В. Босс

ЛЕКЦИИ *по* МАТЕМАТИКЕ

ТОМ 10

Перебор и эффективные алгоритмы

МОСКВА

URSS

ББК 22.18я73

Босс В.

Лекции по математике. Т. 10: Перебор и эффективные алгоритмы: Учебное пособие. — М.: Издательство ЛКИ, 2008. — 216 с.

Книга посвящена теории сложности алгоритмов в той ее части, где речь идет о противостоянии P- и NP-задач. В резонанс с проблемой «P против NP» входит обширная тематика: комбинаторные задачи на графах, неразрешимые проблемы теории алгоритмов, криптография, целочисленное программирование, вероятностные методы, квантовые вычисления, алгоритмы Хачияна и Кармаркара для линейного программирования, а также полиномиальный алгоритм AKS для выяснения простоты числа. Особое внимание уделяется геометрическому взгляду на проблему, который в привычном уже пейзаже обнаруживает свежие ракурсы.

Изложение отличается краткостью и прозрачностью.

Для студентов, преподавателей, инженеров и научных работников.

Издательство ЛКИ. 117312, г. Москва, пр-т Шестидесятилетия Октября, д. 9.
Формат 60×90/16. Печ. л. 13,5. Зак. № 1463.

Отпечатано в ООО «ЛЕНАНД».
117312, г. Москва, пр-т Шестидесятилетия Октября, д. 11А, стр. 11.

ISBN 978–5–382–00642–0

НАУЧНАЯ И УЧЕБНАЯ ЛИТЕРАТУРА

E-mail: URSS@URSS.ru
Каталог изданий в Интернете:
http://URSS.ru
Тел./факс: 7 (499) 135–42–16
URSS Тел./факс: 7 (499) 135–42–46

5774 ID 72195

Оглавление

Предисловие к «Лекциям»

Лучше изредка спотыкаться на деталях,
чем утопить общую картину в формальностях.

Для нормального изучения любого математического предмета необходимы, по крайней мере, 4 ингредиента:

1) *живой учитель*;
2) *обