наибольшей мощности выполнять все модифицированные генетические операторы. Наилучшие результаты дало применение новых модифицированных генетических операторов, основанных на жадной стратегии, метода дихотомии, минимального кластера, метода золотого сечения и метода Фибоначчи. Это позволяет параллельно анализировать все подмножества автоморфных вершин и повышает время нахождения результата.

Например, на рис. 6.71, *а*, *б* показаны нетривиальные, неоднородные графы G и G' .

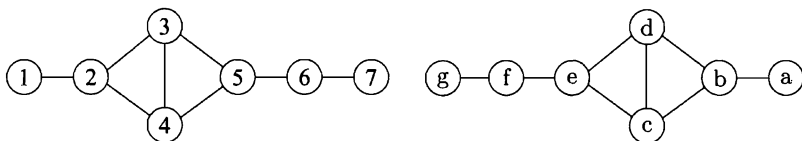


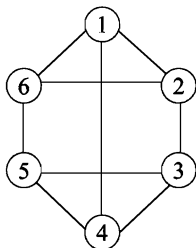
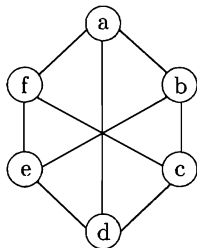
Рис. 6.71. *а*. Нетривиальный граф G Рис. 6.71. *б*. Нетривиальный граф G'

В этих графах $|X| = |X'| = 7$, $|U| = |U'| = 8$, $\rho_1 = 1$, $\rho_2 = 3$, $\rho_3 = 3$, $\rho_4 = 3$, $\rho_5 = 3$, $\rho_6 = 2$, $\rho_7 = 1$; $\rho_a = 1$, $\rho_b = 3$, $\rho_c = 3$, $\rho_d = 3$, $\rho_e = 3$, $\rho_f = 2$, $\rho_g = 1$. Вершина 1 может быть Π -изоморфна вершине g или a . В первом случае нас ждет тупик, а во втором — успех. Мы предположили, что вершина 1 Π -изоморфна вершине a . Тогда на основе связности графа и эвристики разбиения следует, что вершина 7 Π -изоморфна вершине g . После этого вытекает, что вершина 6 Π -изоморфна вершине f . Далее необходимо установить, существует ли соответствие между подмножествами автоморфных вершин $\{2, 3, 4, 5\}$ и $\{b, c, d, e\}$. В этом случае, как отмечалось выше, в общем необходимо выполнить полный перебор между этими подмножествами вершин. Для этих подмножеств это — $4!$. Выполняя такие преобразования получим, что вершина 2 Π -изоморфна вершине b , вершина 3 Π -изоморфна вершине d , 4 — c и 5 — e . Отсюда следует, что графы G и G' (рис. 6.71, *а*, *б*) изоморфны, а подстановка изоморфизма, переводящая один граф в другой, запишется:

$$t = \left\{ \begin{array}{ccccccccc} 1 & 2 & 3 & 4 & 5 & 6 & 7 \\ a & b & d & c & e & f & g \end{array} \right\}$$

Как видно, наличие вершин с различными локальными степенями упрощает процесс распознавания изоморфизма графов. Это позволяет выбирать начальные условия для эвристики разбиения. В примере (рис. 6.71, а, б) всего два способа для выбора разбиения, в то время, как в графах (рис. 6.70, а, б) шесть возможных способов разбиения. В однородных графах все степени вершин одинаковы, поэтому начальные условия выбираются произвольно случайным образом. В этой связи при установлении изоморфизма в однородных графах временная сложность алгоритма в самом лучшем случае равна $O(n)$, в самом худшем случае — $O(n!)$. Если графы неизоморфны, то за одну итерацию установить результат невозможно. Необходимо провести сравнение на изоморфизм одной случайно выбранной вершины из одного графа со всеми остальными вершинами другого графа.

Например, пусть заданы два графа (рис. 6.72, а, б) G и G' . При этом $G = (X, U)$, $G' = (X', U')$, $|X| = |X'| = 6$, $|U| = |U'| = 9$, локальные степени всех вершин графа равны трем.

Рис. 6.72, а. Граф G Рис. 6.72, б. Граф G'

Для распознавания изоморфизма графов применим эвристику разбиения. Предположим, что вершина 1 графа G Π -изоморфна вершине a графа G' . Тогда получим:

$$\{1\} \quad \{2, 4, 6\}_{1+} \quad \{3, 5\}_{1-} \quad (G)$$

$$\{a\} \quad \{b, d, f\}_{a+} \quad \{c, e\}_{a-} \quad (G')$$

Для дальнейшего анализа выбираем соответствующие подмножества наименьшей мощности $\{3, 5\}$ и $\{c, e\}$. Вершины $(3, 5)$ в графе G (рис. 6.72, а) смежны, а вершины (c, e) в графе G' — нет. Следовательно, вершина 1 не может быть изоморфна вершине a . В этой связи необходимо провести аналогичные операции для проверки изоморфизма вершины 1 с остальными вершинами графа G' . Далее получим:

$$\{1\} \quad \{2, 4, 6\}_{1+} \quad \{3, 5\}_{1-} \quad (G)$$

$$\{b\} \quad \{a, c, e\}_{b+} \quad \{d, f\}_{b-} \quad (G')$$

Вершины $(3, 5)$ в графе G (рис. 6.72, а) смежны, а вершины (d, f) в графе G' — нет. Следовательно, вершина 1 не может быть изоморфна вершине b . Продолжая аналогичные построения, имеем, что вершина 1 не может быть изоморфна вершинам c, d, e . Наконец, предположим, что вершина 1 графа G

П-изоморфна вершине f графа G' . Тогда получим:

$$\{1\} \quad \{2, 4, 6\}_{1+} \quad \{3, 5\}_{1-} \quad (G)$$

$$\{f\} \quad \{a, c, e\}_{f+} \quad \{b, d\}_{f-} \quad (G')$$

Вершины $(3, 5)$ в графе G (рис. 6.72, *а*) смежны, а вершины (b, d) в графе G' — нет. Следовательно, вершина 1 не может быть изоморфна вершине b . Проанализированы все вершины графа G' на предмет изоморфизма с вершиной 1 графа G . Во всех случаях ответ отрицательный. Поэтому граф G не изоморфен графу G' .

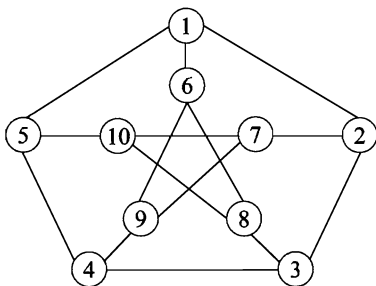


Рис. 6.73. а. Граф G

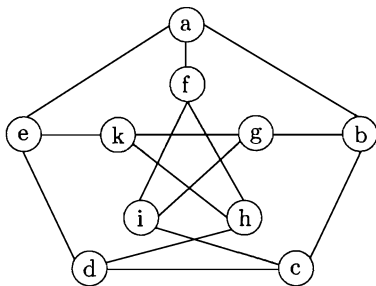


Рис. 6.73. б. Граф G'

Рассмотрим пример распознавания изоморфизма графов (рис. 6.73, *а*, *б*) G и G' . При этом $G = (X, U)$, $G' = (X', U')$, $|X| = |X'| = 10$, $|U| = |U'| = 15$, локальные степени всех вершин графа равны трем. Предположим, что вершина 1 графа G П-изоморфна вершине a графа G' . Тогда получим:

$$\{1\} \quad \{2, 5, 6\}_{1+} \quad \{3, 4, 7, 8, 9, 10\}_{1-} \quad (G)$$

$$\{a\} \quad \{b, e, f\}_{a+} \quad \{c, d, h, g, k, i\}_{a-} \quad (G')$$

Для дальнейшего анализа выбираем соответствующие подмножества наименьшей мощности $\{2, 5, 6\}$ и $\{b, e, f\}$. Предположим, что вершина 2 П-изоморфна вершине b . В этом случае получим:

$$\{1\} \quad \{2\}_{1+} \quad [\{5, 6\}_{1+}]_{2-} \quad [\{7, 3\}_{2+}, \{4, 8, 9, 10\}_{2-}]_{1-} \quad (G)$$

$$\{a\} \quad \{b\}_{a+} \quad [\{e, f\}_{a+}]_{b-} \quad [\{g, c\}_{b+}, \{d, h, k, i\}_{b-}]_{a-} \quad (G')$$

Как видно, П-изоморфизм сохраняется. Продолжая далее, запишем:

$$\{1\} \quad \{2\}_{1+} \quad [\{5, 6\}_{1+}]_{2-} \quad [[\{7\}_{3+}, \{3\}_{7+}]_{2+}, [\{9, 10\}_{7+}, \{4, 8\}_{7-}]_{2-}]_{1-} \quad (G)$$

$$\{a\} \quad \{b\}_{a+} \quad [\{e, f\}_{a+}]_{b-} \quad [[\{g\}_{c-}, \{c\}_{g-}]_{b+}, [\{i, k\}_{g+}, \{h, d\}_{g-}]_{b-}]_{a-} \quad (G')$$

Так как вершины (h, d) в графе G' (рис. 6.73, *б*) смежны, а вершины $(4, 8)$ в графе G (рис. 6.73, *а*) — нет, то вершина 1 не может быть изоморфна вершине a , вершина 2 — b , вершина 7 — g и вершина 3 — c . Продолжая аналогично, получим, что граф G не изоморфен графу G' .

Предлагается подход на основе микро-, макро-, и метаэволюции для нахождения подстановки изоморфизма графов, если она существует. На эта-

не микроэволюции эвристика разбиения позволяет получить строительные блоки. Перераспределение генетического материала на основе генетических операторов и их модификаций выполняется внутри каждого строительного блока. За счет этого уменьшается размер хромосом в популяции и сокращается время реализации основного алгоритма. Например, в хромосоме $p_1: 2, 3, 4, 5$, которая является строительным блоком для графа G (рис. 6.71, а), определим точку мутации между 3 и 4 геном. Выполнив ОМ, получим хромосому потомок вида $p_2: 2, 4, 3, 5$. Она определяет подстановку 2-*b*, 4-*c*, 3-*d* и 5-*e*.



Рис. 6.74. Структурная схема управления генетического поиска

На уровне макроэволюции можно эффективно использовать фрактальные множества, при этом каждый строительный блок представляется как объединенная вершина нового графа. В этом случае размер хромосом уменьшается и появляется возможность увеличить число генераций ГА для получения оптимального результата. Это особенно актуально при анализе однородных неизоморфных графов с одинаковым числом вершин и ребер.

На этапе метаэволюции происходит миграция хромосом из одной популяции в другую и различные модифицированные генетические операторы над популяциями.

Приведем структурную схему генетического поиска для решения переборных комбинаторно логических задач на графах (рис. 6.74) на основе информирующих обратных связей и концепции объединенной эволюции. После реализации ГА на рисунке 6.74 компенсатор при взаимодействии с внешней средой реализует синергетические принципы, а фильтр хромосом поддерживает гомеостаз. При этом лучшие хромосомы отправляются для смешивания популяций и выхода из локальных оптимумов. Редуктор уменьшает размер популяции, устраняя хромосомы со значением ЦФ ниже средней. Блоки сумматор, редуктор и фильтр хромосом позволяют повысить эффективность реализации эволюции и скорость распознавания изоморфизма графов. Следует отметить, что в графах большой размерности с нетривиальными автоморфизмами ($K > 100$) процесс установления изоморфизма резко усложняется, но использование таких схем поиска на порядок снижает временную сложность алгоритма.

Приведенные структурные схемы управления генетическим поиском позволяют повысить качество решения и уменьшить время поиска. Временная сложность алгоритма такого класса лежит в пределах $O(n^2) \div O(n^3)$.

Теория, не проверяемая опытом, при всей красоте концепции теряет вес, не признается: практика, не опирающаяся на взвешенную теорию, оказывается в проигрыше и убытке.

Д. Менделеев

7.1. Программная среда

Эффективность (оптимальность) любого алгоритма, по сравнению с аналогичными алгоритмами, может быть доказана в результате:

- теоретического исследования, т. е. сравнения оценок временной сложности алгоритмов и пространственной сложности алгоритмов;
- экспериментального исследования, т. е. путем проведения тестовых испытаний и сравнения экспериментальных данных использования различных алгоритмов.

При выполнении исследования реализуется серия экспериментов, собираются тестовые данные и проводится статистическое исследование этих данных.

Объектами, на которых проводятся исследования, являются такие оптимизационные задачи на графах, как разбиение; размещение вершин графа на плоскости и в линейке; задача о коммивояжере; раскраска графов; построение независимых подмножеств, клик графов и распознавания изоморфизма графов, рассмотренные выше.

Исследование алгоритмов направлено на то, чтобы обеспечить возможность получения ответов на вопросы прикладного характера, например: какое время потребуется для нахождения локального или глобального решения; какие графы алгоритм обрабатывает более эффективно.

Перед проведением экспериментов уточняется порядок их проведения, поскольку благодаря оптимальной организации планирования эксперимента появляется возможность с минимальными затратами материальных и временных ресурсов получить всю информацию, необходимую для построения адекватных математических моделей исследуемых объектов.

Для проведения объективных тестовых испытаний обычно разрабатывается генератор случайных графов. На вход генератора поступают управляемые параметры. Эти параметры принимают дискретные значения. Идеальным вариантом экспериментального исследования является проведение полного факторного эксперимента. Однако для оптимизационных задач на графах использование такого подхода не представляется возможным по причине больших временных затрат. Поэтому обычно проводят либо подробный факторный эксперимент, либо сокращенную серию тестовых ис-

пытаний, после чего применяют регрессионный анализ для определения зависимости реакции от факторов.

Зависимость $Y = F(X_1, X_2)$ может быть сведена к семейству зависимостей вида: $Y = F(X_1)$ при $X_2 = \text{const}$; $Y = F(X_2)$ при $X_1 = \text{const}$, т. е. фиксируем один из параметров и изменяем второй, в результате чего получаем семейство кривых, которые показывают зависимость выходного параметра от исследуемого. При обработке экспериментальных данных важной задачей является определение (идентификация) закона распределения, к которому можно отнести полученные данные. Если частота наблюдаемых событий близка к величине, предсказываемой теорией, то в дальнейшем можно строить модель исходных или ожидаемых событий на основе теоретического распределения, т. е. упрощается проблема дополнительной генерации необходимого количества данных.

Для оптимизационных задач на графах общей совокупностью являются всевозможные множества произвольных объектов. Наблюдаемые в последовательности n экспериментов значения образуют частичную совокупность значений случайной величины Y . Для обеспечения надежности экспериментальных выводов необходимо решить два основных вопроса:

- какое число единиц частичной совокупности достаточно велико для того, чтобы по нему можно было делать надежные выводы об изучаемом явлении;
- из каких именно единиц надо составить частичную совокупность, чтобы можно было рассматривать ее в качестве представительного образца изучаемой общей совокупности.

«Степень уверенности» или «мера риска» (доверительная вероятность) определяется величиной вероятности, с которой делается соответствующее заключение. Допустимая ошибка при исследованиях устанавливается в зависимости от природы изучаемого явления. В большинстве случаев допустимая ошибка принимается равной 0,05. Число наблюдений определяется по таблице больших чисел, составленной на основании теоремы Бернулли.

Среди свойств распределения одной случайной величины важными являются закон распределения случайной величины и два ее первых момента: математическое ожидание и дисперсия.

Описываемая далее программная система по исследованию характеристик оптимизационных алгоритмов выполнена в виде комплекса программных средств, состоящих из различных программных модулей, помимо которых она включает в себя так же:

- панель ввода/вывода, позволяющую загружать/сохранять тестовые примеры (графы с настройками), выводить на печать и т. п.;
- редактор графов;
- панели настроек алгоритма и ввода параметров;
- панель, позволяющую отображать в процессе работы значение ЦФ, график изменения ЦФ по популяции, абсолютно лучшее значение ЦФ, для визуальной оценки выполнения алгоритма.

Рассмотрим программные модули системы:

1. Программный модуль (ПМ₁) исследования различных модификаций генетических операторов — оператора рекомбинации, кроссинговера, мутации, инверсии, сегрегации, транслокации, их модификаций и совместного использования. Данный модуль построен на основе ПГА и используется для сравнения выходных характеристик одного и того же ГА для одинаковых графов при различных генетических операторах. При этом результатами являются: время работы алгоритма, стабильность алгоритма, лучшее решение, достигнутое в процессе работы, оценка алгоритма по сходимости. Общая структурная схема данного программного модуля приведена на рисунке 7.1.

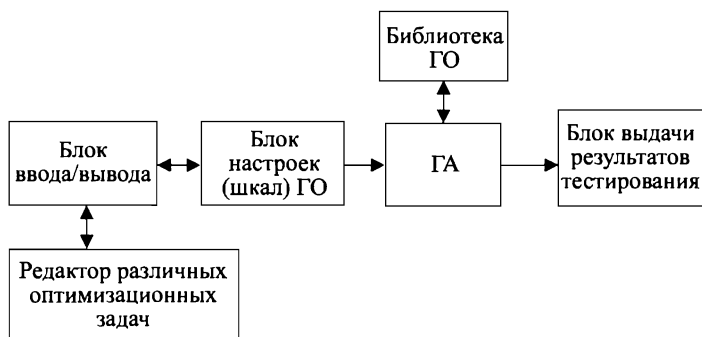


Рис. 7.1. Тестирование генетических операторов

В эту схему входят следующие блоки:

- блок ввода данных задачи (ввод переменных и параметров исследуемых моделей, матриц соединений, критериев и т. п.);
- блок ввода данных настроек алгоритма (размер популяции, типы применяемых операторов, вероятности их использования и т. п.);
- ГА с учетом настроек алгоритма;
- блок редактирования различных оптимизационных задач;
- блок выдачи результатов.

2. Программный модуль (ПМ₂) исследования применяемых в алгоритме эвристических методов поиска, блока итерационного и статистического улучшения. Данная схема выполнена по аналогии с ПМ₁ и состоит из (рис. 7.2):

- блока ввода/вывода;
- блока ввода параметров настроек алгоритма (включаемые функции оптимизации поиска);
- ГА с учетом настроек оптимизации поиска;
- блока выдачи результатов (в качестве выходных данных используются те же характеристики, что и в ПМ₁);
- редактор различных оптимизационных задач;
- панель настроек, позволяющую отключать (или подключать) различные методики улучшения процесса поиска;

- панель работы алгоритма, в которой выводится информация о текущем значении лучшей ЦФ, и график изменения значения ЦФ популяции, а также график изменения решения по всей генерации алгоритма.

Кроме того, в процессе работы создается файл отчета, содержащий в себе код всех хромосом всех итераций на текущей генерации, лучшее значение по каждой популяции и результат глобального экстремума.

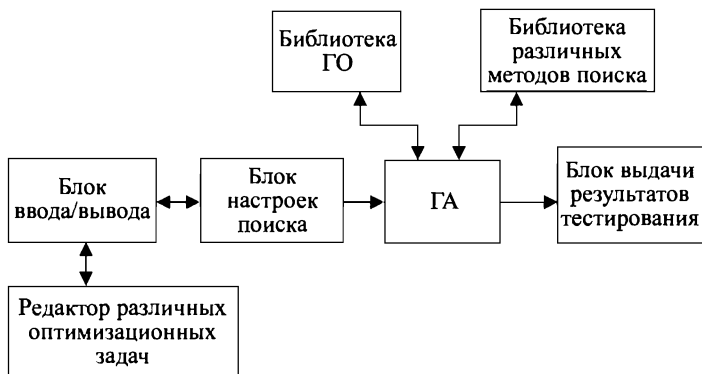


Рис. 7.2. Тестирование методов поиска

3. Программный модуль (ПМ₃) исследования применяемых моделей эволюций, эвристических методов поиска, блока итерационного и статистического улучшения. Данная схема выполнена по аналогии с ПМ₁ и имеет в своем составе (рис. 7.3):

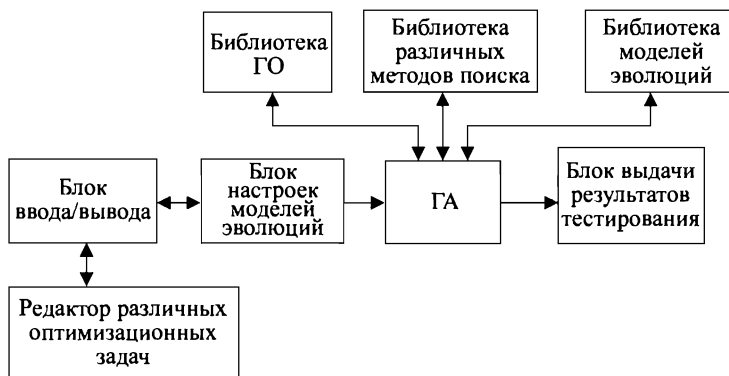


Рис. 7.3. Тестирование эволюций

- блок ввода данных задачи;
- блок ввода параметров настроек алгоритма (включаемые функции оптимизации поиска);
- библиотеку различных моделей эволюций;
- ГА с учетом настроек оптимизации поиска;

- блок выдачи результатов (в качестве выходных данных используются те же характеристики, что и в ПМ₂);
- блок ввода/вывода;
- редактор различных оптимизационных задач;
- панель настроек, позволяющую отключать (или подключать) различные методики улучшения процесса поиска и различные модели эволюций;
- панель работы алгоритма, в которой выводится информация о текущем значении лучшей ЦФ и график зависимости максимального и минимального значения ЦФ популяции, а также график изменения решения по всей генерации алгоритма.

4. Программный модуль (ПМ₄) исследования применяемых в алгоритме методик поиска, блока итерационного и статистического улучшения, блока адаптации. Эта схема выполнена по аналогии с ПМ₃ и имеет в своем составе (рис. 7.4):

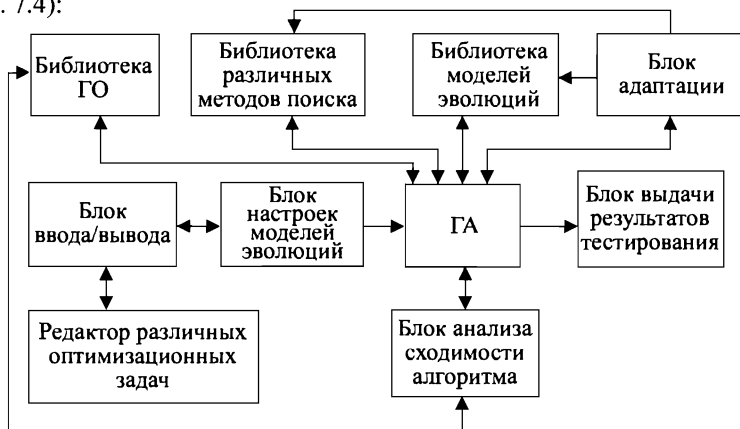


Рис. 7.4. Тестирование блока адаптации

- блок ввода данных задачи;
- блок ввода параметров настроек алгоритма (включаемые функции оптимизации поиска);
- ГА с учетом настроек оптимизации поиска;
- библиотеку различных моделей эволюций;
- блок адаптации;
- блок настроек параметров;
- блок анализа сходимости алгоритмов;
- блок выдачи результатов (в качестве выходных данных используются те же характеристики, что и в ПМ₃).
- блок ввода/вывода;
- редактор различных оптимизационных задач;
- панель настроек, позволяющую отключать (или подключать) различные методики улучшения процесса поиска, разнообразные модели эволюций, а также блок адаптации.

В процессе работы создается файл отчета, содержащий в себе код всех хромосом всех итераций на текущей генерации, лучшее значение по каждой популяции и результат глобального экстремума.

Алгоритмы выполнения исследовательского программного модуля по тестированию элементов структуры ГА показаны на рисунках 7.5–7.7. Выполнение программного модуля данной подсистемы начинается с ввода сведений о задачах. Далее выполняется настройка общих и частных параметров алгоритма, к которым относятся:

- начальное распределение коэффициентов и критериев;
- начальный размер популяции;
- ограничение на число итераций алгоритма (число шагов);
- ограничение на число генераций (количество запусков алгоритма);
- ограничение по значению ЦФ (если не задано — поиск глобального оптимума);
- вероятности использования генетических операторов.

Приведем частные параметры для программного модуля (рис. 7.5): тип операторов кроссинговера, мутации, инверсии, сегрегации, транслокации и селекции. После выполнения всех известных генетических операторов выполняется проверка ЦФ. При неудовлетворительном значении ЦФ процесс исследования начинается сначала.

Частными параметрами для программного модуля (рис. 7.6) являются использование эвристик на основе: статистических методов оптимизации; градиентных методов; методов дихотомии; методов Фибоначчи; методов золотого сечения; фрактальных множеств и др. После выполнения всех известных и поисковых методов выполняется ГА. Если критерий «останова» достигнут, то — выход; если нет, то процесс исследования начинается сначала.

Частные параметры для программного модуля (рис. 7.7) следующие: использование моделей эволюций Дарвина, Ламарка, де Фриза, Поппера, блока адаптации, блока анализа и преодоления сходности.

После реализации моделей эволюций выполняется ГА. Блок адаптации отвечает за обратные связи и установления баланса. Далее выполняется непосредственно ГА в соответствии с используемой схемой с установленными генетическими операторами, которые влияют в итоге на качественные характеристики поиска. Вычислительная сложность данного алгоритма равна $O(n)$, при этом выполняется только генетический поиск.

ГА реализуется с лучшим постоянным набором операторов, определенных при тестировании. В процессе работы алгоритма применяются разработанные методики настройки поиска. Временная сложность данного алгоритма в общем случае при всех включенных эвристиках равна $O(n^3)$. Основная цель эксперимента — выявление общего улучшения, вносимого эвристиками в процесс поиска, а также их поведения при различных типах графов.

Затем выполняется ГА с установленным по результатам проведенного тестирования набором операторов, с выполнением всех эвристических

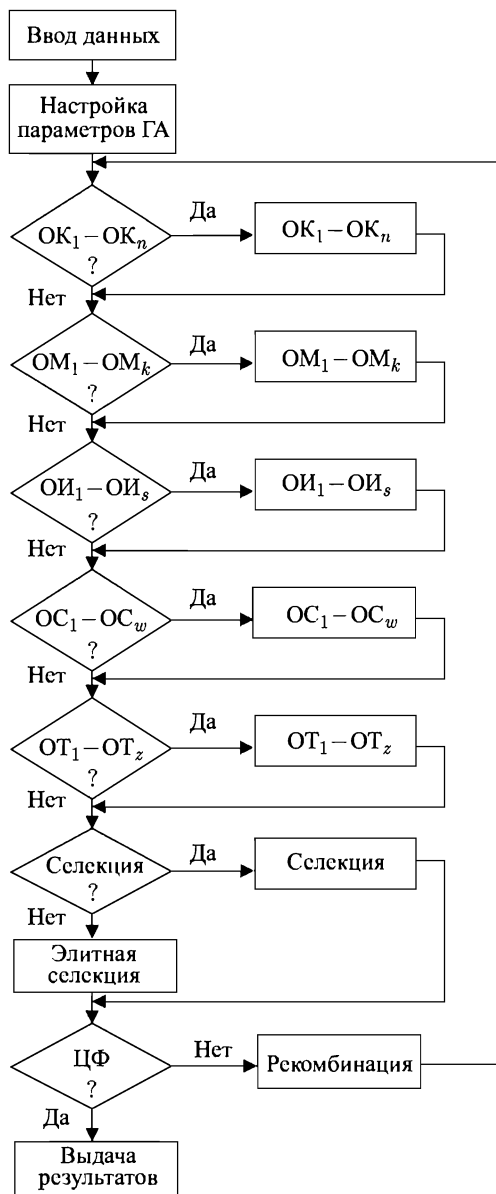


Рис. 7.5. Структурная схема алгоритма исследования генетических операторов

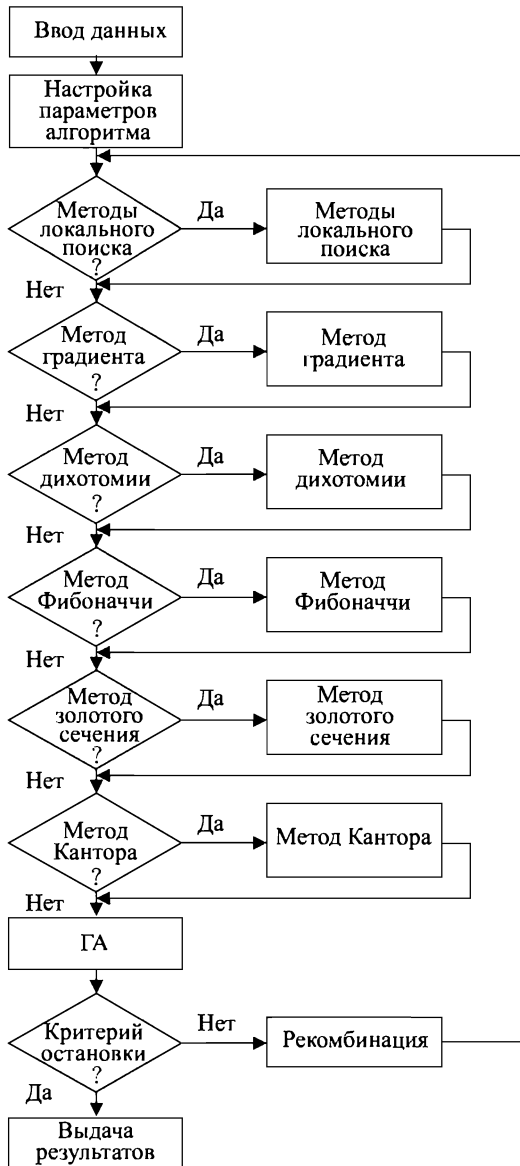


Рис. 7.6. Структурная схема исследования поисковых методов

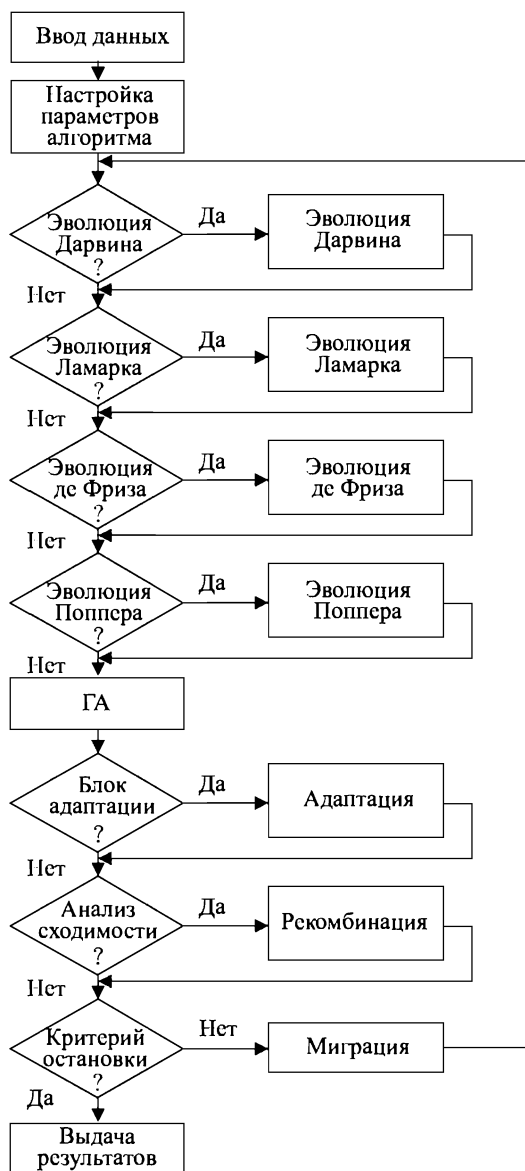


Рис. 7.7. Алгоритм исследования моделей эволюций с адаптацией и миграцией

процедур и моделей эволюций. Согласно настройкам в процессе работы используются блоки адаптации, анализа и преодоления сходимости алгоритма. Временная сложность алгоритма колеблется от $O(n)$ до $O(n^4)$. Основная цель эксперимента — выявление лучших характеристик по времени работы комплекса алгоритмов, а также повышения стабильности алгоритмов. Для определения показателей улучшения разработанных методик, алгоритмов и программных модулей проведен ряд экспериментов, состоящий из трех основных серий — тестирование методик, тестирование операторов и ГА, тестирование моделей эволюций.

Сущность данных экспериментов заключается в тестировании:

а) алгоритмов на одной конкретной задаче с последовательным включением всех разработанных эвристик на одной стандартной задаче и определением значений улучшаемых параметров. Это позволяет оценить эффективность используемых в алгоритме методик оптимизации поиска.

б) алгоритмов на некоторых стандартных тестах с последующим сравнением полученных результатов с уже имеющимися для данной задачи (бенчмарками), просчитанными на каком-либо другом алгоритме. Это позволяет оценить практическую эффективность алгоритма и программных модулей.

Опишем теперь результаты экспериментальных исследований и приведем сравнительные количественные и качественные характеристики для описанных оптимизационных задач на графах.

Для адекватного сравнения использовались стандартные оценки по производительности различных систем (бенчмаркам), представленных ведущими фирмами мира.

7.2. Разбиение графа на части по критерию числа внешних ребер

Приведем интерфейс программы разбиения. При запуске программы на экране появляется главное окно программного комплекса. Его внешний вид представлен на рис. 7.8.

Это окно делится на две части. В правом верхнем углу расположены поля ввода подграфов разбиения. Сумма введенных значений должна соответствовать общему количеству вершин графа. Ниже расположена область окна, в которой показан график изменения ЦФ от количества итераций алгоритма. Под ней расположены окна вывода значений ЦФ для разных типов алгоритмов (последовательный, итерационный, ПГА, ГА с подсистемой самоорганизации), а также окно для ввода числа итераций алгоритма. Под ГА с подсистемой самоорганизации понимают алгоритмы, использующие модели совместных эволюций, принцип синергетики, блоки адаптации, экспертные системы и т. п. (например, рис. 6.1, 6.13, 6.17).

В верхней части главного окна расположено меню, состоящее из следующих пунктов: «файл»; «алгоритмы»; «параметры»; «операции»; «справка».

Пункт главного меню «файл» содержит стандартный набор команд для работы с введенными данными (создание нового, сохранение, открытие и

). Пункт меню «алгоритмы» позволяет выбрать алгоритм, используемый при разбиении графа. Он содержит четыре блока для разных типов алгоритмов. Пункт меню «параметры» состоит из двух подпунктов: вероятности операторов и подсистема самоорганизации (рис. 7.9).

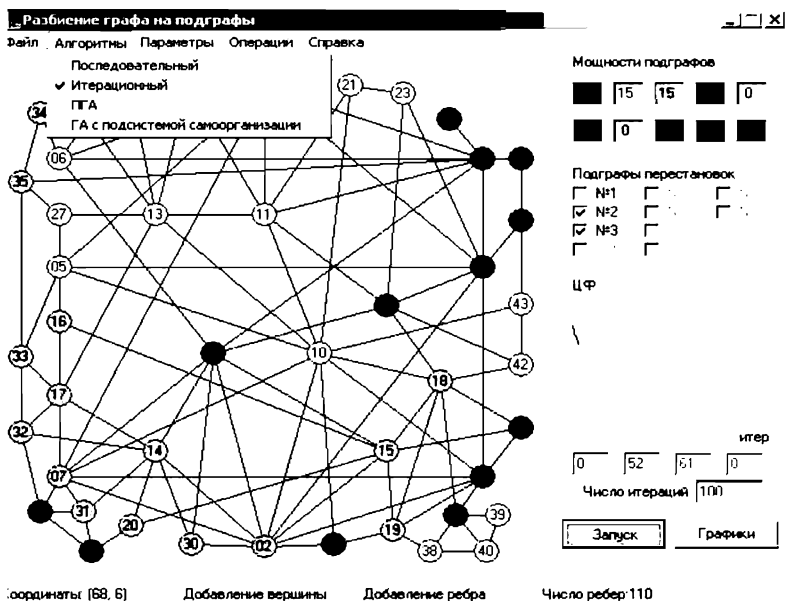


Рис. 7.8. Главное окно

Подпункт «вероятности генетических операторов» (рис. 7.10) позволяет задавать шкалы (вероятности) операторов в процентах. Подпункт «подсистема самоорганизации» (рис. 7.11) позволяет подключать блок самоорганизации для выполнения различных операторов на основе методов дихотомии, золотого сечения, Фибоначчи и др., а также строить порядок из хаоса с повторным запуском.

Меню «операции» содержит команды для построения графов: добавить/удалить вершину; добавить/удалить ребро; переместить вершину. В нижней части окна комплекса расположена панель инструментов. Содержит: индикатор состояния выполнения алгоритма; кнопку запуска алгоритма разбиения; кнопку вывода графиков. После выполнения алгоритма разбиения графа окрашиваются в цвет, соответствующий подграфу, которому они принадлежат. На рис. 7.12 показано лучшее разбиение графа, найденное на основе описанного алгоритма (ЦФ = 39 за 900 итераций).

На рис. 7.13 приведено окно с графиками ПГА и ГА с подсистемой самоорганизации.

Проанализируем влияние размера популяции на получаемые результаты. При этом будем варьировать вероятность всех операторов, а также количество итераций ГА. Построим следующую таблицу (табл. 7.1).

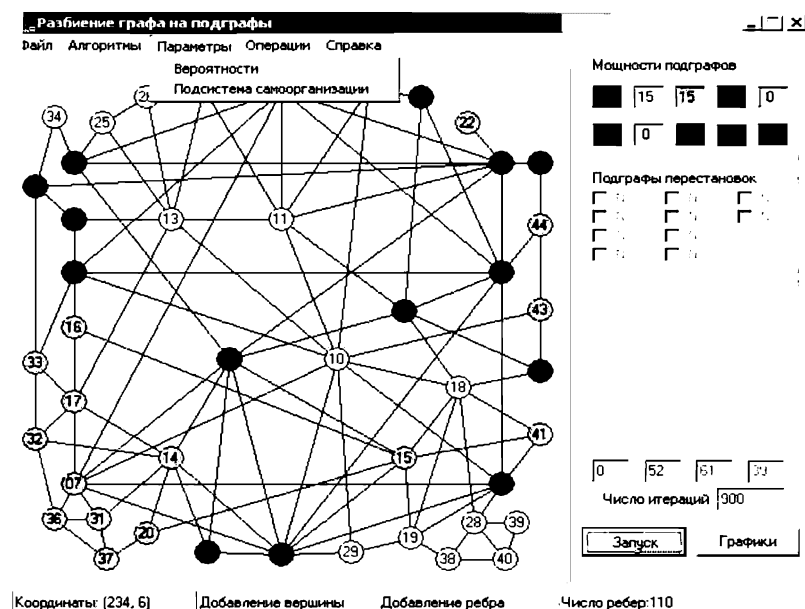


Рис. 7.9. Вид подменю параметры

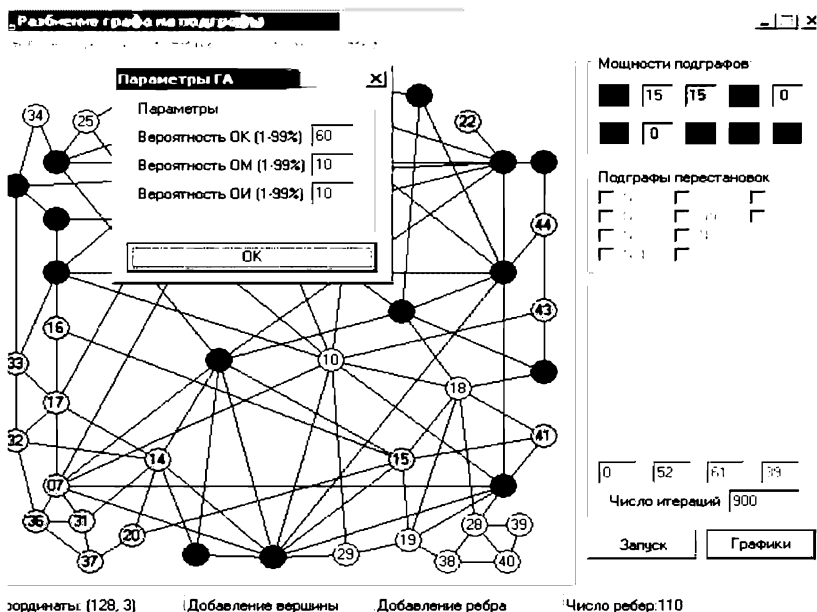


Рис. 7.10. Подпункты вероятности применения операторов

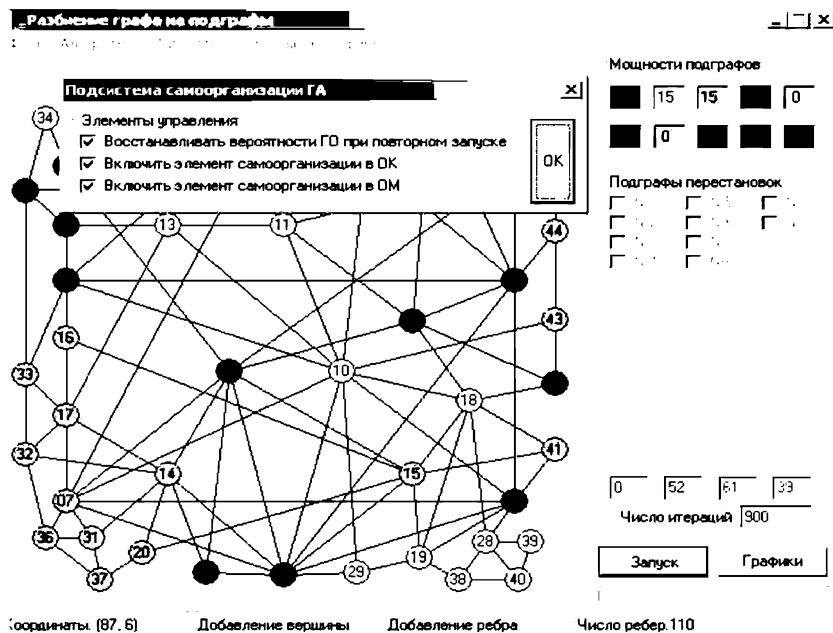


Рис. 7.11. Подпункты подсистемы самоорганизации

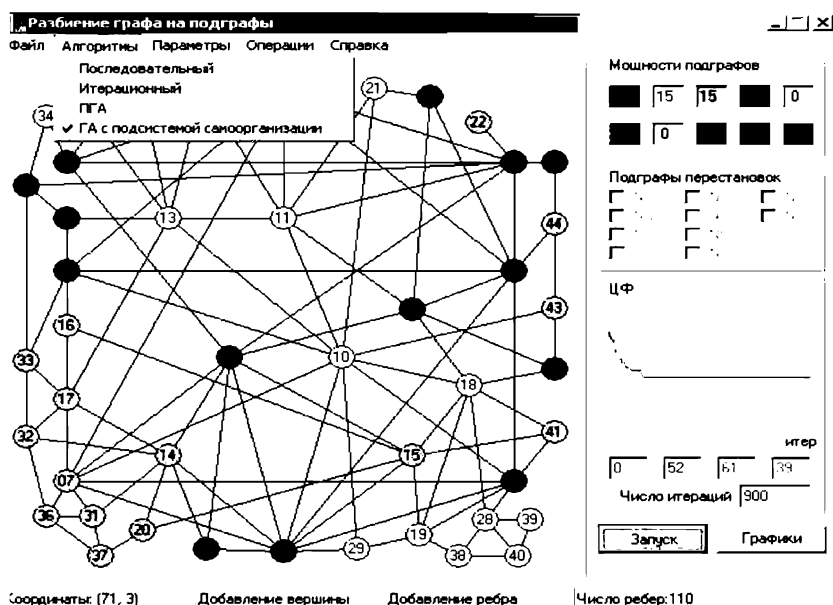


Рис. 7.12. Лучшее разбиение графа

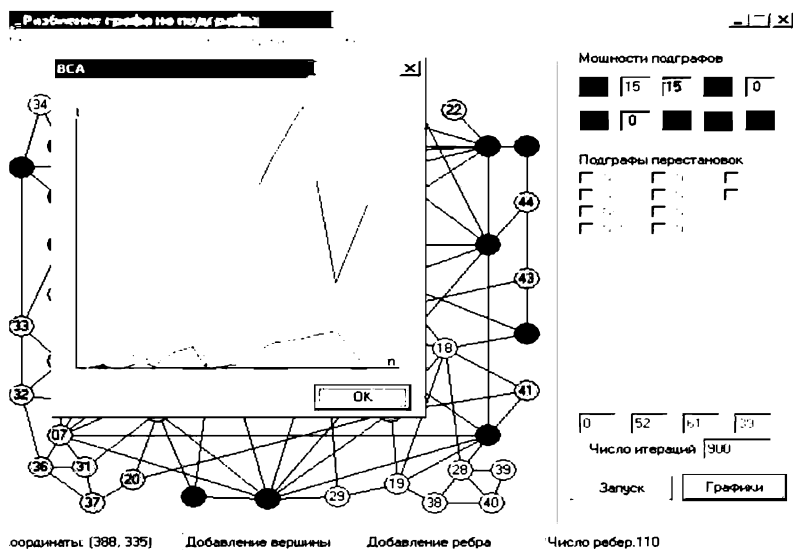


Рис. 7.13. Интерфейс графиков

оторой приведем результаты проведенных экспериментов. В таблице приведены различные данные разбиения графа, содержащего $n = 450$ и $m = 1000$ ребер, на три части по 150 вершин в каждой с минимальным количеством внешних ребер. В первом столбце таблицы указываем номера проведенных экспериментов, во втором — количество вершин, что в таблице приведены данные по исследованию разбиения о нестандартного трудного теста. Здесь изменялись размер популяции, количество итераций ГА, шкалы ОК, ОМ, оператора инверсии и процедура эволюции. В третьем столбце указан размер популяции, в четвертом — количество итераций, в пятом, шестом и седьмом — вероятности применения операторов кроссинговера, мутации и инверсии. В восьмом столбце записан тип проведенной селекции (С — случайная, Э — элитная). В девятом столбце приведены данные для ПГА (время, затраченное на получение лучшего решения и значение ЦФ, т. е. количества внешних ребер для заданного разбиения), а в десятом указаны аналогичные данные для приведенного алгоритма. Для исследования стандартного графа были реализованы так же эволюционные методы разбиения, как последовательные, итерационные, и эволюция в глубину. При этом значение ЦФ в первом случае составило 500 основных единиц, во втором — 500, а в третьем 510.

По результатам экспериментов, сведенных в таблице, построим график зависимости времени (t) получения лучшего решения от количества итераций N_G (рис. 7.14). Из графика следует, что для ПГА время решения практически линейно зависит от числа генераций ГА. Размер популяции, видно из таблицы, влияет на время решения и на получение оптимальных результатов.

Таблица 7.1

1	2	3	4	5	6	7	8	9		10	
№	N	N_p	N_G	$P(OK)$	$P(OM)$	$P(OИ)$	S	ПГА		ГА с само- организацией	
								t	ЦФ	t	ЦФ
1	450	10	50	0,5	0,5	0,5	С	20	600	22	540
2	450	20	100	0,6	0,4	0,4	Э	25	580	27	500
3	450	30	200	0,7	0,3	0,3	Э	40	560	43	480
4	450	40	300	0,8	0,2	0,2	Э	50	540	55	470
5	450	50	400	0,9	0,1	0,1	Э	60	520	60	450
6	450	60	500	0,9	0,1	0,1	Э	65	500	70	430
7	450	70	600	0,9	0,1	0,1	Э	70	500	80	420
8	450	70	700	0,9	0,1	0,1	Э	75	500	85	410
9	450	70	800	0,8	0,2	0,2	Э	90	500	100	400
10	450	70	900	0,8	0,2	0,2	Э	100	500	110	390
11	450	80	1000	0,8	0,2	0,2	С	120	500	130	380
12	450	100	1200	0,9	0,1	0,1	Э	150	500	170	350

Отметим, что с попаданием в локальный оптимум увеличение размера популяции, увеличение количества генераций, изменение шкал генетических операторов в большинстве проанализированных примеров не изменяют величины оптимизированной функции. Использование блока адаптации и комбинированных моделей эволюций, различных типов поиска при незначительном увеличении времени работы алгоритма позволяет решать проблему предварительной сходимости алгоритма и получать оптимальные результаты, что следует из таблицы 7.1.

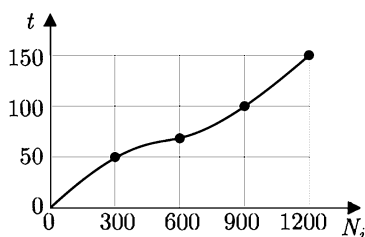


Рис. 7.14 График зависимости времени решения от числа итераций

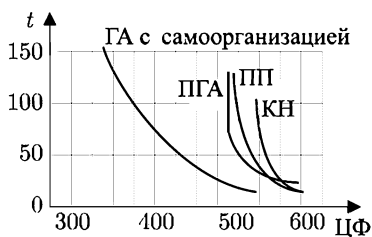


Рис. 7.15 Графики зависимости времени получения лучших решений

На рис. 7.15 приведены графики зависимости времени решения от значения ЦФ для итерационного (ПП), последовательного (КН) алгоритмов, ПГА и ГА с подсистемой самоорганизации.

Из графика и таблицы следует, что описанные ГА дают лучшие решения по качеству при одинаковом или незначительном увеличении времени

работы. До попадания в локальный оптимум управление параметрами генетического поиска позволяет получать лучшие решения. На рис. 7.16 приведен график зависимости времени получения лучшего решения от размера популяции для ПГА.

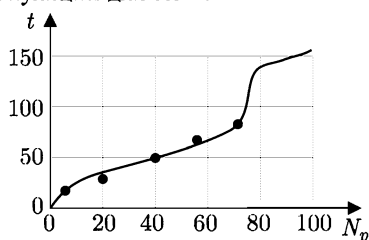


Рис. 7.16 График зависимости времени решения от размера популяции

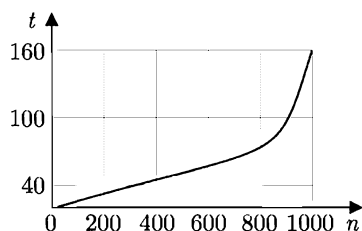


Рис. 7.17 График зависимости времени решения от числа элементов

В таблице 7.2 приведены результаты экспериментов с тем же алгоритмом разбиения, но проанализировано разбиение не одного графа, а набора тестовых задач.

Таблица 7.2

1	2	3	4	5	6	7	8	9	10	11
№	n	m	N_p	N_G	P(МОК)	P(МОМ)	P(МОС)	S	t	ЦФ
1	50	150	50	1000	0,8	0,3	0,1	Э	6	0,54
2	100	300	50	1000	0,8	0,3	0,1	Э	20	0,50
3	200	600	50	1000	0,8	0,3	0,1	Э	30	0,48
4	300	900	50	1000	0,8	0,3	0,1	Э	40	0,41
5	400	1200	50	1000	0,8	0,3	0,1	Э	46	0,45
6	500	1500	40	500	0,8	0,3	0,1	Э	57	0,47
7	600	1800	30	500	0,8	0,3	0,1	Э	60	0,50
8	800	2400	20	500	0,8	0,3	0,1	Э	80	0,52
9	1000	3000	10	500	0,8	0,3	0,1	Э	160	0,54

Исследовано поведение алгоритма разбиения при использовании различных модифицированных операторов. В шестом, седьмом и восьмом столбцах таблицы указаны величины шкал модифицированных операторов кроссинговера, мутации и сегрегации (P(МОК), P(МОМ), P(МОС)). Значение условной ЦФ (см. 11 столбец таблицы) определяется отношением количества внешних ребер к количеству внутренних ребер при разбиении. Необходимо стремиться к такому разбиению, чтобы $ЦФ \Rightarrow 0$. Селекция выбрана элитная. Размер популяции зафиксирован постоянный. Но в процессе решения после каждой генерации размер популяции менялся в основном в сторону уменьшения, оставляя лучшие с точки зрения ЦФ хромосомы. На рис. 7.17 показан график зависимости времени получения лучшего решения от числа вершин исследуемого графа.

Эксперименты показали, что при разбиении графов на части использование блока адаптации, нестандартных методов поиска, модифицированных операторов, совместных моделей эволюций (Дарвина, Ламарка, де Фриза и Поппера) при моделировании эволюции позволяет получать набор оптимальных решений. При этом с большой вероятностью среди этих решений может быть найден глобальный экстремум.

Из приведенных статистических данных следует, что в общем случае время решения линейно зависит от количества генераций и временная сложность описанных алгоритмов подтверждает теоретические предпосылки и приближенно равна $O(n \log n) \div O(n^3)$. Следует заметить, что с увеличением числа итераций в ГА время получения результата разбиения повышается, но это повышение незначительное и компенсируется получением множества локально-оптимальных решений. Анализ графиков и приведенных таблиц позволяет отметить, что описанные ГА требуют больших затрат времени, чем ПГА. Они позволяют получать набор локально-оптимальных решений, решать в общем случае проблему предварительной сходимости. Для каждого конкретного теста (как следует из таблиц) условная ЦФ, в описанных алгоритмах, принимает лучшие значения, причем эти алгоритмы имеют большее быстроедействие, чем метод ветвей и границ, метод моделирования отжига и другие аналогичные методы.

7.3. Исследование генетического алгоритма размещения

Оптимальное решение задачи размещения, являющейся NP-полной задачей, можно найти только в результате полного перебора всех решений и определении наилучшего. Опишем результаты испытаний алгоритмов размещения и их количественные и качественные оценки. Построим таблицу 7.3, в которой приведем результаты тестовых испытаний алгоритма размещения вершин графов в одной линейке.

Здесь столбцы 5–9 — это вероятности реализации операторов. В таблице столбец 11 — это условная начальная длина ребер графа, а столбец 12 —

конечная длина ребер графа. В столбце 13 приведено значение изменения суммарной длины ребер в процентах (Δ). На рис. 7.18 приведен график зависимости суммарной длины ребер графа от времени.

Из результатов исследования следует, что увеличение размера популяции и количества итераций алгоритма приводит к уменьшению суммарной длины, но только до некоторого предела.

Для графа, описанного в таблице, были проведены испытания на основе классических последовательных, итерационных и релаксационных методов. При этом во всех случаях условная ЦФ была в среднем хуже чем в ГА с самоорганизацией на 40%, 30%, 20%

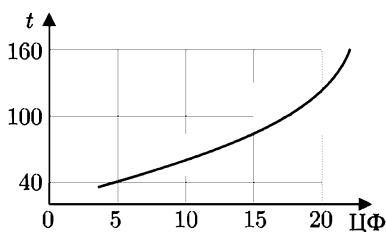


Рис. 7.18. График изменения времени решения от суммарной длины ребер графа

Таблица 7.3

1	2	3	4	5	6	7	8	9	10	11	12	13
№	n	m	N_P	N_G	$P(OM)$	$P(OИ)$	$P(OC)$	$P(OK)$	t	$L(G_H)$	$L(G_K)$	$\Delta\%$
1	600	2400	10	100	0,1	0,1	0,1	0,6	40	3600	3470	3,61
2	600	2400	20	200	0,6	0,1	0,1	0,7	50	3590	3400	5,29
3	600	2400	30	300	0,9	0,2	0,1	0,8	60	3575	3330	6,85
4	600	2400	40	400	0,1	0,2	0,2	0,8	70	3560	3215	9,7
5	600	2400	50	500	0,2	0,3	0,2	0,8	90	3550	2925	17,6
6	600	2400	60	600	0,2	0,3	0,3	0,9	110	3535	2865	18,9
7	600	2400	70	700	0,3	0,4	0,3	0,9	120	3520	2830	19,6
8	600	2400	80	800	0,3	0,4	0,4	0,9	140	3515	2820	19,8
9	600	2400	90	900	0,4	0,5	0,4	0,9	160	3500	2740	21,7
10	600	2400	100	1000	0,5	0,6	0,5	0,9	160	3500	2740	21,7

ответственно. Алгоритм позволяет в общем случае решать проблему сходимости и получать оптимальные результаты. В результате тестовых испытаний отметим следующее. Все теоретические предположения относительно преимуществ ГА с самоорганизацией подтвердилось. Тогда ГА для размещения вершин графов позволяет всегда получать хорошие оптимумы, иметь возможность выхода из них и приближаться к получению глобального оптимума, причем, что особенно важно, время работы алгоритма не уходит из области полиномиальной сложности, а приблизительно равна $O(n \log n) \div O(n^3)$.

7.4. Исследование задачи о коммивояжере

Алгоритм был исследован на целом ряде тестовых задач о коммивояжере с наилучшими результатами, приведенных в литературе — Oliver's-50 и Eilon's-75 [182, 183]. Приведем интерфейс программы решения задачи о коммивояжере (рис. 7.19).

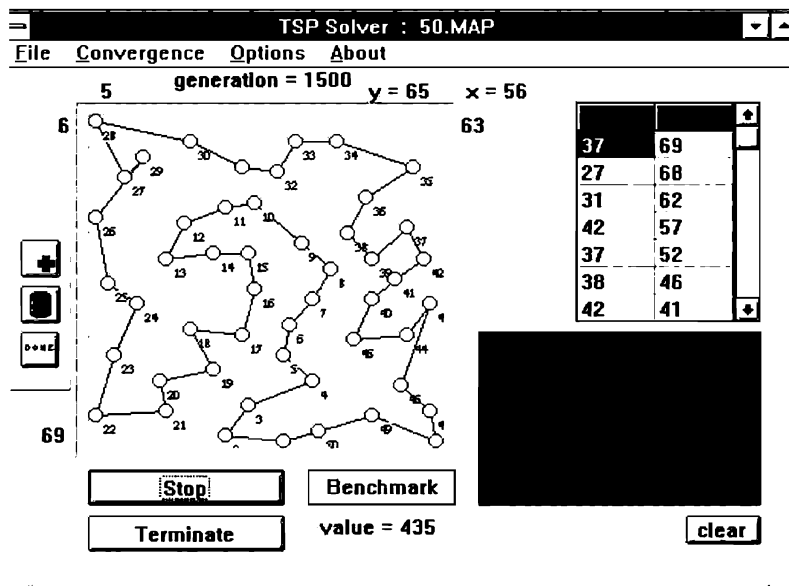


Рис. 7.19. Интерфейс разработанной программы

Окно (рис. 7.19) разделено на три основных части. В левой показана конфигурация маршрута коммивояжера. В правой верхней части координаты маршрута по осям x и y . Здесь (5, 6) — координаты вершины анализируемого графа по осям x и y соответственно. Величина 63 обозначает предельную координату по оси x для размещения графа, а 69 — по оси y соответственно. В правой нижней части окна приводится график изменения ЦФ. В задаче с 30 городами описанный алгоритм на основе эволюционных и жадных стратегий совместно с генетическим поиском по-

нескольких генераций всегда находит оптимальное решение. Оптимальное решение в этой задаче находится в среднем за 2000 итераций.

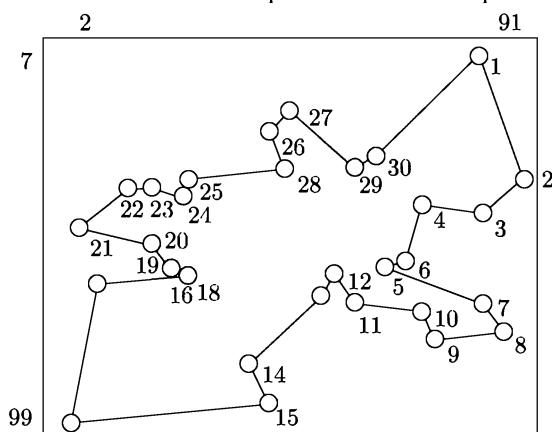


Рис. 7.20. Наилучший полученный маршрут в Oliver-30 (Длина = 419, размер популяции = 20, уровень мутаций = 0,3, элитное число = 10)

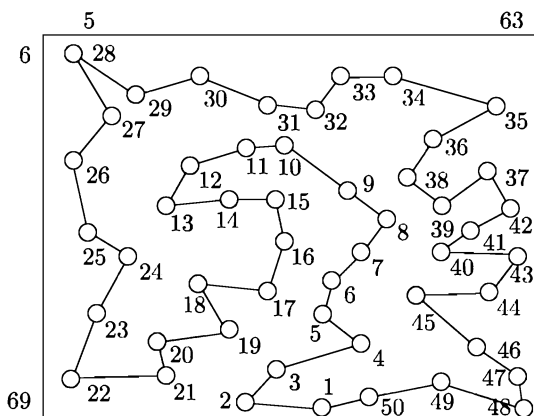


Рис. 7.21. Наилучший полученный маршрут в Eilon-50 (Длина = 425, размер популяции = 20, уровень мутаций = 0,3, элитное число = 10)

Конфигурации маршрута решения для 30 городов показана на рисунке 7.20. Приведем список городов, заданных координатами, в порядке проведения пути (здесь $a(b, c)$ означает, что a — вершина, b и c — координаты):

10(71, 71), 9(74, 78), 8(87, 76), 7(83, 69), 6(64, 60), 5(68, 58),
 4(71, 44), 3(83, 46), 2(91, 38), 1(82, 7), 30(62, 32), 29(58, 35),
 27(45, 21), 26(41, 26), 28(44, 35), 25(25, 38), 24(24, 42), 23(18, 40),
 22(13, 40), 21(4, 50), 20(18, 54), 18(22, 60), 19(25, 62), 17(7, 64),
 16(2, 99), 15(41, 94), 14(37, 84), 13(51, 67), 12(54, 62), 11(58, 69).

Известно, что наилучший путь коммивояжера составлял 423 условных единицы. Наилучший результат, полученный после испытаний алгоритма, составил 419.

Лучший путь для графа на 50 вершин (Eilon's-50) составлял 428 условных единиц, а для графа на 75 вершин (Eilon's-75) — 545. Описанный алгоритм определил конфигурации наилучших маршрутов. Они показаны на рис. 7.21 и рис. 7.22, соответственно.

Новое лучшее решение графа на 50 вершин (Eilon's-50) равно 425:

10(32, 22), 11(27, 23), 12(20, 26), 13(17, 33), 14(25, 32), 15(31, 32),
 16(32, 39), 17(30, 48), 18(21, 47), 19(25, 55), 20(16, 57), 21(17, 63),
 22(5, 64), 23(8, 52), 24(12, 42), 25(7, 38), 26(5, 25), 27(10, 17),
 28(5, 6), 29(13, 13), 30(21, 10), 31(30, 15), 32(36, 16), 33(39, 10),
 34(46, 10), 35(59, 15), 36(51, 21), 38(48, 28), 39(52, 33), 37(58, 27),
 42(61, 33), 41(56, 37), 40(52, 41), 43(62, 42), 44(58, 48), 45(49, 49),
 46(57, 58), 47(62, 63), 48(63, 69), 49(52, 64), 50(43, 67), 1(37, 69),
 2(27, 68), 3(31, 62), 4(42, 57), 5(37, 52), 6(38, 46), 7(42, 41),
 8(45, 35), 9(40, 30).

Новое лучшее решение графа на 75 вершин (Eilon's-75) равно 535:

29(30, 20), 28(27, 24), 27(22, 22), 26(26, 29), 25(20, 30), 24(21, 36),
 47(21, 45), 23(21, 48), 22(22, 53), 21(26, 59), 45(30, 60), 46(35, 60),
 49(40, 60), 19(35, 51), 20(30, 50), 18(33, 44), 17(29, 39), 16(33, 34),
 15(38, 33), 14(40, 37), 13(45, 35), 12(45, 42), 11(41, 46), 10(50, 50),
 9(55, 50), 57(55, 57), 56(62, 57), 55(70, 64), 54(57, 72), 53(55, 65),
 52(50, 70), 48(47, 66), 50(40, 66), 51(31, 76), 43(10, 70), 44(17, 64),
 42(15, 56), 41(9, 56), 40(7, 43), 39(12, 38), 38(11, 28), 37(6, 25),
 36(12, 17), 35(16, 19), 34(15, 14), 33(15, 5), 32(26, 13), 31(36, 6),
 72(44, 13), 71(50, 15), 70(54, 10), 69(50, 4), 68(59, 5), 67(64, 4),
 66(66, 8), 65(66, 14), 64(60, 15), 63(55, 20), 62(62, 24), 61(65, 27),
 60(62, 35), 59(67, 41), 58(62, 48), 8(55, 45), 7(51, 42), 6(50, 40),
 5(54, 38), 4(55, 34), 3(50, 30), 2(52, 26), 1(48, 21), 75(43, 26),
 74(36, 26), 73(40, 20), 30(35, 16).

Показатели по тестам являются лучшими в сравнении со всеми имеющимися в научной литературе (Oliver's-30 — 419, Eilon's-50 — 425, Eilon's-75 — 535) по решению задачи коммивояжера.

Была протестирована серия из 100 графов на 200, 400, 600, 800 и 1000 вершин. При этом временная сложность алгоритма не выходила из области полиномиальной сложности. В лучшем случае временная сложность алгоритма $\approx O(n \log n)$, в худшем случае — $\approx O(n^3)$. Отметим, что отличительной особенностью ГА является способность хорошо работать на

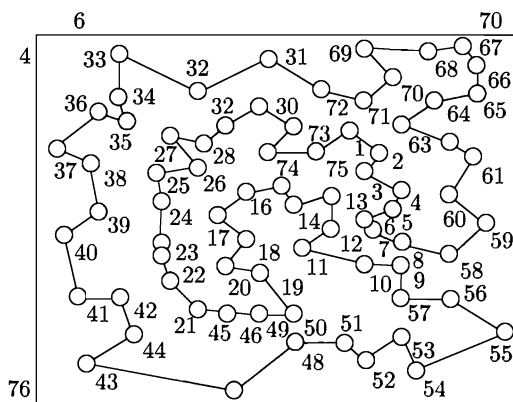


Рис. 7.22. Наилучший полученный маршрут в Eilon's-75 (Длина = 535, размер популяции = 20, уровень мутаций = 0,3, элитное число = 10)

популяциях с малым числом хромосом, что уменьшает время реализации алгоритма. Подобное улучшение связано с использованием различных комбинаций моделей эволюций, блоком адаптации, различными обратными связями, локальным поиском, новыми и модифицированными операторами, а также обменом информацией с внешней средой. Полученные результаты в генетическом решении позволяют говорить об эффективности метода и целесообразности использования в решениях других задач оптимизации.

7.5. Задачи построения независимых множеств, раскраски графа

Задача построения независимых (внутренне устойчивых) подмножеств множества вершин графа неформально может быть сформулирована следующим образом. В заданном графе определяется семейство подмножеств вершин. Внутри каждого подмножества вершины графа между собой не соединены ребрами. Добавление любой вершины графа к такому подмножеству делает его не внутренне устойчивым. Задачи такого рода актуальны, когда необходимо определить наибольшее число групп независимых объектов. Например, в отряде космонавтов определить наибольшее число групп совместимых людей для работы в космосе.

Для экспериментального исследования описанных алгоритмов построения независимых подмножеств были проведены тестовые испытания графов. На их основе построена таблица 7.4, в которую внесены результаты испытания алгоритмов определения независимых подмножеств. Количество вершин графа $n = 100$, количество ребер $m = 400$. В таблице в столбце 2 указан размер популяции, в 3 — число итераций, в 4, 5 — вероятности ОК, ОМ, в 6, 7 — приведены значения условной ЦФ как отношение количества определенных и существующих независимых множеств и время решения в (сек) для ПГА. В 8, 9 — значения условной ЦФ и время для ГА с самоорганизацией.

Таблица 7.4

1	2	3	4	5	6	7	8	9
№	N_p	N_G	P(OK)	P(OM)	ПГА		ГА с самоорганизацией	
					ЦФ	t	ЦФ	t
1	10	100	0,6	0,4	0,70	9	0,76	12
2	20	200	0,7	0,3	0,71	15	0,77	20
3	30	300	0,8	0,2	0,72	25	0,78	30
4	40	400	0,8	0,2	0,73	30	0,79	40
5	50	500	0,9	0,1	0,74	50	0,80	55
6	60	600	0,9	0,1	0,75	70	0,84	75
7	70	700	0,95	0,05	0,76	85	0,86	90
8	80	800	0,8	0,2	0,78	110	0,88	120
9	90	900	0,9	0,1	0,80	130	0,90	140
10	90	1000	0,8	0,2	0,80	165	0,91	180
11	90	1500	0,8	0,2	0,80	190	0,95	210
12	90	2000	0,8	0,2	0,80	210	1	250
13	90	3000	0,8	0,2	0,80	270	1	300

На рисунке 7.23 показаны графики зависимости числа итераций от условной ЦФ при $P(\text{OK}) = 0,8$; $P(\text{OM}) = 0,2$ и постоянном $N_p = 50$ для ПГА и ГА с самоорганизацией. Как видно из графика, после двух тысяч итераций наступает насыщение, т. е. увеличение числа итераций не улучшает значение ЦФ для ПГА. Это говорит о том, что получен локальный или глобальный минимум для данной задачи. Управляя процессом генетического поиска, с помощью адаптации и обратных связей удалось найти параметры, при которых условная ЦФ имеет наилучшее значение. На рисунке 7.24 приведены графики зависимости времени решения от числа итераций. Заметим, что, как и в других задачах оптимизации на графах, время решения практически линейно зависит от числа итераций.

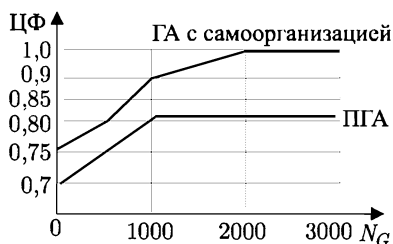


Рис. 7.23 График зависимости условной ЦФ от числа итераций

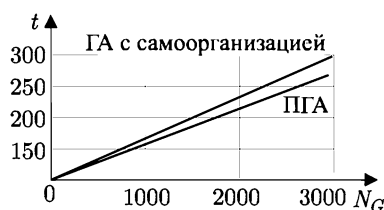


Рис. 7.24 График зависимости времени решения от числа итераций

Временная сложность алгоритма, полученная на основе экспериментов, практически совпадает с теоретическими предпосылками. Для рассмотренных тестовых задач она в лучшем случае $\approx O(n^2 \log n)$, а в худшем случае $\approx O(n^4)$.

Таблица 7.5

1	2	3	4	5	6	7	8	9
№	N_p	N_G	P(OK)	P(OM)	ПГА		ГА с самоорганизацией	
					ЦФ	t	ЦФ	t
1	10	100	0,6	0,4	0,75	8	0,80	10
2	10	500	0,6	0,4	0,76	12	0,82	15
3	10	1000	0,6	0,4	0,72	20	0,83	23
4	20	50	0,6	0,4	0,78	17	0,80	20
5	20	100	0,6	0,4	0,79	19	0,81	22
6	20	500	0,6	0,4	0,82	25	0,84	28
7	20	1000	0,6	0,4	0,82	35	0,85	40
8	30	100	0,6	0,4	0,76	26	0,78	29
9	30	500	0,6	0,4	0,80	35	0,83	40
10	30	1000	0,6	0,4	0,80	50	0,84	55
11	40	50	0,6	0,4	0,77	22	0,80	25
12	40	100	0,6	0,4	0,77	35	0,82	40
13	40	500	0,6	0,4	0,80	45	0,84	50
14	50	100	0,6	0,4	0,82	50	0,87	6
15	50	500	0,6	0,4	0,85	65	0,90	40
16	50	1000	0,6	0,4	0,86	80	0,91	85

Задача раскраски графа — это разбиение множества вершин графа на подмножества несвязанных вершин. Количество подмножеств таких разбиений определяет число «цветов» для раскраски графа. Для экспериментального исследования описанных алгоритмов раскраски графов были проведены тестовые испытания. На их основе построена таблица 7.5, в которую сведены данные о тестовых задачах раскраски графов: количество вершин графа $n = 100$, количество ребер $m = 400$. В таблице в столбце 2 указан размер популяции, в 3 — число итераций, в 4, 5 — вероятности ОК, ОМ, в 6 и 7 приведены значения условной ЦФ как отношение количества определенных и существующих цветов и время решения в (сек) для ПГА. В 8 и 9 столбцах — значения условной ЦФ и время для ГА с самоорганизацией.

На рисунке 7.25 приведены графики зависимости значения условной ЦФ от числа генераций, а на рисунке 7.26 — график зависимости времени решения при $N_p = 50$ от числа итераций. Как следует из графиков, количество итераций почти не влияет на качество решения. Зато вероятность

применения операторов и их последовательность, порядок реализации ГА является решающим фактором. Проведены эксперименты для десятков популяций с направленной селекцией и всегда происходил выход из локальных оптимумов. Отметим, что важно знать хотя бы вид или поведение критерия оптимальности в зависимости от числа элементов или структуры модели. Кроме использования стандартных операторов, проводились эксперименты с модифицированными операторами. При этом время решения в общем случае возрастало не линейно, а квадратично и кубично.

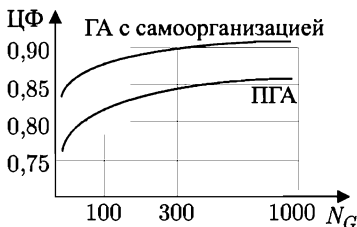


Рис. 7.25 Графики зависимости условной ЦФ от числа итераций

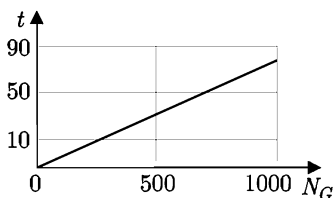


Рис. 7.26 График зависимости времени решения от числа итераций

Для тестовых графов на 100, 200, 300, 400 и 500 вершин были проведены испытания на основе классических последовательных, итерационных и комбинированных методов. При этом во всех случаях условная ЦФ была в среднем хуже чем в ГА с самоорганизацией, на 30%, 25%, 15% соответственно. Для рассмотренных тестовых задач временная сложность алгоритма в лучшем случае $\approx O(n^2)$, а в худшем случае $\approx O(n^4)$. Эксперименты показали, что не всегда увеличение размера популяции улучшает ЦФ. Очевидно, что существует какой-то определенный размер популяции, соответствующий оптимальному значению ЦФ, отыскать который чрезвычайно трудно.

7.6. Задачи распознавания изоморфизма графа

Задача распознавания изоморфизма графов является одной из важнейших комбинаторно-логических задач. Проблема определения изоморфизма двух графов, расположенных на плоскости, заключается в получении одного графа из другого путем перенумерации их вершин. Другими словами, необходимо найти подстановку множество вершин, переводящую один граф в другой. Как отмечалось выше, наибольшую трудность при определении изоморфизма представляют однородные графы. Поэтому экспериментальные исследования проводились при определении изоморфизма однородных графов. Было случайно сгенерировано сто однородных графов, а затем в этих же графах выполнена случайная перестановка их вершин. Следовательно, заранее было известно, что анализируются изоморфные графы, в которых нужно определить подстановку, переводящую один граф в другой. В таблице 7.6 в столбце 2 приведено количество вершин исследуемых графов. В столбце 3 указаны локальные степени вершин графов.

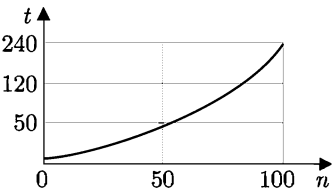


Рис. 7.27. График зависимости времени решения от числа вершин

Применение новых и модифицированных операторов позволяет ускорить процедуру поиска подстановки изоморфизма внутри подмножеств предполагаемо изоморфных вершин.

В столбце 4 приведены размеры популяции при определении части подстановок в автоморфных подграфах.

На рисунке 7.27 приведен график зависимости количества вершин графов от времени решения. На основе полученных данных можно определить, что экспериментальная временная сложность алгоритма ориентировочно составляет $O(n^3)$, хотя в худшем случае может составлять $O(k!)$.

Таблица 7.6

1	2	3	4	5	6	7
№	n	Степень $P(x_i)$	N_P	N_G	$P(OK)$	t
1	10	5	5	100	0,9	5
2	20	8	10	300	0,9	10
3	30	10	15	500	0,9	20
4	40	15	20	700	0,85	30
5	50	20	25	1000	0,85	40
6	60	25	30	1200	0,8	60
7	70	30	35	1500	0,8	90
8	80	35	40	1800	0,75	120
9	90	40	45	2000	0,75	180
10	100	50	50	2500	0,7	240

На рисунке 7.28 построен график зависимости времени получения лучшего решения для оптимизационных задач на графах от количества вершин.

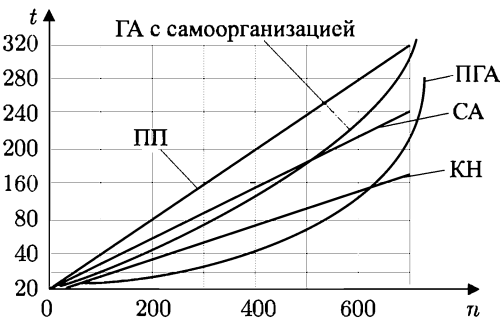


Рис. 7.28. График зависимости времени решения от числа вершин для различных алгоритмов решения оптимизационных задач на графах

Построим таблицу 7.7 качественного сравнения различных алгоритмов решения оптимизационных задач. Анализ графика и приведенной таблицы позволяет отметить, что генетические алгоритмы требуют больших затрат времени, чем, например, случайные и последовательные алгоритмы, но ПГА и ГА с самоорганизацией позволяют получать набор локально-оптимальных решений и, в частном случае, оптимальных решений.

Таблица 7.7

1	2	3	4
№	Название алгоритма	Результаты решения оптимизационных задач	Скорость алгоритма (Временная сложность алгоритма)
1	Полного перебора	Точные, один оптимальный результат	Очень медленные ($O(n!)$)
2	Статистические методы оптимизации	Один квазиоптимальный результат	Очень медленные ($O(n^4) \div O(n^6)$)
3	Итерационные алгоритмы (ПП)	Один локально-оптимальный результат	Средне медленные ($O(n \log n) \div O(n^3)$)
4	ГА с самоорганизацией	Набор квазиоптимальных и возможно оптимальных результатов	Средне медленные ($O(n \log n) \div O(n^3)$)
5	Простые генетические алгоритмы	Набор квазиоптимальных результатов	Средне быстрые ($O(n \log n) \div O(n^2)$)
6	Случайные алгоритмы (СА)	Набор результатов	Средне быстрые ($O(n \log n) \div O(n)$)
7	Последовательные алгоритмы (КН)	Один результат	Быстрые $O(n)$

Экспериментальные исследования показали, что условная ЦФ для них всегда принимает лучшие значения. Как видно из таблицы 7.7, ПГА и ГА с самоорганизацией по скорости практически совпадают с итерационными и несколько хуже последовательных, причем они значительно быстрее, чем методы полного перебора, статистические методы оптимизации и их различные модификации. В отличие от всех рассмотренных алгоритмов — ГА с самоорганизацией позволяют получать набор квазиоптимальных и оптимальных результатов.

По результатам тестовых испытаний сделаем краткие выводы:

1. Реализация ГА с самоорганизацией при разбиении графов на части показала преимущество по сравнению с ПГА и другими классическими методами.

2. Управление процессом генетического поиска при разбиении позволяет находить оптимальные параметры.

3. Применение совместных моделей эволюций, различных методов поиска и модифицированных генетических операторов позволяет повысить качество и уменьшить время размещения.

4. Использование жадных стратегий и новых операторов для задачи коммивояжера позволяет улучшать качество решений.

5. Применение нестандартных архитектур генетического поиска позволяет находить существующие подстановки внутри подграфов и эффективно решать задачи установления изоморфизма графов, раскраски графов, построения независимых подмножеств.

6. Анализ полученных данных позволяет в общем случае получать набор оптимальных решений при незначительном увеличении времени работы алгоритмов.

7. Проведенные серии тестов и экспериментов позволили уточнить теоретические оценки временной сложности алгоритмов оптимизационных задач и их поведение для графов различной структуры. Проведенные комплексные исследования показали улучшение работы предложенных архитектур генетического поиска по сравнению с известными методами.

Три пути ведут к знанию: путь размышления — это
путь самый благородный; путь подражания — это
путь самый легкий и путь опыта — это путь самый
горький.
Конфуций

8.1. Задачи планирования кристалла и сжатия топологии

Задача размещения вершин графа на плоскости связана с инженерной задачей, возникающей при проектировании топологии сверхбольших интегральных схем — планированием кристалла СБИС. Это — одна из задач, определяющая качество конструкторского проектирования СБИС. Задача планирования кристалла состоит в назначении данного множества модулей на плоскости с минимизацией суммы площади при размещении заданных прямоугольников и минимизации суммарной длины цепей, которые должны соединять модули. Тогда проблема планирования может быть определена как оптимизационная проблема, и, следовательно, для ее решения можно использовать методы ЭМ.

В [172] данная задача формулируется следующим образом. Дано множество модулей $M = \{1, 2, \dots, m\}$. Все модули представляются прямоугольниками. Каждый i -й модуль характеризуется кортежем длины три $\langle S_i, l_i, u_i \rangle$, где S_i — площадь модуля, l_i, u_i — верхние и нижние границы отношения высоты h_i к ширине w_i . Здесь $w_i \times h_i = S_i$ и $l_i \leq h_i/w_i \leq u_i$. Планирование состоит в разбиении плоскости горизонтальными и вертикальными линиями на m непересекающихся областей. Каждая область r_i имеет ширину x_i и высоту y_i , такие, что $S_i \leq x_i \times y_i$ ($x_i \geq w_i, y_i > h_i$). Результат альтернативного планирования может оцениваться посредством ЦФ, учитывающей общую площадь компонента и суммарную длину соединений. Общая площадь определяется как

$$S = \sum_{i=1}^m x_i \times y_i,$$

а длина соединений определяется по формуле (6.7):

$$L(G) = \frac{1}{2} \sum_{i=1}^m \sum_{j=1}^m d_{ij} c_{ij}.$$

Здесь c_{ij} — число связей, соединяющих i -й и j -й модули, d_{ij} — расстояние между этими модулями. Для получения математической модели плана кристалла используют так называемую кусочную структуру или «польские

выражения» [172]. На рис. 8.1, *а* показан план кристалла, на рис. 8.1, *б* — его кусочная структура и на рис. 8.1, *в* — «польское выражение». Модели кусочной структуры или «польского выражения» находят применение при решении задач планирования. Эти модели используются при построении популяций в ГА.

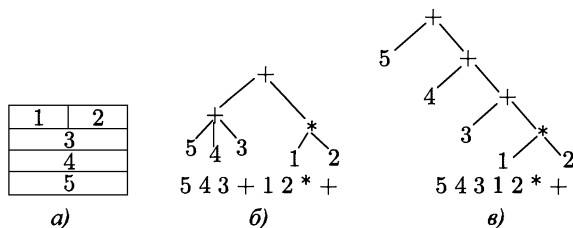


Рис. 8.1. План кристалла и его модели

Структура ГА для решения данной задачи описывается следующим образом. Сначала создается одна или несколько популяций решений, что соответствует $P_a = \{p_{a1}, p_{a2}, \dots, p_{ak}\}$, $k \approx n/2$, где n — число элементов (модулей), $P_b = \{p_{b1}, p_{b2}, \dots, p_{bk}\}$ и т. д. Популяции, как уже отмечалось выше, создаются случайным, направленным или направленно-случайным образом. Следует заметить, что в популяции будут оставаться только реальные решения. При наличии сети ЭВМ каждую генерацию можно параллельно реализовывать на своем процессоре. Рассмотрим случай наличия нескольких популяций. Произведем сортировку хромосом в каждой из популяций. Затем применим описанные схемы селекции для выбора родительских пар, причем в каждой паре родители должны быть из разных популяций. В [172] описаны четыре модифицированных ОК. Для повышения качества решения кроме рассмотренных генетических операторов используются все описанные выше модифицированные операторы. Например, пусть заданы хромосомы (планы кристалла). Применяя к ним модифицированные операторы кроссинговера, транслокации, мутации и др. получим следующий результат (хромосомы p'_1, p'_2, p''_1, p''_2):

p_1 :	1	5	8	7	*	+	+	2	4	3	*	*	+	6	+
p_2 :	2	6	8	*	*	4	+	1	3	*	+	7	5	*	+
p'_1 :	1	5	8	7	*	+	+	3	4	2	*	*	+	6	+
p'_2 :	8	6	2	*	*	4	+	1	3	*	+	7	5	*	+
p''_1 :	7	8	5	1	*	*	+	3	4	2	*	+	*	6	+
p''_2 :	8	6	2	*	+	4	+	3	1	*	*	5	7	+	+

Каждой хромосоме соответствует определенный план кристалла. Основная задача генетических операторов — найти строительные блоки с лучшими характеристиками для копирования их в потомки. С этой точки зрения для задач планирования эффективными оказались операторы сегрегации, мутации и модифицированной инверсии.

Трудоёмкость алгоритма ориентировочно определена так:

$$T \approx T_M + [N_p t_p + (N_p + N_{II})(t_{ок} + t_{ос} + t_{ом} + t_{он}) + N_p t_c] N_G,$$

где T_M — трудоёмкость реализации модели, например, в виде польской записи; t_c — трудоёмкость селекции; $t_{ок}$ — трудоёмкость оператора кроссинговера; t_p — трудоёмкость построения одной хромосомы в популяции; $t_{ом}$ — трудоёмкость оператора мутации; $t_{он}$ — трудоёмкость оператора инверсии; $t_{ос}$ — трудоёмкость оператора сегрегации; N_G — число генераций, N_p — размер популяции, N_{II} — число полученных потомков.

Задачи планирования кристалла тесно связаны с оптимизационными задачами упаковки и сжатия. После выполнения планирования кристалла часто решаются вопросы сжатия топологии СБИС. Сжатие топологии заключается в конвертировании элементов электрических схем в соответствующие элементы топологии и минимизации площади кристалла [117, 159].

Проектировщики сталкиваются с тремя противоречивыми требованиями:

- спроектировать минимизированную по площади топологию;
- максимизировать функциональные возможности СБИС;
- минимизировать время и стоимость проектирования.

Проблема сжатия топологии связана с методологиями топологического проектирования (размещение компонентов и проводников) и геометрического проектирования (физическое размещение элементов топологии). Задачи сжатия обычно решаются на основе символической топологии. Символическая топология дает возможность проектировщику анализировать такие компоненты СБИС, как транзисторы, проводники, ячейки и т. п. Символическая топология обычно транслируется в топологию в виде прямоугольников различного размера. В САПР такая топология представляется в виде диаграмм, представляющих проводники и элементы как различного рода компоненты. Следовательно, использование методов сжатия на основе символической топологии и диаграмм позволяет менять геометрию топологии с минимизацией площади кристалла. В работе [173] дан детальный обзор и классификация методов сжатия. Проблемы сжатия, как и остальные проблемы проектирования топологии, относятся к классу комбинаторно-логических задач и являются NP-полными.

Алгоритмы сжатия обычно классифицируются по двум направлениям. Первое направление показывает, как движутся элементы в процессе сжатия. К этому классу относятся алгоритмы одномерного сжатия (1-Д), двухразмерного (2-Д), трехразмерного (3-Д) и специального сжатия типа (1,5-Д). Сжатие типа 1-Д обычно выполняется только в s или t направлениях, как показано на рис. 8.2, а, б. Проблема 1-Д-сжатия NP-полная. Реализация 2-Д

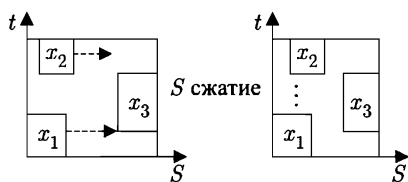


Рис. 8.2. а. Пример 1-Д сжатия топологии по оси S

сжатия происходит одновременно по осям s и t . Второе направление показывает, каким образом производится размещение компонентов. Здесь выделяют три класса алгоритмов. Это — алгоритмы сжатия на основе графов ограничений, виртуально решетчатые алгоритмы и алгоритмы иерархического сжатия. Основными алгоритмами

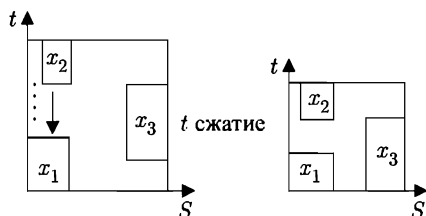


Рис. 8.2. б. Пример 1-Д сжатия топологии по оси t

здесь являются: алгоритмы сжатия на основе графа прямоугольников, алгоритм Айкерса или алгоритм «тени» [173]. Иерархические алгоритмы заключаются в многоуровневом сжатии. Элементы топологии, подлежащие сжатию, обычно описываются в виде прямоугольников: $ЭT_1 = (w_i, h_i, s_i, t_i, m_i)$, здесь w_i — ширина прямоугольника; h_i — высота; s_i, t_i — декартовы координаты; m_i — номер элемента ($1 \leq m_i \leq m$). Расстояния между элементами на плоскости измеряются по формулам (6.5, 6.6). Задача состоит в размещении прямоугольников таким образом, чтобы они не пересекались и площадь, где они расположены, была минимизирована. В такой постановке задачи сжатия переключаются с задачами планирования кристалла, а также с задачами разбиения.

Существующие методы сжатия на основе стохастических методов, моделирования отжига и генетического поиска используют в качестве исходных описанные эвристики.

В работе [173] показана возможность сжатия символической топологии с применением ГА. Сжатие представляется как задача 2-Д. Множество прямоугольников размещается на плоскости, удовлетворяя заданным ограничениям и минимизируя стоимостную функцию площади кристалла. Прямоугольники размещаются на плоскости согласно списку элементарных ограничений. Набор списков, полученных случайно, представляет собой популяцию для реализации ГА. Селекция популяции производится случайно. Выполняются стандартные операторы кроссинговера и мутации. Алгоритм выполняется в двух иерархических уровнях (поверхностный и глубокий). Здесь используется селекция для сравнения списков ограничений. Достоинством алгоритма является простота реализации. Недостатки связаны с предварительной сходимостью алгоритма и невысокими результатами по минимизации площади.

Алгоритм 2-Д-сжатия топологии на основе моделирования отжига сначала уменьшает размер исследуемой поверхности, а затем на основе моделирования отжига выбирает множество ограничений. Основным недостатком состоит в том, что начальное решение выбирается случайно, а оно, в основном, влияет на конечный результат. Сложность алгоритма лежит в пределах $O(n^2) \div O(n^1)$.

В той же работе описано решение задачи 2-Д-сжатия на основе ГА. Целевая функция определяется как максимум средней заполняемости заданной

плоскости. Достоинством алгоритма является введение нового ОК, ориентированного на решение задач сжатия. Данный оператор кроссинговера напоминает оператор сегрегации, выполненный направленным образом.

Стохастический метод для решения 1-Д-сжатия представляет собой объединение генетического поиска с процедурами моделирования отжига. Основная идея этого метода — повышение скорости моделирования отжига за счет использования простых стандартных операторов ГА. Для получения популяции решений используются две стратегии — «жадная» и случайная. Основным недостаток — требование больших затрат на вычислительные процедуры отжига, а также быстрая сходимость алгоритма с попаданием в локальный оптимум.

Предлагается иерархическая процедура решения задач сжатия топологии с использованием параллельного генетического поиска. Опишем кратко схему генетического поиска. Как и во всех описанных ранее алгоритмах, первоначально производится конструирование некоторого множества (четырёх) популяций. В качестве элемента популяции выбирается порядок (реальный) расположения фрагментов топологии и соединений на плоскости, причем каждое соединение представляется прямоугольником, как и размещаемый элемент. Другими словами, каждая популяция будет представлять собой набор списков размещенных элементов и их соединений. Первая популяция будет конструироваться случайным образом, вторая — на основе «жадной» стратегии, третья — на основе последовательных эвристик и четвертая популяция — смешанная (направленно-случайная). В четвертой популяции половина элементов выбирается случайно, а вторая — используя знания о решаемой задаче. Согласно схеме, описанной выше, производим сортировку элементов популяций $P_1 - P_4$ согласно выбранной целевой функции. ЦФ выбрана как стоимостная функция, минимизация которой уменьшит среднюю площадь, занятую компонентами топологии и их соединениями. Производится элитная селекция родителей из каждой популяции. Далее применяются три ОК. Операторы ОК применяются таким образом, чтобы каждый раз получались реальные решения. После выполнения ОК производится сортировка и устранение такого числа наихудших элементов популяций, чтобы размер популяций оставался постоянным. При получении удовлетворительных результатов процесс завершается. В противном случае возможно управление процессом поиска за счет перехода на другой оператор или изменения вероятности применения оператора. Данные процедуры выполняются итерационно и число генераций зависит от наличия вычислительных ресурсов. Трудоемкость алгоритма ориентировочно определена как

$$T = N_G[N_p t_p + (t_{ok} + t_{om} + t_{or} + t_{oc} + t_{on})N_{II}],$$

временная сложность алгоритма лежит в пределах $O(n \log n) \div O(n^3)$.

Таким образом, увеличить вероятность получения оптимального решений задач планирования и сжатия топологии можно за счет управления процессом генетического поиска, путем изменения (определенного увеличения до попадания в локальный оптимум) размера популяций, количества

генераций, вероятности применения генетических операторов, искусственной миграции хромосом, направленной селекции и отбора. Отметим, что эти задачи также являются NP-полными. Описанные эвристики помогают находить значительное число локальных решений, с большой вероятностью нахождения оптимального результата.

8.2. Упаковка блоков

Задача упаковки блоков является одной из важнейших инженерных задач. Она связана с распределением заданных блоков различного размера по стандартным блокам. Она является NP-полной задачей, т. е. не существует оптимального алгоритма для задачи упаковки блоков с полиномиальной временной сложностью.

Задача упаковки в классической постановке определяется следующим образом. Дано конечное множество чисел (размеров элементов) E и константа MAX (размер блока), цель: найти разбиение множества E на минимальное количество подмножеств, таких, что сумма элементов в каждом подмножестве не будет превышать MAX (размера блока).

Отметим, что задача упаковки блоков тесно связана с задачей разбиения графов на части с минимизацией суммарного числа связей K , когда заданное число объектов надо разместить в N блоках с выполнением заданных ограничений и с минимизацией числа блоков. Приведем набор основных эвристик для решения этой задачи [186]:

- FF (First Fit) — это простейшая эвристика, когда, начиная с одного пустого блока, берется объект за объектом и для каждого из них ищется подходящий блок. Если такой блок найден, то в него помещается объект, если не найден, то выбирается другой блок и процесс продолжается до распределения всех объектов.
- FFD (First Fit Decreasing) — в этой эвристике предполагается по заданным предварительно критериям отсортировать объекты, а далее применить эвристику FF.
- BF (Best Fit) — здесь происходит поиск самого заполненного блока, а далее применяется эвристика FF.
- BFD (Best Fit Decreasing) — здесь реализуется эвристика «лучший подходящий в порядке убывания».

Запишем модифицированный FF для задачи упаковки блоков. Пусть подмножества стандартных блоков пронумерованы $X_1, X_2, \dots$ и каждый из них не содержит вершин (блоков, которые должны быть размещены). Вершины $x_1, x_2, \dots$ будут располагаться в блоках $X_1, X_2, \dots$ в указанном порядке. Чтобы разместить x_i , необходимо найти наименьший из X_j такой, что X_j заполнен до уровня $\alpha \leq S(X_j) - S(x_i)$ и разместить x_i в X_j , где $S(X_j)$ — размер (мощность) блока X_j . Теперь X_j заполнен до уровня $\alpha + S(x_i)$. Другими словами, вершина x_i помещается в первый из блоков, куда она может войти без нарушений ограничений по размеру и значению ЦФ Q . Критерий Q комплексный, так как учитывает число заполненных блоков и площадь. Его необходимо минимизировать.

Алгоритмы FFD и BF для построения популяции альтернативных решений задачи упаковки блоков реализуются аналогично. Отметим алгоритм BF; в худшем случае он имеет те же характеристики, что и FF. Для FFD известно, что даже в худшем случае он выдает решения, когда Q отличается от оптимального не более, чем на 22%, причем существуют задачи упаковки блоков, для которых $Q \leq \frac{11}{9}Q_{\text{опт}}$. Оценка BFD в худшем случае совпадает с оценкой работы алгоритма BF. Для построения популяции альтернативных решений задачи упаковки блоков применяют также модификацию FF, называемую RFF (refined first fit) — первый подходящий с удалением. Здесь множество X всех блоков разбивается на 4 подмножества и после упорядочивания всех вершин выполняются пробные помещения вершин в указанные блоки. Оценка RFF: $Q \leq \frac{5}{3}Q_{\text{опт}}$. Например, начиная с одного пустого блока, брать последовательно элементы и для каждого из них искать уже используемый блок с достаточным свободным местом. Если такой блок найден, то поместить в него элемент, иначе взять пустой блок. Это эвристика FF. Если элементы перед распределением отсортировать, то получим эвристику FFD, которая решает задачу лучше, но работает медленнее. А если искать самый заполненный блок, имеющий достаточно места, то это — эвристика BF [186].

Задача упаковки является членом семейства задач группировки, цель которых состоит в разбиении множества элементов E в непересекающиеся подмножества E_i , такие что $\bigcup U_i = U$, $U_i \cap U_j = \emptyset$, $i \neq j$. Цель этой задачи — сгруппировать члены множества U в одну или больше (максимум $|U|$) групп элементов, причем каждый элемент должен находиться только в одной группе. В большинстве задач не все возможные группировки допустимы. Обычно элемент не может быть сгруппирован со всеми возможными подмножествами оставшихся элементов. Цель задачи — оптимизировать целевую функцию, определенную на множестве всех реальных решений.

Опишем группирующий ГА для решения задач одномерной упаковки. Сначала определим целевую функцию, а затем опишем кодировку хромосом и генетические операторы. Целевую функцию определим так:

$$f_{\text{вpp}} = \frac{\sum_{i=1}^N (S_i / \max)^k}{N},$$

где N — число используемых блоков, S_i — сумма размеров элементов в (заполненном) блоке i , $\max$ — размер блока, $k \sim$ константа, $k > 0$. Требуется максимизировать $f_{\text{вpp}}$. Константа k выражает концентрацию на «хорошо» заполненные блоки в сравнении с менее заполненными. Если увеличиваем k , значит, предпочитаем хорошо заполненные «элитные» блоки в сравнении с набором примерно одинаково заполненных блоков. Увеличение k ведет к предварительной сходимости алгоритма в локальном оптимуме.

Очевидно, что в задаче упаковки блоков следует применять групповой подход со знаниями о решаемой задаче, так как целевая функция зависит

от порядка размещения групп элементов. При этом хромосома (альтернативное решение) будет состоять из двух частей:

- элементной части (фиксированной длины, равной числу элементов);
- групповой части (переменной длины).

Например, в хромосоме ACBADC|CBDA часть ACBADC — элементная, а CBDA — групповая. Элементная часть будет кодировать следующее частичное решение (рис. 8.3):

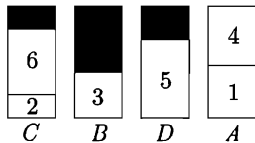


Рис. 8.3. Частичное решение

В элементной части ACBADC хромосомы будет кодироваться решение, в котором первый элемент лежит в блоке А, второй — в блоке С, третий в — В, четвертый в — А, пятый в — D, шестой — в блоке С. Групповая часть хромосомы представляет только группы (блоки в задаче упаковки блоков) и порядок их следования. В рассматриваемой выше хромосоме $A = \{1,4\}$, $B = \{3\}$, $C = \{2,6\}$, $D = \{5\}$. И хромосома может быть записана так: $\{2,6\} \{3\} \{5\} \{1,4\}$. Далее генетические операторы будут работать с групповой частью хромосом, а элементная часть будет служить только для определения, какие элементы формируют группу. Для задачи упаковки блоков опишем оператор скрещивания с заменой [187]. Его работу покажем на примере. Даны две хромосомы родителя p_1 : BFECFADCCEDA|ABCDEF (рис. 8.4) и p_2 : bcdcbcaaddacd | abcd (рис. 8.5).

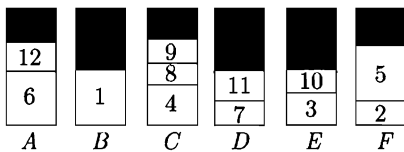


Рис. 8.4. Первый родитель

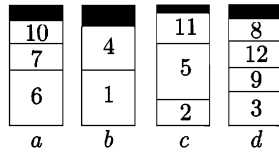


Рис. 8.5. Второй родитель

Возьмем групповые части двух хромосом и выберем случайным образом две точки разрыва в каждой:

$$\begin{aligned} p_1 : & \quad A \quad | \quad B \quad C \quad D \quad | \quad E \quad F \\ p_2 : & \quad a \quad | \quad b \quad c \quad | \quad d \end{aligned}$$

Процесс формирования первого потомка p_3 следующий: блоки между точками разрыва в p_2 вставляются в p_1 после первой точки разрыва. Формирование второго потомка происходит аналогично. Если некоторые элементы в p_3 встречаются дважды, то они устраняются (рис. 8.6). Получим, что p_3 : A b c E (рис. 8.7).

Как видно после устранения элементы 7, 8, 9 не присутствуют в блоках. Для вставки в решение пропущенных элементов (рис. 8.8) используем

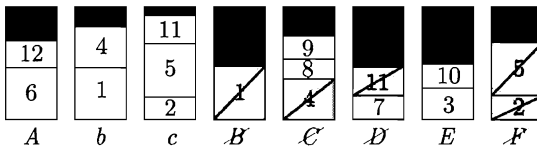


Рис. 8.6. Частичное решение

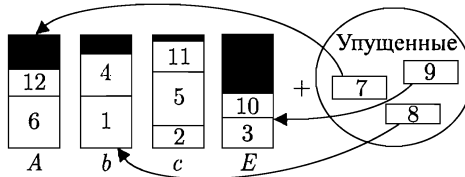


Рис. 8.7. Частичное решение

FFD-эвристику. Здесь сначала выполняется локальная оптимизация по следующему принципу. Берутся один за другим блоки, уже присутствующие в решении, выполняется проверка, возможно ли заменить элемент в блоке на упущенный элемент таким образом, чтобы общий размер элементов в блоке увеличился без переполнения блока. Если такое возможно, то выполняется замена. Замена имеет две особенности. Во-первых, блок заполняется лучше, чем он был до этого. Во-вторых, уменьшается размер пропущенных элементов, и, следовательно, для них будет легче найти место в блоках, которые уже присутствуют в решении. Этот процесс продолжается до тех пор, пока такие замены возможны.

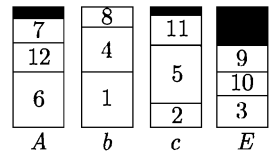
$$bcEbcAAbEEcA:AbcE$$


Рис. 8.8. Альтернативное решение

Опишем оператор мутации, ориентированный на задачу упаковки блоков. В данной хромосоме выбираются, по случайному закону, несколько блоков и устраняются из решения. Естественно, появляются упущенные элементы, которые должны быть помещены в хромосому. Упущенные элементы вначале являются базисом для возможного улучшения неизменных блоков во время стадии замены. Когда замена уже невозможна, FFD-эвристика используется для завершения формирования решения.

В заключение отметим, что рассмотренные эвристики совместно с ГА для решения задачи упаковки блоков позволяют находить оптимальные и квазиоптимальные решения.

8.3. Решение задачи плоской раскладки

В производстве одними из типовых являются задачи рационального раскроя и раскладки материала. Постоянный рост цен на материалы требует минимизации отходов при производстве изделий. Как правило, в качестве критерия эффективности принимается коэффициент использования материала (КИМ). Оптимальный раскрой промышленных материалов как научное направление получил начальное развитие в трудах Л. Канторовича

[202]. Для решения задач раскроя и размещения предложено много математических методов, однако все они ориентированы на решение статической задачи, когда исходная информация заранее полностью известна и не меняется в процессе производства. На производстве лишь 40% подобных задач решается автоматизировано с привлечением строгих математических методов.

Далее рассмотрим решение задач по раскрою материала на производственном участке мебельной фабрики, где необходимо оклеивать плоские древесно-стружечные блоки дорогостоящей пленкой. Поверхность блоков, на которые наклеивается пленка, имеет форму прямоугольников. Далее из оклеенных пленкой блоков собираются кухонные гарнитуры. Размеры блоков и их количество задано и определяется портфелем заказов на гарнитуры. Пленка наклеивается на мембранно-вакуумном прессе, поверхность которого также имеет прямоугольную форму. Процесс оклейки происходит следующим образом: на поверхность пресса кладется цельная пленка; на нее кладутся блоки, которые сверху также покрываются пленкой. Затем происходит запрессовка. Стоит отметить, что блоки необходимо укладывать на поверхность пресса с учетом направленности рисунка пленки.

В зависимости от вида заказа блоки нужно оклеивать различными типами пленки. Поэтому портфель заказов делится на группы по типам пленки и для каждой группы отдельно решается задача по раскрою материала.

Если не рассматривать организационные особенности производства, то задача по раскрою материала формулируется так: необходимо выбрать из портфеля заказов некоторое количество блоков и с учетом направленности пленки уложить их на поверхность плиты пресса; при этом необходимо добиться уменьшения потерь пленки. Так как потери материала должны быть минимальными, соответственно КИМ необходимо максимизировать. Он определяется согласно следующей формуле:

$$\text{КИМ} = S_{\text{бл}}/S_{\text{пл}},$$

где $S_{\text{бл}}$ — суммарная площадь всех блоков, уложенных на поверхность плиты пресса перед прессованием; $S_{\text{пл}}$ — площадь поверхности плиты пресса.

Для того, чтобы оценить качество расчетного (полученного в результате решения задачи) коэффициента, необходимо знать его теоретически максимальное значение, поскольку во многих случаях достижение КИМ = 1 невозможно. Зная это значение, и сравнив его с полученным, при решении задачи, можно сделать вывод о качестве полученного раскроя. Для решения задачи используем гибридную систему, включающую имитационную модель и блок генетической оптимизации (см. рис. 5.2 и 5.3).

Имитационная модель описывает работу системы, реализующей рациональную раскладку блоков. Исходной информацией для модели является портфель заказов, состоящий из определенного набора блоков, для которых заданы размеры (т. е. длина и ширина) и номера в портфеле. В результате работы модель должна осуществлять укладку блоков на поверхность плиты, а по окончании работы выдать рациональное значение КИМ, ко-

торое должно быть близким к оптимальному или равно ему. Для решения задачи оптимизации КИМ использован ПГА. Основу имитационной модели составляет алгоритм, реализующий укладку блоков. Он описывает работу системы, т. е. проверяет возможность укладки очередного блока и осуществляет ее. Исходными данными для него являются значения номеров блоков, которые выдает алгоритм оптимизации, т. е. ГА [203]. Каждый блок имеет в портфеле заказов свой идентификационный номер, который точно его определяет. Для реализации процесса укладки блоков на поверхность плиты пресса необходимо определить номера блоков, которые нужно выбрать из портфеля заказов, и найти очередность их укладки на пресс. Каждому блоку в портфеле заказов ставится в соответствие его порядковый номер, который принимает значения от 0 до количества блоков в портфеле заказов. Именно с порядковыми номерами и работает оптимизационный алгоритм, причем один набор порядковых номеров представляет собой отдельное решение, которое реализуется имитационной моделью и для которого определяется КИМ.

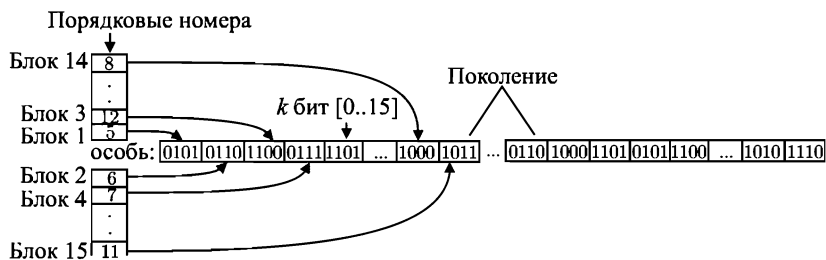


Рис. 8.9. Способ кодирования укладки блоков

При каждом запуске система должна автоматически определять длину строки-хромосомы. Количество генов в каждой особи равно числу блоков в таблице невыполненных заказов. Способ кодирования значений номеров блоков в двоичную форму для работы генетического алгоритма представлен на рис. 8.9. Предположим, что в текущий момент в портфеле заказов находится 15 блоков. Тогда особь представляет собой битовую строку-хромосому длиной 60 бит. Гены в этой строке имеют длину по 4 бита. Гены представляют собой закодированные значения порядковых номеров блоков. Каждый ген имеет длину 4 бита из условия возможности кодирования максимального номера блока (в данном случае 4 бита обеспечивают кодирование в двоичной форме числа 15). Количество генов равно числу блоков, т. е. равно 15. Таким образом, каждая особь — это одно решение, в которой отдельный ген определяет порядковый номер для соответствующего блока.

Оптимизируемой величиной для генетического алгоритма является целевая функция, рассчитываемая для особей. Поэтому необходимо выбрать такую целевую функцию, которая растет с увеличением значения критерия, в качестве которого используется КИМ. Данная функция была выбрана на основании проведенных экспериментов с моделью, исходя из условия обеспечения правильного развития популяции. Она равна: $\text{ЦФ} = (\text{КИМ})^5$.

Основные параметры генетического алгоритма выбирались после проведения предварительных экспериментов с имитационной моделью, и их значения были приняты следующими:

- количество особей в популяции — 30;
- вероятность скрещивания — 0,6;
- вероятность мутации — 0,3.

Момент остановки работы генетического алгоритма определялся из условия:

$$((\text{ЦФ}_{\max} - \text{ЦФ}_{\text{ср}}) / \text{ЦФ}_{\max}) \times 100 < 5\%,$$

где $\text{ЦФ}_{\max}$ — максимальное значение целевой функции в текущей популяции; $\text{ЦФ}_{\text{ср}}$ — среднее значение целевой функции в текущей популяции.

Алгоритм укладки блоков на поверхность пресса является основой имитационной модели. Исходными данными для его работы является набор порядковых номеров блоков, рассчитанных генетическим алгоритмом (рис. 8.10).

Рассмотрим функции отдельных блоков алгоритма:

- **Расшифровка битов.** Здесь осуществляется расшифровка строки-хромосомы, т. е. определение порядковых номеров.
- **Выбор очередного порядкового номера и существуют ли блоки с данным порядковым номером?** Первый блок схемы ведет счет порядковых номеров блоков в портфеле заказов. Второй проверяет, есть ли блоки с данным порядковым номером. Если такие (такой) блоки существуют, то идем дальше; если нет, то определяем следующий порядковый номер.
- **Выбор блока из числа возможных.** Здесь для последующей укладки выбирается один блок из всех блоков с такими же порядковыми номерами. Для укладки выбирается тот блок, номер которого в портфеле заказов наименьший. Это означает также, что блоки с такими же порядковыми номерами уже выходят из рассмотрения на укладку для данной особи.
- **Существуют варианты укладки блока? и Укладка блока.** Если не существует вариантов укладки блока на поверхность плиты пресса, то данная особь заканчивает обрабатываться. Если такие варианты укладки существуют, то происходит укладка одним из двух способов. Укладка блока производится по наименьшей координате, т. е. когда расстояние от левого верхнего угла блока до левого верхнего угла поверхности пресса наименьшее.
- **Данный блок последний? и Расчет ЦФ.** Если данный блок последний, то производится расчет ЦФ для особи, если нет, то производится выбор блока со следующим порядковым номером.
- **Особь последняя?** В данном блоке проверяется, является ли текущая особь последней. Если она не является последней, то происходит обработка следующей особи, если же данная особь последняя, то алгоритм укладки на время завершает свою работу и включается генетический оптимизационный алгоритм.

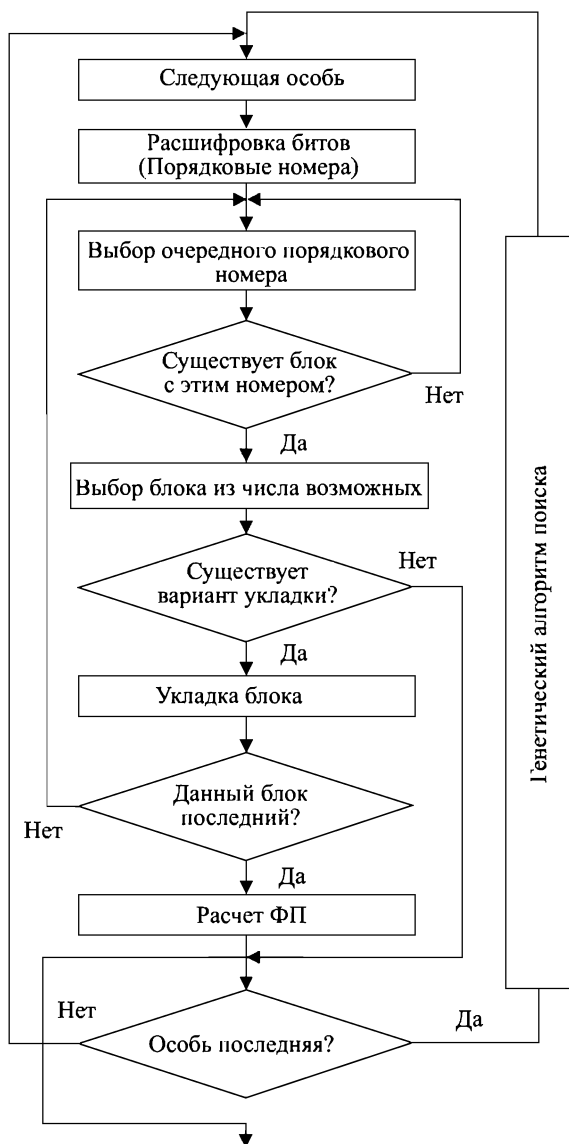


Рис. 8.10. Схема работы алгоритма

ет целевой функции для особи при работе ГА осуществляемой модели. То есть, для каждой особи осуществляется выбор по окончании которой для нее рассчитывается значение функции. Для статистики фиксируется особь с наибольшим значением функции в популяции, и, если она является лучшей по всем поколениям, то фиксируется как промежуточный результат решения задачи. По окончании работы алгоритма в качестве решения оптимизационной задачи выбирается лучшая особь в популяции.

После того, как сформировались карты раскладки, их можно сохранить в файле, который в дальнейшем используется на этапе функционирования участка фабрики. Реальной укладкой одна из карт раскроя приведена на рисунке 8.11.

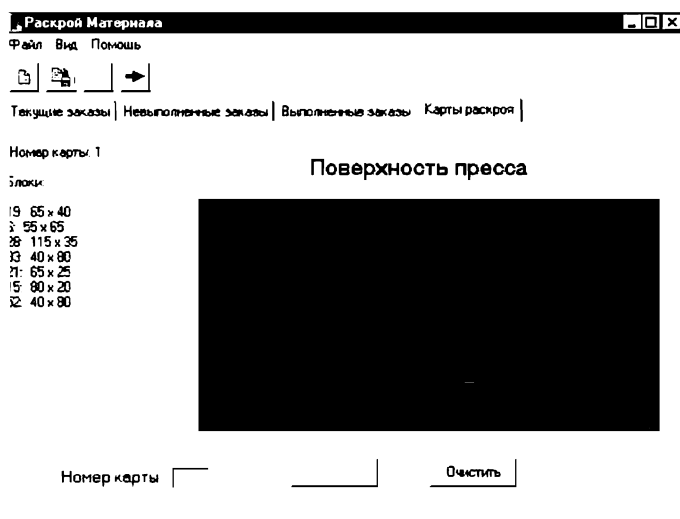


Рис. 8.11. Вид карты раскроя

Критерием оценки качества получаемого решения в данной модели является значение КИМ. Поэтому, чтобы оценить качество получаемого решения, были специально подобранные портфели заказов. Каждый портфель заказов составлялся из таких наборов блоков, для которых существуют оптимальные укладки с КИМ = 1. Для указанных выше параметров алгоритма система в 90% случаев находила решения, при этом КИМ = 0.95, причем в некоторых случаях были получены оптимальные решения, то есть, КИМ = 1.

Введенная выше гибридная система решала задачу раскроя в оптимальном варианте. На реальном производстве каждый обклеиваемый лист является составляющей отдельного кухонного гарнитура, поэтому поступающие на вход системы, разбиваются на блоки, которые образуют портфель заказов. При этом по срочности выполнения

заказы делятся на два типа: срочные и обычные. Очевидно, что срочные заказы должны выполняться в первую очередь и это необходимо учитывать при составлении карт раскроя.

Второй организационной особенностью является то, что участок фабрики выпускает готовые гарнитуры. Сроки изготовления этих гарнитуров различны, поэтому желательно, чтобы больший приоритет на укладку имели блоки того гарнитура, который наиболее близок к завершению. Данное ограничение вытекает из необходимости исключения простоев и обеспечения загрузки производства.

Для учета при принятии решений вышеперечисленных организационных факторов гибридная система принятия решений была модернизирована. Был предложен эвристический подход, согласно которому ЦФ представляется как взвешенная сумма трех параметров, которые должны отвечать за каждый организационный фактор в отдельности. Весовой коэффициент пропорционален важности параметра, входящего в формулу расчета ЦФ. Расчет ЦФ осуществляется по формуле:

$$\text{ЦФ} = a_1 K_1 + a_2 K_2 + a_3 K_3,$$

где K_1 — параметр, учитывающий срочность заказа; K_2 — параметр, учитывающий приоритет заказа; K_3 — параметр, определяющий КИМ (т. е. $K_3 = \text{КИМ}$); a_i — весовые коэффициенты, $i = 1, 2, 3$.

Для расчета параметра K_1 необходимо учитывать срочность заказа для блоков, укладываемых на поверхность прессы. На входе системы все гарнитуры делятся на два типа: срочные и обычные. Так как при составлении портфеля заказов гарнитуры разбиваются на блоки, то все блоки по срочности заказа также делятся на два типа: срочные и обычные. В модели системы при описании каждого блока присутствует параметр срочность, который является булевой переменной и может принимать значения 0 или 1. Именно эти значения используются для нахождения параметра K_1 , которое производится следующим образом: для вновь создаваемой карты параметр K_1 изначально равен 0; для каждого блока, входящего в данную карту, к значению K_1 прибавляется значение параметра срочность. Таким образом, значение K_1 равно числу срочных блоков в данной карте раскроя.

Расчет параметра K_2 осуществляется аналогично расчету параметра K_1 , только для этого используется значение приоритета блока. В описании каждого блока имеется параметр приоритет, который также является булевой переменной и может принимать значения 0 или 1. В отличие от параметра срочность, который задан и не меняется во время работы системы, параметр приоритет изначально для каждого блока имеет значение 0 и увеличивается по мере формирования карт раскроя. В данной схеме принцип увеличения приоритета задан в виде пороговой одноступенчатой функции, значение которой изменяется на 60%. Это значит, что когда число готовых блоков данного гарнитура составляет 60% или более от общего числа блоков в гарнитуре, то для оставшихся (невыполненных) блоков данного гарнитура приоритет становится равным 1.

Параметр K_3 представляет собой КИМ и рассчитывается точно так же, как и ранее.

Весовые коэффициенты a_i определяют значимость каждого из параметров, входящих в формулу для расчета ЦФ. Так как условия производства часто меняются, то для того, чтобы система была более гибкой, значения весовых коэффициентов вводятся пользователем через интерфейсный блок. Таким образом, пользователь сам может определять значимость каждого параметра в зависимости от ситуации. Для проведения исследований весовые коэффициенты были выбраны следующие: $a_1 = 10$; $a_2 = 1$; $a_3 = 5$.

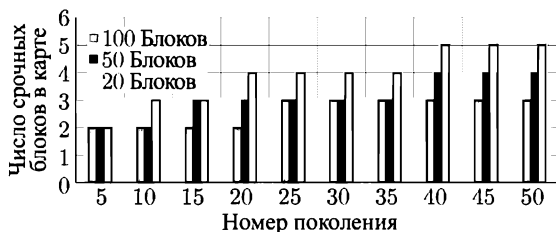


Рис. 8.12. Число срочных блоков в карте раскроя («старый» алгоритм)

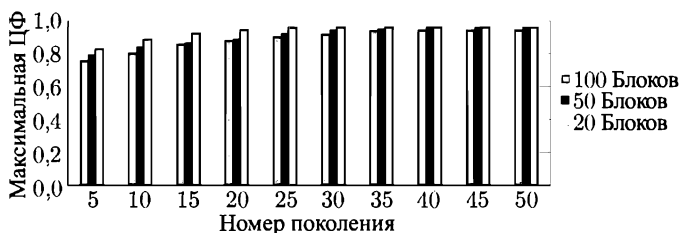


Рис. 8.13. Максимальное значение ЦФ по поколениям («старый» алгоритм)

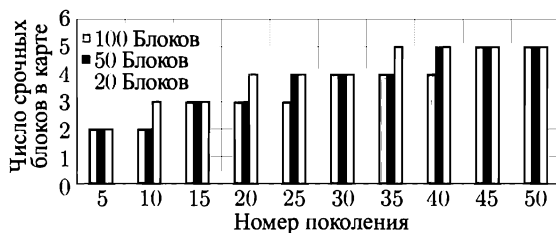


Рис. 8.14. Число срочных блоков в карте раскроя (модифицированный алгоритм)

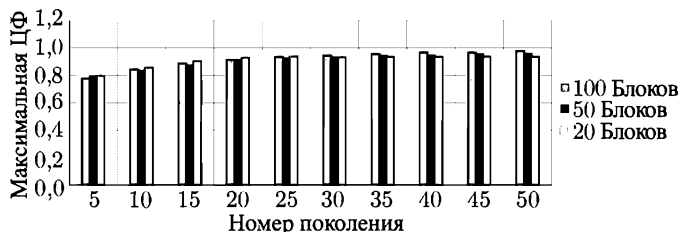


Рис. 8.15. Максимальное значение ЦФ по поколениям (модифицированный алгоритм)

В процессе имитационных исследований первой версии системы было выявлено, что ее недостатком является плохая работа на больших портфелях заказов. Очень часто решения, которые выдавал алгоритм, работая на портфелях заказов, состоящих из большого количества блоков, были хуже, чем решения, получаемые алгоритмом при работе на портфелях заказов, состоящих из небольшого количества блоков. При этом все те блоки, которые входили в меньший портфель заказов, присутствовали в большем портфеле заказов. Это значит, что теоретически карты раскроя, получаемые из большего портфеля заказов, должны быть такими же или лучше, чем карты, получаемые из меньшего портфеля заказов.

Для решения этой проблемы была предложена эвристика, которая заключается в модификации генетического алгоритма. Данная модификация оптимизационного алгоритма заключается в изменении операторов скрещивания и мутации. Поскольку в «старом» алгоритме в качестве оптимизационного алгоритма использовался ПГА, то механизм оператора скрещивания представлял собой скрещивание двух особей предков с разрывом в одной точке (точка разрыва выбирается случайно) и последующим обменом концов строк–хромосом особей предков для получения двух особей потомков. Механизм мутации представлял собой однобитную мутацию, при которой для случайно выбранной особи случайным образом определялся ген и бит в строке–хромосоме; затем данный бит инвертировался. Исследования работы алгоритма для больших портфелей заказов показали, что такая схема работы оказывает малое влияние на развитие популяции, что приводит к плохим результатам. Для решения было предложено использовать механизм многоточечного скрещивания и многобитной мутации.

Для реализации многоточечного скрещивания и многобитной мутации необходимо определить количество точек разрыва и точек мутации. Было предложено сделать оба этих параметра одинаковыми и рассчитывать по формуле:

$$K = N/D,$$

где N — количество не уложенных блоков в портфеле заказов; D — некоторый параметр (делитель).

Как видно из выражения, для определения количества точек разрыва и точек мутации необходимо знать значение делителя. Данный параметр был найден в результате исследований: $D = 10$.

Далее представлены результаты исследований системы при различном числе блоков в портфеле заказов, соответственно 20, 50 и 100 блоков.

Первые две гистограммы иллюстрируют работу «старого» алгоритма, причем рассматривается не только изменение ЦФ по поколениям, но и число срочных блоков в карте (рис. 8.12 и 8.13). Следующих две гистограммы отражают работу модернизированного алгоритма (рис. 8.14 и 8.15).



Рис. 8.16. Зависимость максимальной ЦФ от значения делителя

На рисунке 8.16 приведен график зависимости значения максимальной ЦФ от значения делителя в формуле для расчета количества точек разрыва и точек мутации. Как видно из графика, существует некоторое оптимальное значение данного параметра.

8.4. Планирование поставок на многопродуктовый склад

Планирование поставок товаров на склады торговых фирм является одной из основных задач организации материальных и информационных потоков в разветвленной сети поставщиков товаров и заказчиков. Преследуемая цель — обеспечение требуемого уровня обслуживания множества заказчиков, а также получения максимальной отдачи средств от вложенного капитала. Решение поставленной задачи усложняется стохастическим характером процесса в системе (изменение спроса, недопоставки товаров на склад и т. п.).

Практически во всех торговых объединениях встает вопрос о своевременном пополнении запасов, а именно, об определении моментов времени, когда товар нужно заказать у его производителей, с учетом времени доставки, и количества заказываемого товара. Завышенный уровень запасов приводит к большим издержкам, связанным с хранением запасов, с другой стороны заниженный уровень влечет издержки, связанные с неспособностью удовлетворить покупателя (снижение уровня оборота денежных средств, потеря покупателей в конкурентной борьбе), частые заказы приводят к издержкам, связанным с доставкой товара. По мнению экспертов [205], каждый процент сокращения уровня запасов может быть приравнен к десятипроцентному росту оборота денежных средств, а если бы удалось поставить под контроль хотя бы 75% колебания уровня инвестиций в товарно-материальные запасы, экономика развитых стран никогда бы не испытывала кризисных явлений.

В наиболее простом случае на складе хранится запас одного продукта (однопродуктовая система планирования поставок) либо нескольких независимых продуктов. Тогда изменение запаса на складе некоторого i -го товара можно представить, как показано на рисунке 8.17. Это — система управления запасами с постоянным объемом заказа. В существующей системе планирования она рассматривается как идеализированная.

В идеальном случае уровень запасов уменьшается с постоянной интенсивностью, равной среднесуточному сбыту S_{di} , а объем поставки равен объему заказа Q_i , т. е. нет недопоставки. Выдача заказа на поставку осуществляется в тот момент, когда текущее значение запаса J_i снизится до некоторого уровня P_i , который называется точкой заказа. То есть, условие выдачи заказа на поставку $J_i \leq P_i$. Время реализации поставки с момента выдачи заказа до поступления товара на склад представляет собой величину L . J_{0i} — уровень запаса товара на начало планового периода.

Для определения точки заказа P_i (критического уровня) в такой системе необходимо знать время доставки L и среднесуточный сбыт S_{di} . Однако

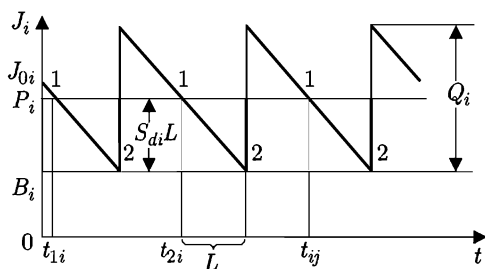


Рис. 8.17. Идеализированная система с постоянным объемом заказа: 1 — моменты подачи заказа; 2 — моменты получения заказа

этого недостаточно, так как фактический сбыт за время доставки заказа может превысить среднее значение и наступит временная нехватка товара (дефицит). Поэтому при определении точки заказа к ожидаемому сбыту за время доставки заказа добавляется резервный или страховой запас B_i .

Точку заказа в этом случае можно определить по формуле:

$$P_i = B_i + S_{di}L.$$

Моменты времени t_{ji} , в которые осуществляются заказы поставок, могут быть рассчитаны следующим образом:

$$t_{1i} = \frac{J_{0i} - P_i}{S_{di}}, \quad t_{ji} = t_{j-1i} + L + \frac{B_i + Q_i - P_i}{S_{di}},$$

где t_{1j} — момент времени, в который осуществляется первый заказ на поставку для данного наименования товара; t_{ji} — момент времени, в который осуществляется последующий заказ на поставку для данного наименования товара; j — номер заказа на поставку.

Для систем с постоянным объемом заказа величина Q_i рассчитывается, исходя из годовых издержек по формуле Уилсона [204]:

$$Q_i = \sqrt{\frac{2C_{0i} \cdot S_i}{C_{ui} \cdot k_i}},$$

где C_{0i} — издержки выполнения заказа на i -й товар; S_i — количество i -го товара, реализованного за год; C_{ui} — закупочная цена единицы i -го товара; k_i — издержки хранения i -го товара, выраженные как доля закупочной цены.

Данные расчеты были получены, исходя из предположения, что в рассматриваемой системе заказы на поставку для различных товаров независимы и поставка содержит продукцию одного наименования. При этом не учитываются то, что часть рассматриваемых характеристик системы имеет ярко выраженный случайный характер, например, L и S_i .

В случае многономенклатурного запаса на складе задача планирования поставок превращается в многопараметрическую оптимизационную задачу большой размерности. Сложность процессов в рассматриваемой системе

и их стохастичность делают проблематичным получение ее математического описания, адекватного реальности. Это приводит к необходимости использования имитационного моделирования и отказ от строгих математических оптимизационных методов.

Рассмотрим решение задачи планирования поставок с использованием гибридной системы принятия решений, структура которой аналогична представленной на рис. 5.1. Для ее построения применена интеллектуальная имитационная среда РДО (Ресурсы–Действия–Оперции) [134, 205]. Имитационная модель предназначена для генерации вариантов планов поставок производителем товаров на склад фирмы и оценки их эффективности, а блок оптимизации обеспечивает выбор оптимальных значений управляющих переменных, передаваемых в модель для составления эффективных планов поставок.

Критерием оптимизации являются суммарные потери на интервале планирования, рассчитываемые по формуле:

$$W = W_I + W_S + W_D,$$

где W_I — потери от хранения, W_S — потери от невозможности отгрузки товаров заказчикам из-за отсутствия товаров на складе, W_D — потери от оплаты поставок.

На складе фирмы хранятся товары N наименований, для каждого из которых определены специфические условия и сроки хранения. Товары на склад поставляются одним производителем, для этого он использует собственные транспортные средства. Фирма имеет четыре канала реализации товаров (четыре типа клиентов): крупные оптовики, магазины, дилеры (по регионам), представительства фирмы. Каналы различаются потребляемыми объемами товаров, их номенклатурой, а также частотой обращений на торговую фирму по поводу получения товаров. Обычно заказ от клиента содержит товары нескольких наименований, но независимость спроса на разные товары позволяет рассматривать этот заказ как несколько заказов, различающихся наименованиями товаров.

Данная фирма действует по принципу системы управления запасами с постоянным объемом поставок от производителя по каждому наименованию товара. Это означает, что после того, как фирма отгрузит все заказанные товары (которые могут быть отгружены) клиентам на настоящий день, производится анализ текущего состояния склада. Проверяется запас товара данного наименования. Если он снизился до определенного уровня, называемого критическим (или точкой заказа) P , то делается заявка производителю на поставку заданного объема товара данного наименования.

Так как фирма располагает многономенклатурным запасом, то система с постоянным объемом заказа модифицируется введением для каждого наименования товара предкритического уровня P_{ri} . Если запас товара какого-либо наименования достиг критического уровня, проверяются запасы остальных товаров на предмет достижения предкритического уровня. Если для какого-либо из товаров этот уровень достигнут, то он заказывается вместе с тем товаром, количество которого достигло критического уровня.

Эта модификация позволяет более эффективно использовать транспортные средства производителя.

Для расчета составляющих потерь используется имитационная модель работы фирмы по обслуживанию заказов и организации поставок товаров на склад. Управляющей информацией для принятия решений о заказе поставок в имитационной модели, как и в реальной системе, являются точки заказов (критические — P_i и предкритические — P_{ri} уровни запасов товаров, где i — номер товара). От выбора точек заказов зависят получаемые в результате моделирования составляющие потерь, и соответственно критерий оптимизации (определяемый на основе имитации): $W = F(P_i, P_{ri})$, $i = \overline{1, N}$, где N — количество наименований товаров. Заданы допустимые диапазоны варьирования точек заказов D_{Pi} — для критических и D_{Pri} — для предкритических уровней заказов. Необходимо найти такую комбинацию значений обоих уровней, чтобы значение критерия было минимальным:

$$\{P_i, P_{ri}\}, \quad i = \overline{1, N}, \quad P_i \in D_{Pi}, P_{ri} \in D_{Pri}: \quad W \rightarrow \min.$$

Объектом моделирования является работа торговой фирмы по управлению поставками, запасами и обслуживанием заказов клиентов. Исходными данными для системы планирования служат статистические данные об объемах спроса, о заявках и поставках, хранящиеся в информационной базе фирмы.

Процессы в рассматриваемой системе описываются в терминах РДО-метода с помощью модифицированных производственных правил. В имитационной модели описаны следующие возможные действия, протекающие на фирме:

- поступление заказов от клиентов в систему — генерируются на основе статистической информации;
- принятие решений об отгрузке товаров клиентам. Данные действия моделируют отгрузку товаров по заказам, срок отгрузки которых меньше или равен текущему дню (если на складе достаточно товара данного наименования, при этом в первую очередь обслуживаются заказы с наименьшим сроком отгрузки, что обусловлено необходимостью снижения потерь от невозможности отгрузки товара в срок);
- принятие решений о заявках на поставку товаров: если количество товара какого-либо наименования на складе снизилось до (или ниже) критического уровня и для этого товара нет заказанных, но не полученных поставок, то создается заявка на поставку данного товара, время прихода поставки задается ожидаемым днем прихода, количество, заказываемое по данному наименованию товара величина постоянная (система с постоянным объемом заказа). Затем проверяются запасы товаров всех остальных наименований на предмет достижения предкритического уровня, и если такие есть, то создаются заявки, параметры которых задаются по тому же принципу, что и для товара, уровень запаса которого снизился до критического уровня;

- приход поставок от производителя товаров. Если в системе есть поставки, у которых назначенный день прихода равен текущему дню, то вычисляется значение количества доставленного товара данного наименования с учетом статистической информации о недопоставках. Далее товар поступает на склад фирмы.

Моделирование осуществляется на протяжении квартала.

Для использования ГА необходимо кодирование значений точек заказов в двоичную форму. Способ кодирования представлен на рис. 8.18. Особь представляет собой битовую строку-хромосому длиной 350 бит. Гены в этой строке имеют длину по 7 бит и представляют собой закодированные значения точек заказов. Выбор длины гена равной 7 бит обусловлен тем, что гены в РДО представляются в виде целочисленных параметров типа ресурсов Особи. Целое число в РДО представляется в виде двух байтов. Из них один бит — знаковый. Из оставшихся пятнадцати бит четырнадцать используются для представления точек заказов.

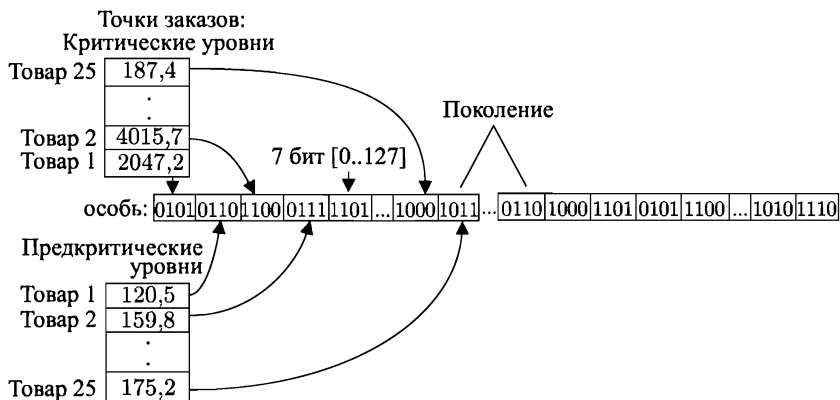


Рис. 8.18. Кодирование точек заказов

Таким образом, в одном параметре типа ресурсов Особи закодированы два гена — две точки заказов: критический и предкритический уровни для товара одного наименования. Для представления точек заказов в случае двадцати пяти наименований товаров используются 25 подобных параметров ресурса.

Так как семью битами может быть представлено число от 0 до 127, необходим пересчет диапазонов критических (D_{Pi}) и предкритических уровней (D_{Pr_i}) в диапазон D от 0 до 127. Этот пересчет осуществляется по формулам:

$$G_{2i-1} = \frac{P_i}{D_{Pi}} \cdot 127, \quad G_{2i} = \frac{P_{r_i}}{D_{Pr_i}} \cdot 127,$$

где i — номер товара, G_{2i-1} и G_{2i} — представления в виде десятичных чисел, закодированных в семи битах значений точек заказов.

Имея эти формулы, для любой особи возможен обратный пересчет из генов особи в критические и предкритические уровни запасов для каждого наименования товаров.

Было выяснено, что в суммарных потерях всегда присутствует такая составляющая, как потери от хранения. Изменение этой составляющей для различных комбинаций точек заказов невелико из-за невысоких значений годовых затрат на хранение. Поэтому в качестве функции пригодности особей была выбрана показательная функция, которая быстро растет с ростом показателя степени:

$$H = 2^{\frac{W_{\max} - W}{M} \cdot 50}, \quad (8.1)$$

где H — функция пригодности, $W_{\max}$ — максимально возможное значение потерь, которое выбрано на основании результатов моделирования с превышением полученной максимальной величины суммарных потерь на два порядка. Если для какой-либо особи значение суммарных потерь превысит $W_{\max}$, у этой особи будет очень малое значение функции пригодности.

Основными параметрами генетического алгоритма являются: количество особей в поколении; число поколений; вероятность скрещивания; вероятность мутации. Значения этих параметров были взяты из результатов исследований, выполненных на имитационной модели.

Исходная популяция генерируется случайным образом, при этом создаются особи со значениями генов от 0 до 127. Число генерируемых особей равно размеру поколения.

Товары на склад поставляются одним производителем. Транспортировка товаров от производителя на склад фирмы осуществляется транспортными средствами производителя. Имеются четыре канала реализации товаров, отличающихся объемами сбыта и частотой обращений на склад: для крупных оптовиков; для магазинов; для дилеров (по регионам); для представительств фирм.

Заказ от клиента содержит товары нескольких наименований, но независимость спроса на разные наименования товаров позволяет рассматривать заказ как несколько заказов, различающихся наименованиями товаров.

Расчет функции пригодности ведется имитацией работы фирмы в течение квартала, т. е. для каждой особи осуществляется прогон, по окончании которого рассчитывается функция пригодности по формуле (8.1). В ходе прогона осуществляется принятие решений о заявках на поставки. При этом для определения критического и предкритического уровней для каждого наименования товара производится расшифровка особи и полученные значения точек заказов используются при принятии решений.

Результатом решения оптимизационной задачи является лучшая особь по всем поколениям. Значения точек заказов, которые будут использоваться в торговой фирме при принятии решений о заявках на поставки в процессе работы в квартале, осуществляются путем расшифровки лучшей особи.

Эксперименты проводились на различных по напряженности (среднесуточный спрос по каждому наименованию товара) портфелях заказов от клиентов. Так как из-за большого числа характеристик и их комбинаций трудно привести интегральную характеристику, однозначно характеризующую данный портфель, то были проведены эксперименты для трех вари-

антов портфелей. Эти портфели различались средним количеством товара в заказе по каждому наименованию товара и каналу, а также интервалами времени между приходами заказов по каждому из каналов.

Для сравнения была смоделирована работа фирмы, когда значения точек заказа назначались эвристическим путем. Они были выбраны следующим образом:

- критические уровни брались в среднем с двукратным превышением величины среднесуточного спроса, умноженной на время доставки;
- предкритические уровни брались, исходя из вероятности прихода заказа на товар данного наименования;
- диапазоны варьирования точек заказов были выбраны следующим образом:
 - для критических уровней: в среднем с пятикратным превышением среднесуточного спроса, умноженным на время доставки;
 - для предкритических уровней диапазоны были выбраны одинаковыми и равными максимальному значению для случая выбора их экспертом — 400%.

Оценка суммарных потерь может быть получена на основе моделирования работы фирмы в квартале для лучшей по всем поколениям особи.

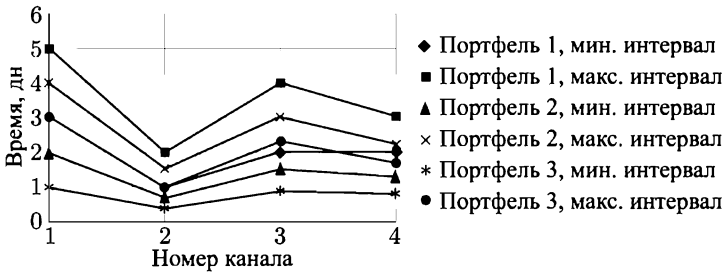


Рис. 8.19. Интервалы между приходами заказов

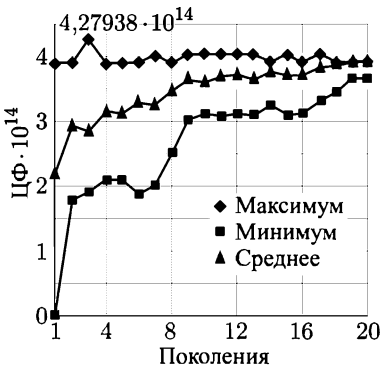


Рис. 8.20. Значения целевой функции для портфеля № 1

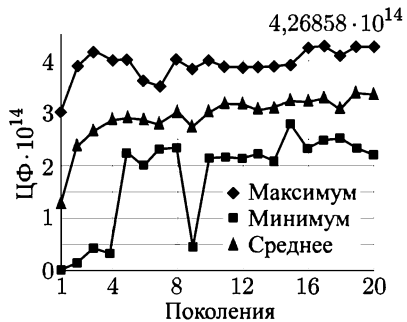


Рис. 8.21. Значения целевой функции для портфеля № 2

Оптимизация с помощью ПГА проводилась для 20 особей в поколении. 20 поколений, вероятности скрещивания — 0,7 и вероятности мутации — 0,06.

Моделировались три портфеля заказов, отличающихся интенсивностью потоков заказов различных типов (рис. 8.19).

Результаты экспериментов с использованием простейшего генетического алгоритма представлены на рис. 8.20–8.22. Здесь приведено изменение целевой функции по поколениям. Сравнение суммарных потерь при использовании эвристики и простейшего генетического алгоритма для оптимизации выбора точек заказа представлены в табл. 8.1.

По результатам экспериментов можно отметить, что рост среднего значения функции пригодности по популяциям (поколениям) демонстрирует работоспособность алгоритма, а максимальное значение целевой функции в пересчете на критерий W дает устойчивое (в среднем около 60%) снижение потерь по сравнению со случаем назначения точек заказа на основе среднего спроса за время доставки.

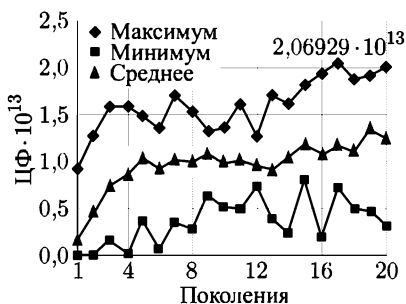


Рис. 8.22. Значения целевой функции для портфеля № 3

Таблица 8.1

Номер портфеля	Потери при использовании эвристик	Потери при использовании ПГА	Эффективность ПГА (%)
1	5806724	2794428	51.8
2	8907316	2798501	68.6
3	38081229	11531600	69.7

8.5. Планирование работы производственного участка

Задачи составления производственных планов являются комбинаторными задачами большой размерности. От их решения зависит эффективность производства. Однако из-за их сложности (задачи реальной размерности относятся к NP-полным) чаще всего задачи планирования решаются интуитивно человеком с использованием различных эвристик. В настоящем разделе представлены результаты применения гибридной системы (рис. 5.1) для составления планов работы производственного участка. В гибридной системе поддержки принятия решений использован ПГА, который осуществляет оптимизацию применения некоторого множества эвристических правил с целью комплексного учета различных факторов производства на качество составленного плана работы.

Рассмотрим задачу оперативного планирования работы участка по обработке древесно-стружечных плит в условиях динамически изменяющегося портфеля заказов. Рассматриваемый участок является частью мебельного комбината, который выпускает мебельные стенки, состоящие из элементов типа шкаф, тумбочка и других. Производство носит позаказный характер — работа ориентирована на текущий пакет заказов клиентов. Каждый заказ включает в себя набор стандартных изделий. Поэтому главная цель работы участка — это своевременное выполнение заказов. Следовательно, целью планирования являются правильно назначаемые, согласованные с заказчиками и соблюдаемые сроки выполнения заказов [206].

Участок реализует завершающий этап в технологическом процессе изготовления деталей, предшествующий сборке. Поступающие на него заготовки представляют собой древесно-стружечные плиты, имеющие определенные размеры, с необработанными кромками. На участке их обрезают до требуемых размеров, обклеивают по периметру шпоном и проводят специальную механическую обработку поверхностей. Заготовки поступают на вход участка отдельными партиями. Партия представляет собой штабель одинаковых заготовок, размер которых может изменяться в пределах от 30 до 60 штук. Полученные на участке детали такими же партиями передаются для окончательной сборки из них изделий.

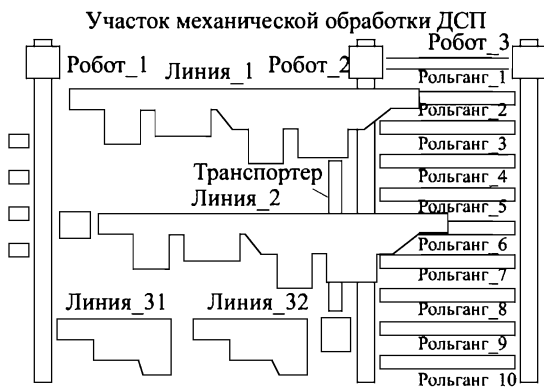


Рис. 8.23. Структура участка

Структура размещения оборудования на участке представлена на рисунке 8.23.

С входных рольгангов очередная выбранная для обработки партия заготовок транспортируется роботом_1 на вход линии_1 или линии_2 в зависимости от реализуемого техпроцесса. Данные линии являются кромкообразующими, они полностью взаимозаменяемы и работают в параллельном режиме. На них заготовки обрабатываются до требуемых размеров, а также обклеиваются их кромки. Заготовки на данных линиях проходят только одностороннюю обработку (обрабатывается только правая и передняя по движению кромки). После этого заготовки транспортируются с линии_1 транспортером, а с линии_2 роботом_2 на приемный накопи-

тель линий _31 и линии _32. Это линии фирмы «Нотат» для формирования кромок, которые полностью взаимозаменяемы. На них производится специальная обработка кромок (недоступная для линии _1 и линии _2), а также обработка поверхностей заготовок. В зависимости от типа заготовки она может обрабатываться как на одной из этих линий, так и на обеих. Линия _31 и линия _32 могут работать параллельно или последовательно. С них заготовки транспортируются роботом _1 либо на линию _1, либо на линию _2, в зависимости от того, на какой из этих линий транспортируемая партия заготовок обрабатывалась вначале. После повторной обработки на линии _1 или на линии _2 теперь уже партии готовых деталей попадают на выходные рольганги, где они ожидают момента комплектации изделия, чтобы при помощи робота _3 быть доставленными на выход участка.

Различные детали имеют различные технологические маршруты обработки (последовательность прохождения оборудования участка), что также усложняет процесс планирования. Один из возможных вариантов маршрутов приведен на рисунке 8.24, на котором крестами показаны места выполнения операций обработки.

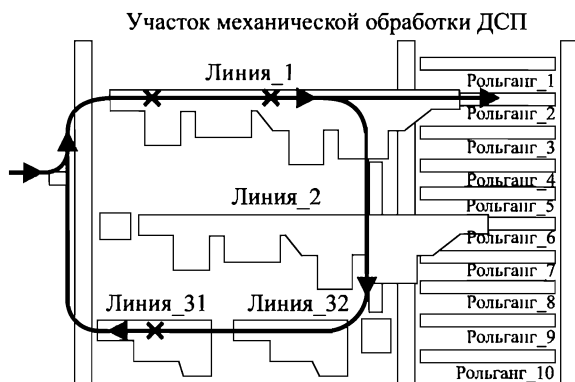


Рис. 8.24. Пример маршрута обработки

Окончание обработки партии заготовок фиксируется с прибытием на выходной штабелеукладчик последней обработанной детали. Переналадка линии под другую партию деталей производится только после окончания обработки предыдущей партии. Технологический процесс детали может разветвляться с целью получения различных модификаций исходной детали.

Рассмотренные условия работы участка позволяют выявить факторы производства, которые влияют на качество плана и которые должны учитываться при его составлении. Для учета отдельного фактора при планировании может быть сформулировано некоторое эвристическое правило. Однако трудность состоит в том, что необходимо сразу использовать набор правил, чтобы решение удовлетворяло сразу многим требованиям (часто противоречивым), для этого и предлагается использовать гибридную систему поддержки принятия решений, включающую в себя имитационную модель и ПГА (рис. 5.1).

Следовательно, система планирования должна демонстрировать правильное использование эвристик для принятия решения на каждом шаге поиска и выбора следующего шага решения в зависимости от того, какой получен результат в текущей ситуации. Метод решения такого класса задач должен предусматривать возможность выбора различных функций предпочтения на любом шаге поиска решения.

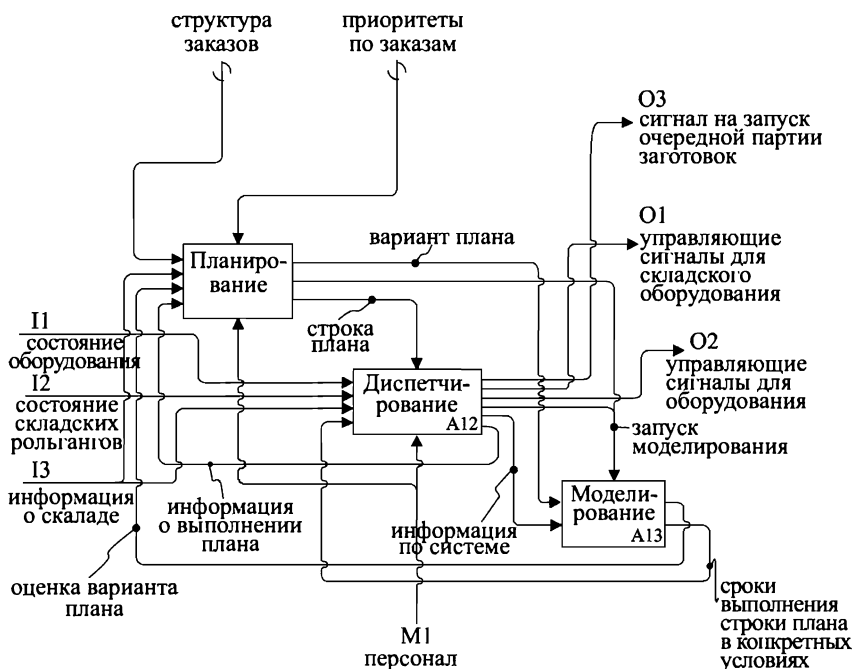


Рис. 8.25. Функциональная структура системы планирования

Функции системы планирования представлены на рисунке 8.25.

Исходными данными для системы управления участка является множество деталей, требующих изготовления на участке.

Для планирования используется имитационное моделирование в сочетании с эвристическими приоритетными правилами. Каждая деталь имеет свой приоритет, с учетом которого система управления составляет план работы участка. То есть, Приоритет — это числовая характеристика детали (работы), которая используется при выборе детали из всех возможных. Деталь с наибольшим приоритетом включается в план перед деталями с меньшими значениями приоритетов. Поэтому от правильного выбора правил назначения приоритетов зависит качество составляемого плана.

Приоритетные правила делятся на простые и составные. Простые правила основываются на какой-либо одной характеристике детали. Для условий рассматриваемого производства было принято решение использовать следующие приоритетные правила.

Статические, т. е. не меняющиеся в процессе использования:

- правило плановых сроков DDATE (Due date) — приоритет принимает значение, пропорциональное сроку, к которому должна быть изготовлена деталь. Он нацелен на запуск вперед тех деталей, которые раньше должны быть изготовлены;
- правило LPT (Longest processing-time) — приоритет назначаемый детали пропорционален длительности ее обработки;
- правило поэтапных плановых сроков OPNDD (Operation due date) — приоритет назначается в соответствии со сроками выполнения промежуточных операций над деталью;
- правило кратчайшей операции SPT (Shortest processing-time) — правило, обратное правилу LPT. Его цель — уменьшить межоперационное пролеживание деталей.

Динамические, изменяющиеся во времени:

- правило по декору DEK — детали имеющие декор, такой же как предыдущая включенная в план, имеют больший приоритет, чем остальные;
- правило по стандартным деталям DET — стандартные детали запускаются в производство раньше уникальных, при этом преследуется цель уменьшения переналадки оборудования;
- правило оставшихся этапов. Первой запускается деталь с наименьшим числом невыполненных операций обработки;
- правило временного резерва на этап S/OPN (Slack per operation);
- правило случайного выбора RND (Random) — детали выбираются равновероятно.

Данные приоритетные правила были выбраны исходя из того, что они ориентированы на временные характеристики заказов и на их плановые сроки.

Формируемый план должен обеспечивать своевременное выполнение заказов. В качестве критерия T решения задачи оперативного планирования будем использовать суммарное отклонение времени выполнения заказов от плановых сроков. Критерий представляет собой взвешенную сумму абсолютных значений отклонений как в положительную, так и в отрицательную стороны:

$$T = \sum_{i=1}^I a_i * |t_i| + \sum_{j=1}^J b_j * |t_j|,$$

где t_i и t_j — соответственно опережение и запаздывание заказов во времени [мин.]; a_i, b_j — весовые коэффициенты такие, что $a_i \ll b_j, i = 1, 2, \dots, I; j = 1, 2, \dots, J; n = I + J$ — общее количество заказов в пакете.

Для исследования эффективности приоритетных правил проводилась имитация работы участка с различными значениями исходных данных. Для каждого прогона генерировались различные по составу и по плановым срокам заказы, т. е. прогоны различались по величинам средних времен обработки деталей. Полученные результаты для рассматриваемых правил, не позволили выбрать из них для дальнейшего использования какое-либо одно, превосходящее все остальные по выбранному критерию в изменяю-

щихся условиях производства. Все правила показали нестабильную работу при изменении напряженности плановых заданий, а также не обеспечивали необходимую минимизацию установленного критерия.

На основе проведенных исследований эффективности простых приоритетных правил был выбран вид составного правила как взвешенной суммы приоритетов, получаемых с использованием простых правил:

$$prior_i = \alpha_1 U_{1i} + \alpha_2 U_{2i} + \dots + \alpha_{n-1} U_{(n-1)i} + \alpha_n U_{ni},$$

где α_i — весовой коэффициент, учитывающий важность j -го приоритетного правила; U_{ji} — значение приоритета i -й детали, полученное при использовании j -го приоритетного правила.

Весовые коэффициенты имеют значения, пропорциональные важности соответствующего параметра. В описанных условиях производства система оперативного планирования должна быть адаптивной, иметь возможность подстраиваться под ситуацию на участке и к изменяющемуся портфелю заказов. Введение в составное приоритетное правило различных элементарных приоритетных правил позволяет комплексно учесть те положительные свойства расписания, которые обеспечивают эти правила по отдельности. Варьируя степень значимости отдельного приоритетного правила в составном можно получать требуемые свойства плана. Для получения оптимальных значений весовых коэффициентов применен ПГА.

Одна из главных проблем данного подхода — получение адекватной оценки для принимаемых решений, т. е. прогнозирование их последствий в будущем. Для этой цели используется имитационная модель, реализованная на языке РДО. Используемый для определения оптимальных значений весовых коэффициентов ПГА также реализован на РДО (см. п. 5.2).

ПГА работает с абстрактной информацией, что является одним из его главных достоинств. Однако для этого требуется кодирование информации, с которой должен работать алгоритм. Так как состав приоритетных правил уже определен, то для варьирования остаются весовые коэффициенты. Для ПГА была принята следующая кодировка исходных данных:

- один ген хранит в себе весовой коэффициент одного из приоритетных правил;
- хромосома включает в себя все 8 правил, каждому из которых соответствует определенный ген;
- позиция гена-правила в хромосоме остается неизменной.

Составленная таким образом особь-хромосома представляется в двоичном формате для участия в работе ПГА. Каждый ген состоит из одного байта, т. е. в нем может храниться число от 0 до 255. Так как для кодировки вещественного числа необходимо использовать дополнительные разряды, то изначально весовой коэффициент представлен в гене в виде целого числа в интервале $[0..255]$, которое потом масштабируется на необходимый нам интервал $[0,0..1,0]$ путем деления содержимого гена на 255,0.

На рисунке 8.26 приведены результаты одного из прогонов ПГА. Здесь представлены изменения критерия оптимизации T для худшей, лучшей особей в поколениях, а так же среднее значение критерия по поколениям.

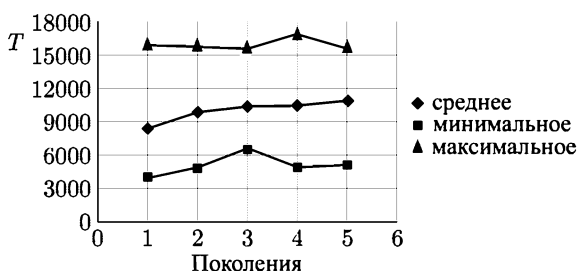


Рис. 8.26. Изменение критерия по поколениям

Кроме кодировки исходной информации необходимо также правильно определить вид целевой функции. ЦФ определяется для каждой отдельно взятой особи. В представленной задаче ЦФ должна максимизироваться — это обусловлено особенностями работы генетических операторов. Так как критерий решения нашей задачи T необходимо минимизировать, то ЦФ принимается следующего вида:

$$\text{ЦФ} = 1800 - T,$$

где 18000 — максимально возможное значение критерия T , которое определяется по интервалу генерации плановых сроков заказов.

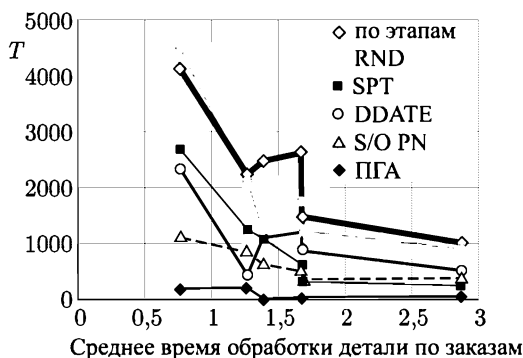


Рис. 8.27. Сравнение эффективности приоритетных правил

На рисунке 8.27 представлены сравнительные результаты работы отдельных приоритетных правил и составного правила. Из графиков видно явное преимущество комбинированного приоритетного правила над используемыми простыми. Таким образом, применение простого генетического алгоритма обеспечивает рациональное сочетание положительных свойств, обеспечиваемых расписанию отдельными приоритетными правилами, что позволяет осуществлять эффективное планирование в позаказной системе производства.

8.6. Динамический оптимальный раскрой материала

Улучшение использования материала на основе применения оптимизационных алгоритмов при разделке бревен имеет важное значение для предприятий лесоперерабатывающей промышленности. Рассмотрим особенности решения этой задачи применительно к типовому участку разделки бревен фирмы Heidelberg Holzverarbeitung GmbH, ФРГ (рис. 8.28) [207].

На участке каждый ствол дерева пилят на пилоблоки для дальнейшего изготовления из них балок и брусьев (далее изделий). Пилоблок представляет собой отрезок бревна, который характеризуется длиной, большим и малым диаметрами. В него укладывают (в зависимости от меньшего диаметра и длины) несколько изделий, как это показано на рисунке 8.29, причем эти изделия могут принадлежать различным заказам. В каждое бревно могут быть вписаны, в общем случае, различные пилоблоки, при этом варианты вписывания отличаются по эффективности использования материала. Исходной информацией для работы участка служат заказы потребителей, образующие портфель заказов. Заказ содержит данные о заказчике, дату получения заказа и плановую дату изготовления, число единиц в заказе, размеры балки или бруса в позиции заказа (ширину, высоту и длину), объем, тип дерева и другую информацию.

Оборудование участка (см. рис. 8.28) включает в себя конвейеры, транспортеры, позицию измерения размеров бревен, торцовочной пилы, накопителей и пульта диспетчера. Приемные транспортеры 1 и 2 служат для приема партий бревен и выдачи их по одному на конвейеры. Распилка бревна на пилоблоки осуществляется на позиции распила. Имеется позиция измерения, расположенная на конвейере 2. При прохождении бревна через систему ультразвуковых датчиков проводится замер диаметра бревна через каждые 5 см длины. Профиль бревна высвечивается на ЭВМ оператора, который, исходя из портфеля заказов, решает, как его распилить. Для этого он анализирует портфель заказов и, используя свой опыт (эвристики), ставит на экране ЭВМ метки для распила.

Транспортер 3 служит для передачи пилоблоков с конвейера 3 на конвейер 4. Вдоль конвейера 4 расположены 28 накопителей (имеющие номера с 1 по 28) для пилоблоков различной длины. Накопители 29 и 30 служат для остатков от бревен после распила. Таким образом, на вход участка поступают партии бревен, которые затем пилятся по одному в произвольном порядке, а на выходе накапливаются пилоблоки.

Бревно условно представляется как усеченный конус, размеры которого полностью известны после прохождения бревна через позицию измерения (см. рис. 8.28). В сечении бревно круглое или эллиптическое. Искривление бревен в данной постановке не учитывается, что отвечает, по статистическим данным фирмы, приблизительно 90% случаев в реальном производстве. Длина бревна может достигать 24 м.

В каждый момент времени t в системе имеется множество заказов, образующих портфель заказов ($N_z(t)$ — число заказов в момент времени t):

$$Z(t) = \{Z_i(t) / i = 1, 2, \dots, N_z(t)\}.$$

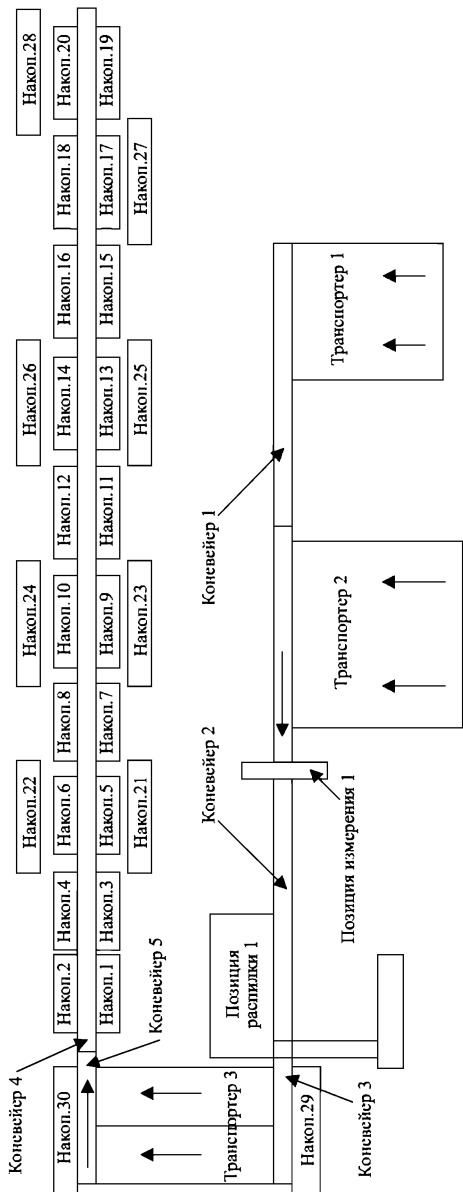


Рис. 8.28. План участка разделки бревен

Каждый i -й заказ из множества $Z_i(t) \in Z(t)$ содержит в общем случае несколько позиций:

$$Z_i = \{z_{ij}/j = 1, 2, \dots, n_i\},$$

где n_i — число позиций в заказе.

Каждая j -я позиция i -го заказа имеет следующие атрибуты:

$$z_{ij} = \langle H_{ij}, W_{ij}, L_{ij}, k_{ij}, \lambda_{ij}, t_i, T_i \rangle_i,$$

где k_{ij} — число изделий j -го вида в i -м заказе; t_i — время его поступления в систему; T_i — плановый срок выпуска заказа; λ_{ij} — дополнительные свойства: вид древесины, класс качества изделий, сорт древесины и др.; H_{ij}, W_{ij}, L_{ij} —соответственно высота, ширина и длина j -го изделия, они определяются существующим стандартом, например, DIN 4070 (табл. 8.2).

Таблица 8.2

Измерения	Сечения бруса			Сечения балока	
Ширина/толщина, мм	60/60	100/100	140/140	100/200	180/220
	60/80	100/120	140/160	100/220	200/200
	60/120	120/120	160/160	120/200	200/240
	80/80	120/140	160/180	120/240	
	80/100	120/160		160/200	
	80/120				
Длина, мм	3000...6000 Шаг 250			3000...6000 Шаг 250	

В процессе проведения исследований разработана имитационная модель участка распиловки бревен с использованием интеллектуального языка

ка имитационного моделирования РДО [207].

Специфика данной задачи требует рассмотрения большого числа возможных компоновок заказов в распиливаемом бревне. Учитывая, что время на обработку данных и принятие решения ограничено малым временем прохождения распиливаемого бревна от точки измерения до пилы, получаем оптимизационную задачу с ограничением на время принятия решения. Решения желательно принимать не дольше чем, за 10–15 с. В противном случае возникающие простои существенно снижают эффективность работы участка.

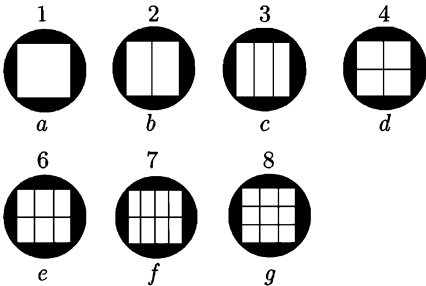


Рис. 8.29. Варианты вписывания брусев и балока в сечение бревна

При большом объеме портфеля заказов оператор не в состоянии проанализировать все или хотя бы достаточное количество вариантов решений и выбрать оптимальный. Задача усложняется тем, что портфель заказов все время пополняется новыми заказами в реальном масштабе времени, что делает работу оператора еще более трудной. Поэтому требуется автоматизировать принятие решения при разделке бревна применительно к рассмотренным условиям производства, освободив от нее оператора и оставив ему функции активного контроля процесса раскроя.

Рассмотрим использование ПГА для этой цели. В нем используем три генетических оператора: воспроизведения, односточечного кроссинговера и мутации. Для его использования необходимо выбрать способ кодирования информации в двоичную форму. Удачный способ кодирования может значительно увеличить производительность и точность получаемых результатов, тогда как неудачный может привести к возникновению тупиковых ситуаций.

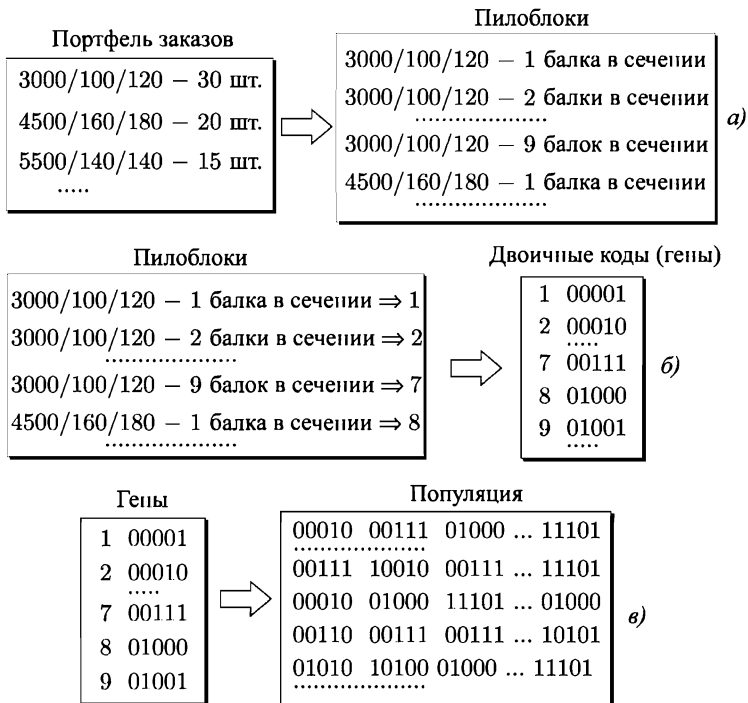


Рис. 8.30. Формирование (а) пилоблоков, кодирование (б) пилоблоков, формирование популяции (в)

Предложенный способ кодирования представлен на рис. 8.30 и заключается в следующем. Сначала, исходя из текущего состояния портфеля заказов и стандарта по длине и форме сечения (рис. 8.29), генерируем множество всех возможных способов укладки изделий в пилоблоки. При этом не учитываем возможность их вписывания в бревно (см. рис. 8.30. а). Далее

каждому пилоблоку присваиваем двоичный порядковый номер, соответствующий гену в генетическом алгоритме (рис. 8.30, б). Из полученных генов случайным образом составляется особь (в понятиях задачи особи соответствует бревно). Так как бревно не может превышать по длине 24 м, а пилоблок (согласно ГОСТу) не превышает 3 м, то особь состоит из восьми генов. Из получаемых вариантов особей формируется популяция (рис. 8.30, в).

При переводе номеров пилоблоков в двоичный формат и генерации из них особей популяции в подавляющем большинстве случаев получаем число, не кратное формату особи. В результате имеются лишние комбинации, которые в действительности не имеют смысла. Этот недостаток трудно избежать, так как он обусловлен спецификой решаемой задачи. Поэтому все эти комбинации попадают в разряд «запрещенных» и из-за них вносятся некоторые изменения в структуру генетического алгоритма.

При формировании популяции необходимо определить ее оптимальный размер. Так как он сильно коррелирует с другими параметрами генетического алгоритма, то его значение было определено после нескольких экспериментальных прогонов алгоритма для единичного бревна. Эти эксперименты осуществляли на имитационной модели участка. Кодирование проводим при появлении на участке каждого нового бревна. Бревна генерировались датчиками случайных чисел последовательно с параметрами, вычисляемыми по эмпирическим зависимостям.

В качестве оптимизируемой величины (функции пригодности) был применен коэффициент использования материала (КИМ) по объему. Как ограничение использовали КИМ по длине, что не помешало ему иметь высокие значения в процессе решения.

КИМ по объему U_v^t определяется как суммарный объем пилоблоков, деленный на объем бревна:

$$U_v^t = \frac{\sum_{k=1}^K V_{pk}}{V_t}.$$

КИМ бревна по длине U_l^t равен суммарной длине пилоблоков, деленной на длину бревна, из которого они выпилены. Этот критерий можно использовать, если есть заказы на изделия большой длины:

$$U_l^t = \frac{\sum_{k=1}^K L_{pk}}{L_t},$$

где L_{pk}, V_{pk} — длина и объем k -го пилоблока соответственно; L_t, V_t — длина и объем бревна соответственно.

С организованной указанным образом популяцией работает простейший генетический алгоритм, использующий, как указывалось выше, три оператора. Этот алгоритм также как и имитационная модель был запрограммирован в среде РДО. Рассмотрим операторы простого генетического алгоритма.

Воспроизведение. Каждой особи в популяции присваивается соответствующее по величине значение функции пригодности. В качестве таковой используется величина КИМ, которую необходимо оптимизировать. Цель воспроизведения состоит в том, чтобы отобрать для дальнейшей работы «сильнейшие» особи. Реализуется это с помощью простейшей рулетки (розыгрыша). Чем больше функция пригодности, тем больший сектор на рулетке получает данная особь, т. е. тем большая вероятность для данной особи воспроизвести в новой популяции свою копию. Рулетка раскручивается столько раз, сколько особей в популяции, после чего получается очередная новая популяция.

Скрещивание. К полученной после воспроизведения популяции применяется оператор скрещивания. Он случайным образом выбирает из популяции две особи. Затем случайным же образом выбирается порядковый номер бита в особи, по которому она будет рассекаться. После чего особи обмениваются отсеченными концами. Как результат получаем новую популяцию.

Мутация. Данный оператор сдерживает распространение особо «сильных» особей в популяции на первых поколениях с целью гарантированного непопадения в локальный оптимум. Для этого оператор мутации инвертирует определенное число битов в популяции на противоположные. Применение генетических операторов для поставленной задачи раскроя имеет некоторые особенности. При воспроизведении розыгрыш воспроизводимых особей реализуется с помощью случайного выбора на суммарном поле значений функции пригодности. Мутация представляется отношением числа измеренных битов к общему числу битов в популяции. Это соотношение при разработке ПГА выбирали экспериментальным путем при реализации алгоритма на единичных бревнах.

Структурная схема ПГА представлена на рис. 8.31. Для учета физической реализации особи были введены так называемые «штрафы», которые уменьшали значение функции пригодности особи, если та не удовлетворяла ограничениям по длине или диаметру.

После апробации генетического алгоритма на одном бревне была набрана статистика, на основе которой определялись размер популяции, число поколений и число изменяющихся битов в одном поколении. Алгоритм работает до тех пор, пока не будут выполнены все заказы, входящие в портфель заказов. Получение портфеля заказов осуществлялось на имитационной модели с помощью генераторов случайных чисел. На имитационной модели генерировались параметры поступающих на распиловку бревен. Внутренний цикл алгоритма реализует генетический алгоритм в том виде, как это было описано выше. После выбора оптимального варианта раскроя очередного бревна из портфеля заказов удаляются выполненные заказы (вписанные в данное бревно) и оставшиеся заказы составляют портфель заказов, используемый для раскроя следующего бревна (внешний цикл алгоритма). Функция пригодности рассчитывалась, исходя из вписывания в усеченный конус последовательности цилиндров-пилоблоков.



Рис. 8.31. Структурная схема ПГА

С использованием алгоритма осуществлялась имитация распиловки на портфелях из 10, 20, 30, 50 и 100 заказов и он был модифицирован с учетом оптимальных значений параметров. Представленный на рис. 8.31 цикл поиска осуществляется до тех пор, пока не будет исчерпан весь портфель заказов. Время работы алгоритма для указанных портфелей заказов составило несколько секунд, что меньше требуемого.

На рис. 8.32 приведен вид интерфейса оператора, где представлен результат раскроя очередного бревна.

В рамках решения задачи было проведено сравнение простейшего генетического алгоритма с тремя эвристическими алгоритмами, предложен-

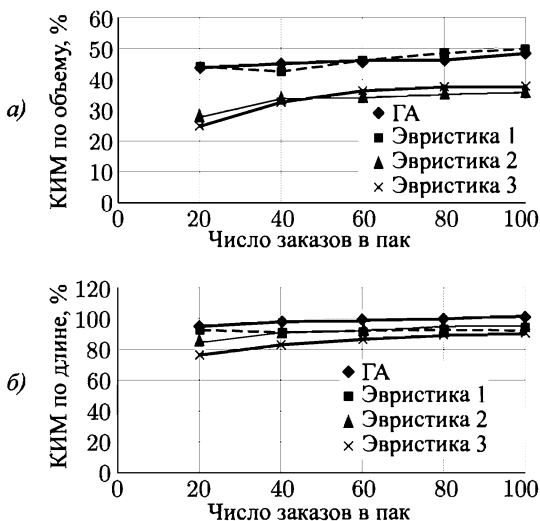


Рис. 8.33. Сравнительные характеристики алгоритмов раскря по использованию объема (а) и длины (б) бревна

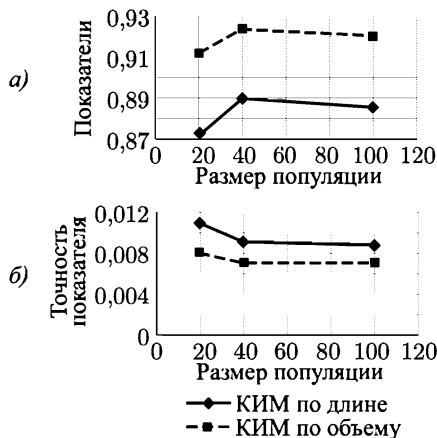


Рис. 8.34. Зависимости показателей работы ГА (а), их точности (б) от размера популяции

расчетов и лишь к незначительному улучшению качества раскря.

Влияние изменения числа заказов в системе на показатели ее работы представлено на графиках (рис. 8.35). Условия экспериментов были приняты следующими: число поколений = 10; $p_c = 0,6$; $p_m = 0,03$. Представлены средние значения показателей для бревна, их среднее квадратическое отклонение по раскря 100 бревен. Число заказов N изменялось от 5 до 50. Размер популяции принимал значения 20 и 40 особей.

Из графиков видно, что чем больше число заказов в системе, тем лучшее значение коэффициента использования материала может быть получено

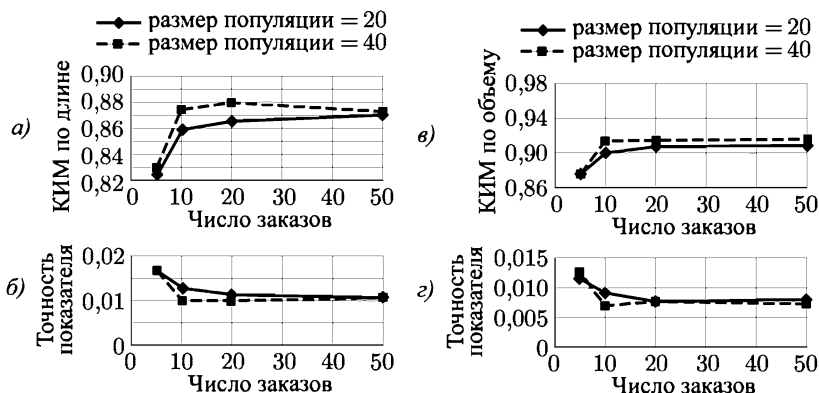


Рис. 8.35. а, б. Зависимости показателей работы системы по раскрою 100 бревен

Рис. 8.35. в, г. Зависимости показателей работы системы по раскрою 100 бревен

с использованием генетической оптимизации. Показатели использования материала по длине и по объему с увеличением числа заказов больше 10 улучшаются незначительно.

В целом можно сделать вывод, что ПГА осуществляет раскрой бревна за время, приемлемое для оперативного управления. В зависимости от используемого показателя качества должны выбираться различные параметры алгоритма. Чем больше заказов в системе, тем больше есть вариантов хороших решений и тем лучше работает алгоритм поиска. Алгоритм обеспечивает достаточно хорошую точность получаемых показателей.

8.7. Задача оптимальной раскладки грузов на поддоне

Рассмотрим применения ГА для нахождения оптимальной раскладки коробок одинаковых размеров на поддоне (паллете). Данная задача часто возникает в логистике, где осуществляется либо хранение, либо транспортировка грузов, упакованных в коробки и укладываемых на стандартные поддоны. При этом коробки могут быть произвольного размера, так как в различных странах используются различные стандарты, но на поддон укладываются лишь коробки одинакового размера. По существующим нормам коробки могут на 5 мм выступать за пределы поддона (рис. 8.36. а). Коробки на смежных уровнях должны лежать так, чтобы перекрывать друг друга и, тем самым, обеспечивать устойчивость штабеля (рис. 8.36. б). Система предназначена для нахождения оптимальной раскладки коробок на стандартных поддонах по уровням и может

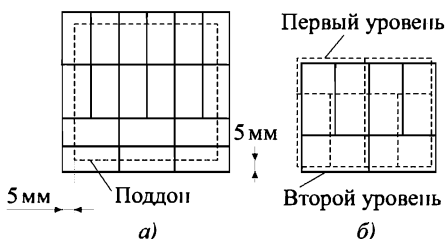


Рис. 8.36. Требования к раскладке: а) ограничения; б) перекрытие уровней

использоваться на складах с ручной погрузкой грузов в виде коробок на поддоны.

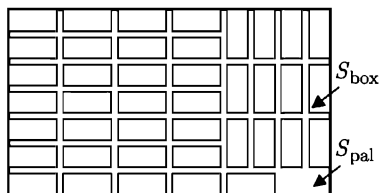
Оптимальным решением данной задачи следует считать решение, удовлетворяющее следующим условиям:

- 1) коэффициент заполнения поддона коробками максимально высокий;
- 2) варианты раскладок, найденные системой, должны обеспечивать возможность качественного штабелирования.

В качестве входной информации система учитывает:

- размеры коробки;
- размеры поддона;
- размер популяции;
- коэффициент новых особей по поколениям;
- вероятность мутации;
- размер выступа коробок за пределы границы поддона.

Выходными данными должно являться описание и отображение оптимального решения. Основным критерием оптимизации является коэффициент заполнения площади поддона (рис. 8.37):



$$K = \frac{S_{box} \times n_{box}}{S_{pal}} \rightarrow 1,$$

где S_{box} — площадь коробки; n_{box} — количество коробок на одном уровне поддона; S_{pal} — площадь поддона.

Рис. 8.37. Вид раскладки

Наименьшей неделимой единицей биологического вида, подверженной действию факторов эволюции, является особь a_k^t (индекс k обозначает номер особи, а индекс t — некоторый момент времени эволюционного процесса). В нашей задаче в качестве аналога особи a_k^t используется одно из возможных решений — раскладка.

Каждая особь имеет некоторые свойства:

- тело — описание особи. В нашем случае описание представлено в виде линейной бинарной строки, где 0-му значению бита соответствует горизонтальное расположение коробки, 1-му — вертикальное. Такое описание особи (раскладки) дает легкое применение операторов генетики и ускоряет работу программы в целом, что в конечном итоге приводит к лучшим результатам. Для преобразования такой линейной цепочки применяется специальный алгоритм укладки. Каждая следующая коробка укладывается на первое попавшееся место на поддоне по приоритету слева-направо, сверху-вниз. Так, раскладка, приведенная на рисунке 8.37, соответствует особи: 0 0 0 0 1 1 1 1 0 0 0 0 0 0 0 0 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0;
- коэффициент степени приспособленности отдельной особи в поколении. В нашем случае он определяется как взвешенная функция двух коэффициентов — коэффициента заполнения поддона K и коэффициента

ента заполнения условных линий поддона S со своим коэффициентом важности k_S (рис. 8.38):

$$F = \frac{S \times k_S + K}{1 + k_S}.$$

Коэффициент заполнения условных линий поддона S рассчитывается по формуле:

$$S = \frac{\sum a_i^2 \times ((P_x - d_i)/P_x)}{\sum a_i^2},$$

где P_x — ширина поддона; $a_i = \frac{I-i}{i}$, I — общее число линий, d_i — незаполненность по линии.

Для работы программы необходимо задать некоторые входные параметры, среди которых будут: размеры коробок (Bx, By), размеры поддона (Px, Py), а также некоторые коэффициенты. Первое поколение особей (раскладок) генерируется по случайному закону. В качестве условий окончания процесса эволюции популяции может использоваться либо окончание времени эволюции $t > T$, либо выполнение равенства $P^t = 0$, что означает, что получены все генотипы в хромосомном наборе популяции.

После того, как получена популяция, содержащая лучшие особи, следует использовать несколько лучших особей для формирования раскладки коробок на нескольких уровнях штабеля. Так можно, например, все четные уровни заполнять согласно лучшей особи, а все нечетные — согласно второй по качеству особи популяции (рис. 8.39).

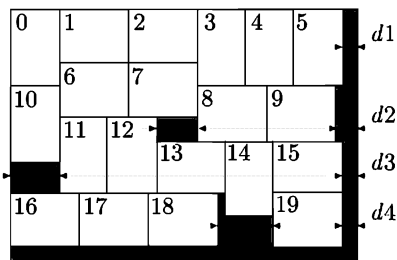


Рис. 8.38. Определение качества раскладки (особи)

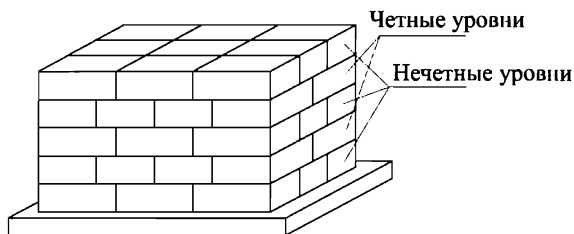


Рис. 8.39. Укладка штабеля на поддоне

Программа, реализующая генетический алгоритм, написана на языке программирования Borland C++ с использованием объектного подхода, т. е. все необходимые основные данные и процедуры реализованы как свойства и методы соответствующих классов. Такой подход позволяет просто и наглядно представить необходимые данные и описать методы работы

с ними, при этом легко реализуется идеология динамических объектов, позволяющая снизить требования на необходимую для работы программы оперативную память.

Исходя из текущей реализации генетического алгоритма, описанной выше, в программе используются 3 основных вида объектов: коробка, экземпляр, популяция. При этом объект популяция содержит в себе множество объектов типа экземпляр, каждый из которых в свою очередь содержит множество объектов типа коробка.

Методы объекта BOX (коробка) позволяют создавать описание коробки (K), после чего K в состоянии сама отвечать на основные вопросы, возникающие при укладке коробок и подсчете коэффициентов, такие как: Не занимает ли коробка места с координатами x , y и определенной ориентацией? Не расположена ли коробка на уровне y ? и т. п. При этом K «умеет» отображать себя на экране в заданных координатах (метод $\text{show}(x, y)$).

Методы объекта PERSON (особь) позволяют создать раскладку с некоторым описанием, т. е. проверить, как укладывается данное описание на поддон и вычислить соответственно коэффициент заполнения поддона K , количество переложённых коробок, подсчитать коэффициент заполнения поддона по уровням n , соответственно, найти функцию пригодности S для данного варианта укладки.

Методы объекта POPULATION (популяция) позволяют сгенерировать новую популяцию с заданными свойствами (свойства описаны глобальными переменными) и проводить на ней основные генетические операции (селекцию, скрещивание и мутацию), обеспечивая смену поколений. Существует также метод, позволяющий подсчитывать для текущего поколения максимальное, минимальное и среднее по популяции значение функции пригодности и определять лучшую особь.

На полученной модели оптимизации укладки коробок на поддоны были проведены эксперименты для нахождения оптимальных значений глобальных коэффициентов и проверки работоспособности модели.

Как показали исследования, качество получаемых результатов сильно зависит от значений глобальных параметров таких как:

1. MAX_POP — количество особей в популяции.
2. K_NEW — количество новых особей (число скрещиваний).
3. K_MUT — коэффициент мутации (часть изменяющейся длины особи).
4. LIFE — время «жизни» особи (максимальное число поколений, которое она может существовать).
5. K_NEAR — коэффициент близости особей.
6. k_S — коэффициент влияния на функцию жизнеспособности.

Значения этих параметров, обеспечивающие эффективное функционирование системы, были получены экспериментально и соответственно равны: $K\_NEW = 2$; $K\_MUT = 2$; $LIFE = 7$; $K\_NEAR = 10$; $k_S = 0,9$.

С этими значениями в дальнейшем и проводились эксперименты. При этом, оставляя неизменными размеры поддона, варьируя значение отношения размеров коробок, получаем три основных вида размеров коробок:

1. Размеры коробки не кратны друг другу (ширина не кратна длине).
2. Размеры коробки кратны друг другу (ширина кратна длине).
3. Размеры коробки кратны друг другу (ширина кратна длине) и они кратны какому-либо из размеров поддона.

Анализ полученных результатов показал, что ПГА ведет себя лучше (дает большие значения коэффициента заполнения поддона) в 1-м, самом общем, случае и незначительно хуже — во 2-м и 3-м случаях (по всей видимости это объясняется тем, что эвристический метод реализует именно поиск кратных сочетаний размеров для оптимизации).

Вид зависимостей коэффициента заполнения площади поддона от отношения ширины коробки к длине приведен на графиках (рис. 8.40). Здесь представлены результаты раскладки коробок различных размеров. Значения коэффициента больше единицы связаны с тем, что коробки могут выходить за границу поддона.

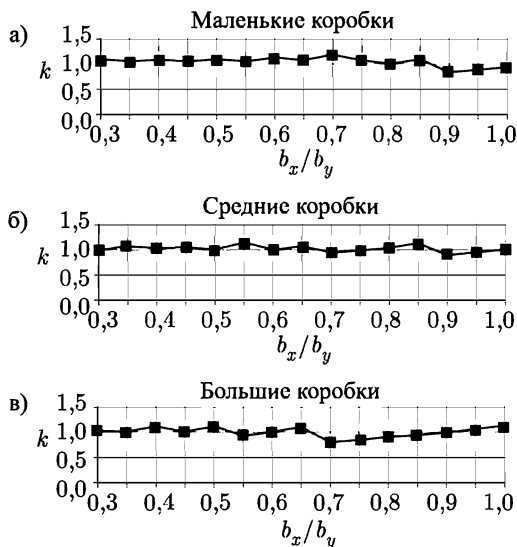


Рис. 8.40. Результаты работы системы

В результате можно сделать следующие выводы:

1. Генетические алгоритмы можно использовать для нахождения оптимальных укладок коробок на поддоны, но получаемые с их помощью варианты укладок сложны для выполнения вручную и предпочтительно использование автоматизированных систем.
2. Варианты укладок, полученные с помощью ПГА, легко поддаются штабелированию ввиду большой неоднородности вариантов.

Заключение

Наука — это попытка привести хаотическое
многообразие нашего чувственного опыта
в соответствие с некоторой единой
системой мышления.

А. Уинстетин

Подводя итоги, отметим, что существование эволюции — исторический факт. Доказательства существования эволюции относятся к различным областям биологии и генетики. Ч. Дарвин, К. Поппер, Н. Дубинин, И. Шмальгаузен и другие ученые понимали эволюцию как результат действия отбора на различные вариации и альтернативы, встречающиеся в природе. Они возникают на основе случайных процессов и являются разнообразными. Естественный процесс, анализирующий эти альтернативы, представляет собой сложный механизм, направляющий их по одной из многочисленных ветвей эволюционного прогресса.

Дж. Холланд, его ученики и другие ученые предложили использовать модели описанных в книге эволюций для решения задач искусственного интеллекта, оптимизации в технике, конструирования, проектирования, технологической подготовки производства и др. При этом полученные результаты показывают перспективность данного направления. Авторы выбрали ряд основных инженерно-практических задач и показали преимущество использования эволюционного моделирования для получения множества качественных результатов за приемлемое время.

Основные результаты монографии можно сформулировать следующим образом. Рассмотрены основные принципы эволюции в живых и искусственных системах. Проанализированы подходы к построению архитектур искусственных систем на основе различных моделей эволюций. Описаны основные проблемы синергетики. Рассмотрены состояния, проблемы, перспективы, способы построения и развития иерархических искусственных систем. Исследованы стратегии взаимодействия поисковых методов и эволюционного моделирования. Приведены нестандартные архитектуры решения инженерных задач, позволяющие получать набор квазиоптимальных решений за полиномиальное время. Сформулированы ряд положений и основные принципы теории эволюционного моделирования. Проанализирована теорема генетических алгоритмов, показывающая вероятность выживания лучших решений. Сформулирована постановка оптимизационных задач принятия решений на графах. Описаны генетические операторы, использующие фрактальные структуры, методы дихотомии, золотого сечения и чисел Фибоначчи. Рассмотрены подходы к решению основных инженерных задач. Они состоят в использовании комбинированных методов генетического поиска, моделей синтетической эволюции, агрегации фракталов, иерархического триединого подхода и позволяют получать набор

локально-оптимальных решений за полиномиальное время. Это дает возможность распараллеливать процесс оптимизации, эффективно управлять поиском, получать оптимальные и квазиоптимальные решения за время, сопоставимое с временем реализации практических алгоритмов.

Теория эволюционного моделирования — это незаконченное фундаментальное исследование. Она постоянно совершенствуется и дополняется. Отметим, что новые научные направления в эволюционном моделировании непрерывно развиваются и все более широко применяются в различных научных исследованиях. Авторы сделали первую попытку объединить некоторые эволюционные, гомеостатические и синергетические принципы для повышения качества управления генетическим поиском при решении инженерных задач. Мы надеемся, что внесли определенный вклад в теорию и практику эволюционного моделирования, сделав небольшой, но важный шаг в этом направлении. Необходима дальнейшая разработка фундаментальной теории эволюционного моделирования и широкое проведение экспериментальных исследований. Надеемся, что монография позволит интенсифицировать исследования в этой области.

СПИСОК ЛИТЕРАТУРЫ

1. Большая советская энциклопедия. Т. 29. — М.: Изд-во Сов. энцикл., 1978.
2. *Инге-Вечтомов С. Г.* Введение в молекулярную генетику. — М.: Высшая школа, 1982.
3. *Приходченко И. И., Шкурат Т. П.* Основы генетики человека. — Ростов-на-Дону: Феникс, 1997.
4. *Дарвин Ч.* Происхождение видов путем естественного отбора. Соч., т. 3. — М.–Л.: Академия, 1939.
5. *Шмальгаузен И. И.* Проблемы дарвинизма. — Л.: Наука, 1969.
6. *Ауэрбах Ш.* Генетика. — М.: Атомиздат, 1966.
7. *Дубинин Н. П.* Избранные труды. Т. 1. Проблемы гена и эволюции. — М.: Наука, 2000.
8. *Моисеев Н. Н.* Алгоритмы развития. — М.: Наука, 1987.
9. *Пригожин И., Стенгерс И.* Порядок из хаоса. — М.: Прогресс, 1986.
10. *Алиханян А. В. и др.* Общая генетика. — М.: Наука, 1985.
11. *Дульнев Г. Н.* Введение в синергетику. — С.-Пб.: Проспект, 1998.
12. *Майер Э.* Популяции, виды и эволюция / Пер. с англ. — М.: Мир, 1974.
13. *Фогель Л., Оуэнс А., Уолли М.* Искусственный интеллект и эволюционное моделирование. — М.: Мир, 1969.
14. *Эшби У. Р.* Введение в кибернетику. — М.: ИЛ, 1959.
15. *Винер Н.* Кибернетика или управление и связь в животном мире. — М.: Сов. радио, 1968.
16. *Курейчик В. В.* Эволюционные методы решения оптимизационных задач. — Таганрог: Изд-во ТРТУ, 1999.
17. *Курейчик В. В.* Эволюционные, синергетические и гомеостатические методы принятия решений. — Таганрог: Изд-во ТРТУ, 2001.
18. *Курейчик В. М.* Генетические алгоритмы. — Таганрог: Изд-во ТРТУ, 1998.
19. *Курейчик В. М.* Генетические алгоритмы и их применение. — Таганрог: Изд-во ТРТУ, 2002.
20. *Ламарк Ж. Б.* Философия зоологии. Т. 1, 2. — М.–Л.: Академия, 1939.
21. *Тимофеев-Ресовский Н. В., Воронцов Н. Н., Яблоков А. В.* Краткий очерк теории эволюции. — М.: Наука, 1977.
22. *Аристотель.* Сочинения в четырех томах. Т. 1. — М.: Мысль, 1976.

23. Practical Handbook of Genetic Algorithms. Editor I. Chambers. V. 1.— Washington, USA: CRC Press, 1995.
24. Practical Handbook of Genetic Algorithms. Editor I. Chambers. V. 2.— Washington, USA: CRC Press, 1995.
25. Practical Handbook of Genetic Algorithms. Editor I. Chambers. V. 3.— Washington, USA: CRC Press, 1999.
26. Кордюм В. А. Эволюция и биосфера.— Киев: Наукова думка, 1982.
27. Кейлоу. Принципы эволюции / Пер. с англ.— М.: Мир, 1968.
28. Эволюционная эпистемология и логика социальных наук: Карл Поппер и его критики // Составители Д. Г. Лахути, В. Н. Садовский, В. К. Финн.— М.: Эдиториал УРСС, 2000.
29. Кастти Дж. Большие системы. Связность, сложность и катастрофы / Пер. с англ.— М.: Мир, 1982.
30. Добровичев О. В. Неустойчивое равновесие коллективных систем физико-химической, биологической и социальной природы // Российский химический журнал. — 1995. — Т. 39. — № 2. — С. 48–54.
31. Гумилев Л. П. Этносфера: История людей и история природы.— М.: Экопрос, 1993.
32. Малафеева А. А. Теория моделирования процессов эволюции и управления в сложных системах / Дис. ... док. техн. наук.— Владимир, 1999.
33. Тарасов В. Б. Новые стратегии реорганизации и автоматизации предприятий: на пути к интеллектуальным предприятиям // Новости искусственного интеллекта.— 1996.— № 4.— С. 40–84.
34. Тарасов В. Б. От многоагентных систем к интеллектуальным организациям: философия, психология, информатика.— М.: Эдиториал УРСС, 2002.
35. Искусственный интеллект: В 3 кн. Кн. 1. Системы общения и экспертные системы. Справочник / Под ред. Э. В. Попова.— М.: Радио и связь, 1990.
36. Искусственный интеллект: В 3 кн. Кн. 2. Модели и методы. Справочник / Под ред. Д. А. Поспелова.— М.: Радио и связь, 1990.
37. Нильсон Н. Принципы искусственного интеллекта.— М.: Радио и связь, 1985.
38. Хант Э. Искусственный интеллект.— М.: Мир, 1978.
39. Левин Р. и др. Практическое введение в технологию искусственного интеллекта и экспертных систем с иллюстрациями на Бейсике.— М.: Финансы и статистика, 1990.
40. Поспелов Д. А. «Десять горячих точек» в исследованиях по искусственному интеллекту // Интеллектуальные системы (МГУ). — 1996. — Т. 1, вып. 1–4. — С. 47–56.
41. Попов Э. В., Фридман Г. Р. Алгоритмические основы интеллектуальных роботов и искусственного интеллекта.— М.: Наука, 1976.

42. Тарасов В. Б. Искусственная жизнь и нечеткие эволюционные системы — основные теоретические подходы к построению интеллектуальных организаций // Известия АН. Сер.: Теория и системы управления.— 1998. — № 5. — С. 12–23.
43. Суворов В. В. Интеллект в нейробиологии, психологии и интерактивных технологиях.— М.: Изд-во МГУ, 1999.
44. Уинстон П. Искусственный интеллект.— М.: Мир, 1988.
45. Эндрю А. Искусственный интеллект / Пер. с англ.// Под ред. и с предисл. Д. А. Поспелова.— М.: Мир, 1985.
46. Шкловский В. Б. Вселенная, жизнь, разум.— М.: Наука, 1989.
47. Варшавский В. И., Поспелов Д. А. Оркестр играет без дирижера: размышления об эволюции некоторых технических систем и управлении ими.— М.: Наука, 1984.
48. Редько В. Г. Эволюционная кибернетика.— М.: Наука, 2001.
49. Лорьер Ж. Л. Системы искусственного интеллекта.— М.: Мир, 1991.
50. Поспелов Г. С. Искусственный интеллект — основа новой информационной технологии.— М.: Наука, 1988.
51. Таранов П. С. Мудрость трех тысячелетий.— М.: Изд-во АСТ, 1998.
52. Таранов П. С. Энциклопедия высокого ума.— М.: Изд-во АСТ, 1997.
53. Таранов П. С. Философия сорока пяти поколений.— М.: Изд-во АСТ, 1998.
54. Таранов П. С. Золотая философия.— М.: Изд-во АСТ, 1999.
55. Платон. Сочинения. В четырех томах. Т. 1.— М.: Мысль, 1990.
56. Диоген Лаэртский. О жизни, учениях и изречениях знаменитых философов.— М.: Мысль, 1979.
57. Осуга С. Обработка знаний.— М.: Мир, 1989.
58. Уэно Х. и др. Представление знаний.— М.: Мир, 1989.
59. Осипов Г. С. Приобретение знаний интеллектуальными системами.— М.: Наука, 1997.
60. Вагин В. П. Дедукция и обобщение в системах принятия решений.— М.: Наука, 1988.
61. Пепейвода И. И. Прикладная логика. Учебное пособие.— Новосибирск.: Изд-во НГУ, 2000.
62. Будущее искусственного интеллекта // Под ред. К. Е. Левитина и Д. А. Поспелова. — М.: Наука, 1991.
63. Поспелов Д. А., Осипов Г. С. Прикладная симеотика // Новости искусственного интеллекта.— М.: 1999.— № 1.— С. 9–35.
64. Синергетика // Труды семинара. Том 3.— М.: Изд-во МГУ, 2000.
65. Синергетика // Труды семинара. Том 4.— М.: Изд-во МГУ, 2001.

66. Финн В. К. Философские проблемы логики ИС // Новости искусственного интеллекта.— М.: 1999.— № 1.— С. 36–51.
67. Моисеев И. И. Математические задачи системного анализа.— М.: Наука, 1981.
68. Моисеев И. И. Современный рационализм.— М.: МГВН Кокс, 1995.
69. Месарович М., Мако Д., Такахара. Теория иерархических многоуровневых систем.— М.: Мир, 1973.
70. Хакен Г. Синергетика. Иерархия неустойчивостей в самоорганизующихся системах и устройствах.— М.: Мир, 1985.
71. Хакен Г. Информация и самоорганизация.— М.: Мир, 1991.
72. Колесников А. А. Синергетическая теория управления.— Таганрог: ТРТУ; М.: Энергоатомиздат, 1994.
73. Климонтович И. Ю. Без формул о синергетике.— Минск: Высшая школа, 1985.
74. Синергетика и методы науки.— С.-Пб.: Наука, 1998.
75. Василькова В. В. Порядок и хаос в развитии социальных систем. (Синергетика и теория социальной самоорганизации).— С.-Пб.: Лань, 1999.
76. Пригожин И., Стенгерс И. Время, хаос, квант. К решению парадокса времени.— М.: Эдиториал УРСС, 2000.
77. Синергетика и психология. Тексты. Вып. 1.— М.: Союз, 1997.
78. Анохин П. К. Избранные труды. Кибернетика функциональных систем.— М.: Медицина, 1998.
79. Климантович Ю. Л. Введение в физику открытых систем. Синергетика // Труды семинара. Том 3.— М.: Изд-во МГУ, 2000.— С. 100–143.
80. Конфуций. Уроки мудрости: Сочинения.— М.: ЭКСМО-Пресс, 1999.
81. Горский Ю. М. Основы гомеостатики. (Гармония и дисгармония живых, природных, социальных и искусственных систем).— Иркутск: Изд-во ИГЭЛ, 1988.
82. Теслинов А. Г. Развитие систем управления: методология и концептуальные структуры.— М.: Глобус, 1998.
83. Галицын Г. А., Петров В. М. Гармония и алгебра живого.— М.: Знание, 1990.
84. Эбелинг В., Энгель А., Файстель Р. Физика процессов эволюции.— М.: Эдиториал УРСС, 2001.
85. Пайтген Х. О., Рохтер П. Х. Красота фракталов.— М.: Мир, 1990.
86. Лоскутов А. Ю., Михайлов А. С. Введение в синергетику.— М.: Наука, 1990.
87. Кроновер Р. М. Фракталы и хаос в динамических системах. Основы теории.— М.: Постмаркет, 2000.
88. Holland John H. Adaptation in Natural and Artificial Systems: An Introductory Analysis with Application to Biology, Control, and Artificial Intelligence.— USA: University of Michigan, 1975.

89. *Goldberg David E.* Genetic Algorithms in Search, Optimization, and Machine Learning.— USA: Addison-Wesley Publishing Company, Inc., 1989.
90. Handbook of Genetic Algorithms / Edited by Lawrence Davis.— USA, New York: Van Nostrand Reinhold, 1991.
91. Эволюционные вычисления и генетические алгоритмы / Составители: Э. Д. Гудман, А. П. Коваленко // Обзорение прикладной и промышленной математики.— М.: Изд-во ТВП, 1996.
92. *Батищев Д. А.* Генетические алгоритмы решения экстремальных задач.— Воронеж: Изд-во ВГТУ, 1995.
93. *Букатова И. Л.* Эволюционное моделирование и его приложения.— М.: Наука, 1991.
94. *Букатова И. Л.* Эволюционные технологии — средства интенсивной информатизации. — М.: РАН, ИРЭ, препринт № 5(593), 1994.
95. *Расстригин Л. А.* Статистические методы поиска.— М.: Наука, 1968.
96. *Rastrigin L. A.* Random Search in Evolutionary Computations // Proceedings 1st International conf., Evolutionary Computation and Its Application, EvCA, 96.— Moscow, 1996.— P. 135–143.
97. *Koza J. R.* Genetic Programming.— Cambridge/MA:MIT Press, 1992.
98. *Koza J. R.* Genetic Programming, 2.— Cambridge/MA:MIT Press, 1994.
99. Genetics Algorithms / Editor Lawrence Elbaum // Proceedings of the 1st International conf.— New Jersey, USA: Associates Publishers, 1985.
100. Genetics Algorithms / Editor J. Grefenstette // Proceedings of the 2nd International conf.— New Jersey, USA: Associates Publishers, 1987.
101. Genetic Algorithm / Editor D. Schaffer // Proceedings 3d International conf., San Mateo.— USA: Morgan Kaufman Publishers, 1989.
102. Genetics Algorithms / Editors R. Belew, L. Booker // Proceedings of the 4th International conf., San Mateo.— USA: Morgan Kaufman Publishers, 1991.
103. Genetics Algorithms / Editor R. Forrest // Proceedings of 5th International conf., San Mateo.— USA: Morgan Kaufman Publishers, 1993.
104. Genetics Algorithms / Editor R. Forrest // Proceedings of 6th International conf., San Mateo.— USA: Morgan Kaufman Publishers, 1995.
105. Genetics Algorithms / Editor T. Back // Proceedings of the 7th International conf., San Francisco.— USA: Morgan Kaufman Publishers, Inc, 1997.
106. Genetics Algorithms / Editor David Goldberg // Proceedings of the 8th International conf., San Francisco.— USA: Morgan Kaufman Publishers, Inc, 1999.
107. *Осыка А. В.* Экспериментальное исследование зависимости скорости сходимости генетического алгоритма от его параметров // Известия АИ. Сер.: Теория и системы управления.— 1997.— № 5.— С. 100–111.

108. *Скурихин А. Н.* Генетические алгоритмы // *Новости искусственного интеллекта.* — 1995. — № 4. — С. 6–46.
109. *Математическая энциклопедия.* Т. 1: А.–Г. — М.: Изд-во Сов. энцикл., 1972.
110. *Алексеев О. В. и др.* Автоматизация проектирования радиоэлектронных средств. — М.: Высшая школа, 2000.
111. *Курицкий Б. Я.* Оптимизация вокруг нас. — Л.: Машиностроение, 1989.
112. *Карелин В. П., Родзин С. И.* УМП по методам математического программирования (поисковой оптимизации). — Таганрог: Изд-во ТРТУ, 1999.
113. *Батищев Д. И., Львович Я. Е., Фролов В. Н.* Оптимизация в САПР. — Воронеж: Изд-во ВГУ, 1997.
114. *Ботвинник М. М.* О решении неточных переборных задач. — М.: Советское радио, 1979.
115. *Осипов Г. С.* Состояние и исследование и несколько слов о будущем // *Новости искусственного интеллекта.* — 2002. — № 1. — С. 3–13.
116. *Емельянов С. В., Нанпельбаум Э. Л.* Методы анализа сложных систем. Выбор в условиях неопределенности // *Итоги науки и техники. Сер.: Техническая кибернетика.* — М.: ВИНТИ, 1978. — Т. 9, ч. 1, 2.
117. *Sherwani Naveed.* Algorithms for VLSI Physical Design Automation. — Boston/Dordrecht/London: Kluwer Academic Publishers, 1995.
118. *Курейчик В. М.* Математическое обеспечение конструкторского и технологического проектирования с применением САПР. — М.: Радио и связь, 1990.
119. *Potts C. I., Giddens T. D., Yadav S. B.* The Development and Evaluation of an Improved Genetic Algorithm Based on Migration and Artificial selection // *IEEE Trans. on Systems, Man and Cybernetics.* — 1994. — V. 24. — № 1. — P. 73–86.
120. *Shahookar K., Mazumder P.* A Genetic Approach to standard Cell Placement Using Meta-Genetic Parameter Optimization // *IEEE Trans. on CAD.* — 1990. — V. 9. — № 5. — P. 500–511.
121. *Cohon J. P., Paris W. D.* Genetic Placement // *IEEE Trans. on CAD.* — November, 1987. — V. 6. — № 6. — P. 956–964.
122. *Davis L.* Genetic Algorithms and Simulated Annealing. — San Mateo, USA: Morgan Kaufman Publisher, 1987.
123. *Харари Ф.* Теория графов. — М.: Мир, 1977.
124. *Кристофидес Н.* Теория графов. Алгоритмический подход. — М.: Мир, 1978.
125. *Кини Р. Л., Райфа Х.* Принятие решений при многих критериях: предпочтения и замещения. — М.: Радио и связь, 1981.
126. *Ларичев О. И.* Теория и методы принятия решений, а также Хроника событий в Волшебных Странах. — М.: Логос, 2000.
127. *Трахтенгерц Э. А.* Компьютерная поддержка принятия решений. — М.: Синтез, 1998.

128. Proceedings of the 2nd International conf. Adaptive Computing in Engineering Design and Control'96.— Uk, Plymouth, March, 1996.
129. Абилов Ю. А., Алиев Р. А., Насиров И. М. ГА с групповым выбором и направленной мутацией // Известия АН. Сер.: Теория и системы управления.— 1997.— № 5.— С. 96–99.
130. Күрейчик В. М., Родзин С. И. Эволюционные алгоритмы: генетическое программирование // Известия АН. Сер.: Теория и системы управления.— 2002.— № 1.— С. 127–137.
131. Nordin P., Banzhaf W. Evolving turing-complete programs for a register machine with self-modifying code. Proceedings of the sixth inter. conf. on genetic Programming.— San Francisco: Morgan Kaufmann, 1995.
132. Клар Дж. Системология. Автоматизация решения системных задач / Пер. с англ. — М.: Радио и связь, 1990.
133. Шеннон Р. Имитационное моделирование — искусство и наука.— М.: Мир, 1978.
134. Емельянов В. В., Ясиновский С. И. Введение в интеллектуальное имитационное моделирование. Язык РДО.— М.: АНВИК, 1998.
135. Емельянов В. В., Ясиновский С. И. Гибридная система для планирования производства на основе генетических алгоритмов, методов имитации и экспертных систем // Известия ТРТУ.— 1996.— № 3.— С. 4–9.
136. Емельянов В. В. Гибридные системы управления на основе интеллектуальной имитационной среды РДО // Труды II-ой Международная конференция «Проблемы управления и моделирования в сложных системах».— Самара: 2000.— С. 263–268.
137. Джексон П. Введение в экспертные системы / Пер. с англ.: Уч. пос.— М.: Вильямс, 2001.
138. Статические и динамические экспертные системы. Уч. пос. / Э. В. Попов, И. Б. Фоминых, Е. Б. Кисель, М. Д. Шапот.— М.: Финансы и статистика, 1996.
139. Yasinovsky S. I. Genetic algorithm based hybrid system for job-shop scheduling // Proc. of First Int. Confr. on Evolutionary Computation and Its Applications. EvCA'96.— Moscow, June 24–27, 1996.— P. 316–320.
140. Фон Нейман Дж. Теория самовоспроизводящихся автоматов / Пер. с англ.— М.: Мир, 1971.
141. Цетлин М. Л. Исследования по теории автоматов и моделированию биологических систем.— М.: Наука, 1969.
142. Варшавский В. И. Коллективное поведение автоматов.— М.: Наука, 1973.
143. Городецкий В. И., Грушинский М. С., Хабалов А. В. Многоагентные системы (обзор) // Новости искусственного интеллекта.— 1998.— № 2.— С. 64–116.
144. Тарасов В. Б. Агенты, многоагентные системы, виртуальные сообщества: стратегическое направление в информатике и искусственном интеллекте // Новости искусственного интеллекта.— 1998.— № 2.— С. 5–63.

145. *Поспелов Д. А.* Ситуационное управление. Теория и практика.— М.: Наука, 1986.
146. *Artificial Life* / Ed. by C. G. Lagton.— Redwood-City: Addison Wesley, 1989.
147. *Artificial Life* / Ed. by C. G. Lagton, Ch. Taylor, J. D. Farmer, S. Rasmussen.— Redwood-City: Addison Wesley, 1992.
148. *Амосов Н. М., Касаткин А. М., Касаткина Л. М., Талаев С. А.* Автоматы и разумное поведение.— Киев: Наукова думка, 1973.
149. *Глушков В. М.* О кибернетике как науке // Кибернетика, мышление, жизнь.— М.: Мысль, 1964.
150. *Колмогоров А. Н.* Автоматы и жизнь // Возможное и невозможное в кибернетике.— М.: Изд-во АН СССР, 1964.
151. *Ляпунов А. А.* Проблемы теоретической и прикладной кибернетики.— М.: Наука, 1980.
152. *Поспелов Д. А., Пушкин В. Н.* Мышление и автоматы.— М.: Сов. радио, 1972.
153. *Поспелов Д. А.* Вероятностные автоматы.— М.: Энергия, 1970.
154. *Гааза-Рапопорт М. Г., Поспелов Д. А.* От амёбы до робота: модели поведения.— М.: Наука, 1987.
155. *Галимов Э. М.* Феномен жизни: между равновесием и нелинейностью. Происхождение и принципы эволюции.— М.: Едиториал УРСС, 2001.
156. *Емельянов В. В., Разумцова Ю. В., Назарова А. Е.* Многоагентная система управления складским комплексом // Реинжиниринг бизнес-процессов на основе современных информационных технологий. Системы управления знаниями. Труды 6-й Российской научно-практической конференции (РБП-СУЗ-2002).— М.: МЭСИ, 2002. — С. 114–118.
157. *Валькман Ю. Р.* Интеллектуальные технологии исследовательского проектирования: формальные системы и семиотические модели.— Киев: Port-Royal, 1998.
158. *Смирнов С. В.* Онтологический анализ в системах компьютерного моделирования: Дис. ... докт. техн. наук.— Самара, 2002.
159. *Корячко В. П., Курейчик В. М., Норенков И. П.* Теоретические основы САПР.— М.: Энергоатомиздат, 1987.
160. *Бершадский А. М.* Применение графов и гиперграфов для автоматизации конструкторского проектирования РЭА и ЭВА.— Саратов: Изд-во СГУ, 1993.
161. *Кормен Т., Лейзерсон И., Ривест Р.* Алгоритмы: построения и анализ.— М.: МЦМО, 2000.
162. *Морозов К. К. и др.* Методы разбиения схем РЭА на конструктивно законченные части.— М.: Советское радио, 1978.
163. *Курейчик В. М., Курейчик В. В.* Генетический алгоритм разбиения графа // Известия АН. Сер.: Теория и системы управления.— 1999.— № 5.— С. 79–87.

164. Кнут Д. Искусство программирования. Т. 3.— М.: Финансы и статистика, 2002.
165. Курейчик В. М., Курейчик В. В. Фрактальный алгоритм разбивания графа // Известия АН. Сер.: Теория и системы управления.— 2002.— № 4.— С. 79–87.
166. Kling R. M., Banerjee P. Empirical and Theoretical Studies of the Simulated Evolution Method applied to standard Cell Placement // IEEE Trans. on CAD. — 1991.— V. 10, № 10.— P. 1303–1315.
167. Shahookar R., Mazumder P. VLSI Placement Techniques ACM Computing Surveys.— 1991.— V. 23.— № 26.— P. 142–220.
168. Paris W. GENIF: A new placement Algorithms / Thesis.— USA: University of Virginia, 1989.
169. Mayer M. Parallel GA for the DAG Vertex spelling problem / Thesis.— USA: University of Missouri, 1993.
170. Saab Y., Rao V. Stochastic Evolution: A Fast Effective Heuristic for some General Layout Problems / Proceedings 27th, ACM //IEEE Design automation conference (DAC).— June, 1990.— P. 16–31.
171. Курейчик В. М., Курейчик В. В. Генетический алгоритм размещения графа // Известия АН. Теория и системы управления.— 2000.— № 5.— С. 67–74.
172. Cohoon J. and e. a. Distributed Genetic Algorithms for the Floorplan Design Problem // IEEE Trans. on CAD.— April, 1991.— V. 10, № 4.— P. 483–492.
173. Kureichik V., Goodman E., Tetelbaum A. Genetic Algorithm Approach to Compaction, Bin Packing, and Nesting Problems, Technical Report 940702.— Michigan State University, 1994.
174. Kureichik V., Davidenko V., Miagkikh V. Genetic algorithms for restrictive channel routing // Proc. of the 7th International conference on Genetic Algorithms.— USA, MSU, East Lansing, 1997.— P. 636–642.
175. Liu X., Sakamoto A., Shimamoto T. Restructive Channel Routing with Evolution Programs // Trans. IEICE.— 1993.— V. E76-A.— № 10.— P. 1738–1745.
176. Yoshimura T., Kuh E. S. Efficient algorithms for channel routing // IEEE Trans. Comput.-Aided Des. Integrated Circuits & Syst.— 1982.— V. 1.— № 1.— P. 25–35.
177. Burstein M. Channel routing. Layout Design and Verification.— Elsevier Science, 1986.— P. 133–167.
178. Tseng H. P., Sechen C. A Gridless Multi-Layer Channel Router Based on a Combined Constraint Graph and Tile Expansion Approach // Department of Electrical Engineering, Box 352500, University of Washington, Seattle, WA 98195, ISPD.— 1996.— P. 210–217.
179. Lieng J., Thulasiraman K. A Genetic Algorithm for Channel Routing in VLSI Circuits, Evolutionary Computation.— MIT, 1994.— V. 1(4).— P. 293–311.
180. Reed J., Sangiovanni-Vincentelli A., Santomauro M. A New Symbolic Channel Router: YACR2 // IEEE Transactions on Computer-Aided Design. —July, 1985.— V. CAD-4, № 3.

181. Joobbani R., Siewiorek D. P. WEAVER: A Knowledge-Based Routing Expert // IEEE Design & Test.— February, 1986.— P. 12–23.
182. Oliver I., Smith D., Holland J. R. A study of permutation crossover operators on the traveling salesman problem // Proc. of the Second International Conf. on Genetic Algorithms.— P. 224–230.
183. Kureichik V. M., Miagkikh V. et. al., Some New Features in Genetic Solution of the TSP // Proceedings of the second International conference.— Plymouth, UK, 1996.— P. 294–296.
184. Курейчик В. В. Построение и анализ генетических алгоритмов раскраски графа на основе моделей искусственных систем // Труды международного конгресса ICAI'2001. Искусственный интеллект в 21-м веке.— М.: ФИЗМАТЛИТ, 2000.— С. 665–675.
185. Нечепуренко М. И. и др. Алгоритмы и программы решения задач на графах и сетях.— Новосибирск: Наука, 1990.
186. Falkenauer E. Setting New Limits in Bin Packing with a Grouping GA Using Reduction // Technical report R0108.— Belgium, March 1994.
187. Kao, Cheng-Yan A Stochastic Approach for The One-Dimensional Bin-Packing Problems // Proc. Int. Conf. on Systems, Man, and Cybernetics.— Chicago, 1992.— P. 1545–1551.
188. Kureichik V., Tetelbaum A. Graph Isomorfizm Algorithm for Regular VLSI Structure // Proc. 28th Annual conf. in Information Sciences and systems.— Princeton, USA, 1994.— P. 17–23.
189. Read R. C., Cornell D. G. The graph isomorphism disease // Journal of Graph Theory, USA.— 1977.— № 1.— P. 339–363.
190. Курейчик В. М., Королев А. Г. Метод для распознавания изоморфизма графов // Кибернетика, АН УССР.— Киев, 1977.— № 2.— С. 15–21.
191. Warnaar D. B., Chew M., Olarin S. Method for detecting isomorphism in graphs using vertex degree correspondence with partitioning // American society of Mechanical Engineers, DE.— USA, 1993.— V. 47.— P. 219–224.
192. Курейчик В. В. Генетический алгоритм определения изоморфизма однородных графов // Известия ТРТУ.— Таганрог, 1997.— № 3.— С. 60–63.
193. Курейчик В. М. Генетические алгоритмы. Состояние. Проблемы. Перспективы // Известия АН. Теория и системы управления. — 1999.— № 1.— С. 144–160.
194. Saab Y. G., Rao V. B. Fast Effective Heuristics for the Graph Bisectioning Problem // IEEE, Transactions on CAD. — January 1990. — V. 9.— № 1. — P. 91–98.
195. Wei Y. C., Cheng C. K. A two-level two-way partitioning algorithm // Tech. report CH2924-9, University of California, San Diego, IEEE, 1990.
196. Ching-Wei Yeh, Chung-Kuan Cheng, Ting-Ting Y. Lin. A general purpose multiple way partitioning algorithm // Proceedings 28th ACM/IEEE Design Automation Conference, paper 25/1, 1991. — P. 421–425.

197. *Kernighan B., Lin S.* An efficient heuristic procedure for partitioning graphs // *Bell Syst. Tech. J.* — Feb 1970. — V. 49. — P. 291–307.
198. *Fiduccia C., Mattheyses R.* A linear time heuristics for improving network partitions // *Proceedings 19th ACM/IEEE Design automation conference.* — 1982. — P. 175–181.
199. *Saab Y.* A new effective and efficient multi-level partitioning algorithm // *Proceedings Design, Automation and Test in Europe Conference 2000, Paris, France, 27–30 March 2000.* — P. 112–116.
200. *Bui T. N., Moon B. R.* Genetic algorithm and graph partitioning // *IEEE Trans. Comput.* — 1996. — V. 45. — P. 841–855.
201. *Мелихов А. Н., Беритейн Л. С., Курейчик В. М.* Применение графов для проектирования дискретных устройств. — М.: Наука, 1974.
202. *Канторович Л. В., Залгаллер В. А.* Рациональный раскрой промышленных материалов. — Новосибирск: Наука, 1971.
203. *Афонин П. В.* Система рационального раскроя материала с применением генетического оптимизационного алгоритма // Четвертая Международная летняя школа-семинар по искусственному интеллекту для студентов и аспирантов. Сборник научных трудов. — Минск: 2000. — С. 125–128.
204. Исследование операций: В 2-х томах. Том 2. Модели и применение / Пер. с англ. // Под ред. Дж. Моудера, С. Элмаграби. — М.: Мир, 1981.
205. *Захаров П. А.* Гибридные системы и принятие решений при управлении запасами // *Интеллектуальное управление: новые интеллектуальные технологии в задачах управления (ICIT'99).* Труды международной конференции, Переславль-Залесский. — М.: ФИЗМАТЛИТ, 1999. — С. 164–168.
206. *Емельянов В. В., Крючков М. Ю., Штаутмайстер Т.* Оптимальное оперативное планирование участка обработки ДСП // *Компьютерная хроника.* — 1998. — № 1. — С. 49–60.
207. *Емельянов В. В., Штаутмайстер Т.* Оперативное управление раскромом материала на лесоперерабатывающем предприятии. — М.: АНВИК, 1999.

Предметный указатель

- Абдукции 59
- Автомат 202, 205
 - с линейной тактикой 204
 - Роббинсона 205
 - самовоспроизводящийся 202
- Агент 51, 209
 - взаимодействие 210, 216
 - мобильный 210
 - интеллектуальный (когнитивный) 209
 - реактивный 209
- Адаптация 52, 62
 - эволюционная 57
- Алгоритм
 - генетический 15, 91, 175, 216
 - дихотомического разбиения 260
 - гибридный 51, 307
 - кластерно-ориентированный 152
 - модифицированный 99
 - оптимизационный 15
 - проблемно-ориентированный 153
 - простой 96
 - размещения 357
 - градиентов сопряженный 127
 - групповой 156
 - жадный 303, 304
 - итерационный 136, 350
 - комбинированный 136
 - оптимизационный 136
 - планаризации 317
 - последовательный 136
 - разбиения графа 243, 246
 - сканирующий 158
 - точный 136
 - циклический 314
 - эвристический 262, 294
 - эволюционный 224
- Аналогия 56
- Ассимиляция 52
- Аттрактор 72
- Вершина
 - графа 211
 - мультиграфа
 - смежные 302, 317
 - список 302
- Вид 25
- Ген 21, 23
- Генетика 19
- Геном 37
- Генотип 22
- Гиперграф 234
- Гомеостаз 29, 74
- Гомеостат 74
- Граф 234, 244, 311, 312
 - гомеоморфный 311
 - изоморфный 16
 - исориентированный 311
 - планарный 311
 - плоский 311
 - полный 311, 312
 - состоящий 211
 - топологии 284
 - цикл 293
- Действие 179
- Дерево
 - Штейнера 16, 276, 277
 - покрывающие 16
- Дихотомия 15
- Дуга 211
- Жизнь искусственная 52, 202, 218
- Задача
 - инженерные 241, 333
 - канальной трассировки 281
 - комбинаторно-логические 333
 - минимизации пересечений 315
 - о коммивояжере 16, 293
 - определения изоморфизма графов 269
 - определения планарности графов 269
 - оптимизационные 10, 95, 241
 - планирования 394
 - кристалла 369
 - поставок 386
 - работ 394
 - плоской раскладки 409

- построения клик графов 302
- построения независимых подмножеств графов 302
- построения покрывающих деревьев графов 276
- разбиения графа 243
- размещения 264, 357
- раскраски графов 302
- — вершин графов 16
- раскроя материала 377, 400
- — — оптимальный 395

Задача раскроя

- — плоского 311, 377, 400
- — двумерного 311, 377
- транспортная 210
- трассировки 276
- — соединений 16, 276, 280
- — каналов 288
- упаковки блоков 374
- Штейнера 276
- NP-полная 243

Знание 35

Исчерпание 60

Изменчивость 20, 21

- направленная 21
- последовательная 19
- ненаправленную 21
- — комбинативная 22
- — мутационная 22

Изоморфизм графов 333, 338

Инверсия 102

Интеллект

- искусственный 50, 51
- — гибридный (синергетический) 52
- — распределенный 52

Кибернетика эволюционная 36

Кластер 90, 252

- квазимиимальный 252
- минимальный 252

Клик графа 16

Ковер Серпинского 88

Критерий

- Мак-Лейна 312
- Понтрягина-Куратовского 311,
- Уитни 312
- Харари-Татта 312
- Кроссиггер 21, 95

— двухточечный 97

— диагональный 160

— «жадный» 99, 297

— многоточечный 97

— однотоочечный (простой) 96

— порядковый 97

— универсальный 99

— циклический

— частично-соответствующий 98

Матрица 317

— инцидентности 314

— реберно-ориентированная 314

— смежности графа 254, 293

— циклов 314

Мейоз 23

Метод

— востой и границ 264

— горизонта 15, 132

— градиентный 126

— динамического программирования 246

— дискретной оптимизации 242

— дихотомии 122, 267, 351

— золотого сечения 15, 124, 267, 351

— итерационный 136, 248

— комбинаторный 136, 245

— математический 235

— матричный 246

— миграции и искусственной селекции 137–138

— моделирования отжига 264

— Монте-Карло 130, 267

— наискорейшего подъема 127

— наискорейшего спуска 127

— направленного перебора 121, 243, 245

— оптимизации 121, 181

— — — статистические 121, 128

— — — с накоплением 128

— — — с адаптацией 128

— — — с самообучением 129

— парных и групповых перестановок 245

— пенсовские 134

— — с возвратом 131

— последовательного приближения 245

— «прерывистого равновесия» 137

— релаксации 126, 245

- случайного поиска 130
- стохастически-итерационный 128, 143
- Фибоначчи 15, 123, 270
- циклический 312
- численный 125
- эвристический 246
- эволюционный 175
- Миграция 229
- Митоз 23
- Множество
 - Кантора 88
 - независимое 302
 - подграфов 243
 - подмножеств 132
 - устойчивое 72
 - фрактальное 88
- Моделирование
 - имитационное 14, 44, 396
 - интеллектуальное 178
 - интегрированное 44
 - эволюционное 9, 11, 175, 202
- Модель 233
 - адекватность 177, 236
 - алгоритмическая 236
 - аналитическая 234
 - аналоговая 236
 - гиперциклов 39
 - дискретная 236
 - имитационная 177, 180, 195, 237, 373
 - математическая 234, 235
 - эмпирическая 234
 - полная 234
 - квазивидов 36, 38
 - коллективного поведения 202
 - нейтральной эволюции 36, 39
 - сайзров 39
 - эмпирическая 235
- Мутация 21, 95, 193
 - двухточечная 101
 - однотоочечная (простая) 100
 - позиционная 101
- Наследственность 20
- Оператор 95
 - вставки 103
 - инверсии 101
 - кроссинговера 21
- — двухточечный 97
- — диагональный 160
- — жадный 99, 297
- — многоточечный 97
- — модифицированный 258, 297
- — модифицированный жадный 301
- — однородный 305
- — однотоочечный (простой) 96
- — порядковый 97
- — универсальный 99
- — циклический 98
- — частично-соответствующий 98
- — мутации 100
- — двухточечный 101
- — модифицированный 162
- — однотоочечный 100
- — позиционный 101
- — рекомбинации 102
- — сегрегации 102
- — селекции (репродукции) 95
- — на основе заданной шкалы 96
- — на основе рулетки 96
- — турнирный 96
- — элитный 96
- — транслокации 102
- — удаления 103
- — частичной мутации 97
- Операция 179
- Оптимум
 - глобальный 44
 - квазиоптимальный 132
 - локальный 81
- Организация
 - виртуальная 49
 - интеллектуальная 47
 - постсейлоровская 46
 - сейлоровская 46
- Отбор 24
 - бессознательный 24
 - движущий (ведущий) 24
 - дизруптивный (разрывающий) 25
 - естественный 20, 21
 - искусственный 20
 - методический 24
 - стабилизирующий 24
- Перебор направлений 245
- Подход
 - эволюционный 11

- генетический 62
- гомеостатический 62
- комбинированный 12
- синергетический 62
- системный 14
- Поиск
 - в глубину 15, 132, 318
 - в ширину 15, 132, 245
- генетический 96
- — с миграцией 229
- градиентный 121
- двунаправленный 131
- комбинаторный 134, 136
- локализованный 30
- модифицированный 121
- на И-ИЛИ дереве 15
- одномерный 122
- пассивный 15, 122
- последовательный 12, 122
- равномерный 126
- с возвращением 15
- случайный 130
- эволюционный 91
- Популяция 21, 25
 - автоматов 201, 203
- Правило
 - продукционное 179, 211
 - продукционное модифицированное 178
- Программирование
- генетическое 167, 175
- динамическое 246
- эволюционное 175
- Процесс 181
 - итерационный 247
- Разбиение 16
 - итерационное 257
- Размещение 16
- Рассуждение 56
- Рескомбинация 103
- Релаксация 245
- Репродукция 104
- Ресурс 179
- Самоорганизация 42, 62, 64, 65, 73
- Связность структурная 40
- Селекция 95
 - на основе заданной шкалы 96
 - на основе рулетки 96
 - турнирная 96
 - элитная 96
- Синергетика 14, 65, 72
- Синтез
 - структурный 234
 - параметрический 234
- Система
 - гибридная 177, 198, 223, 378, 393
 - динамическая 84, 239
 - естественная 10
 - имитационная 10
 - искусственная 10, 19
 - — интеллектуальная 10, 59
 - информационная 44
 - многоагентная 51, 218
 - многомодельная 228
 - открытая 60
 - производственная 43
 - сложная 40
 - экспертная 54, 57
 - — динамическая 41, 54–57
 - эволюционная 55, 195
- Скрещивание 95, 182, 184
- Слот 57
- Стратегия
 - жадная 267, 297
 - золотого сечения 267
 - эволюционная 175
- Точка принятия решений 212
- Трассировка соединений 16
- Триада 36
- Фенотип 21
- Фрактал 87
- Фрейм 57
- Функция
 - пригодности (полезности) 93
 - целевая 93, 104
- Хаос 67, 73, 303
- Хромосома (особь, индивидуальность) 20
- Шаблон 296
- Эвристика
 - жадная 302, 303
 - специальная 303
 - последовательная 302

- Эволюция 10, 19
 - де Фриза 13, 31
 - Ж. Ламарка 13, 29
 - К. Поппера 35
 - макро 278
 - мста 278
 - микро 278
 - нейтральная 39
 - прогрессивная 34
 - синергетическая (интегрированная) 13, 32, 33
 - теологическая 29
 - теория 27
 - Ч. Дарвина 13, 24
- Экспертная система 54, 195
- Энтропия 66, 67

Научное издание

ЕМЕЛЬЯНОВ Виктор Владимирович

КУРЕЙЧИК Владимир Викторович

КУРЕЙЧИК Виктор Михайлович

ТЕОРИЯ И ПРАКТИКА ЭВОЛЮЦИОННОГО МОДЕЛИРОВАНИЯ

Редактор *И.Л. Легостаева*

Оригинал-макет: *Е.В. Третьяков*

Оформление переплета: *А.Ю. Алехина*

ЛР № 071930 от 06.07.99. Подписано в печать 19.06.03.

Формат 60×90/16. Бумага офсетная. Печать офсетная.

Усл. печ. л. 27. Уч.-изд. л. 29,7. Тираж 300 экз. Заказ №

Издательская фирма «Физико-математическая литература»

МАИК «Наука/Интерпериодика»

117997 Москва, Профсоюзная, 90

E-mail: fizmat@maik.ru

Отпечатано с диапозитивов

в РГУП «Чебоксарская типография № 1».

428019 Чебоксары, пр. И. Яковлева, 15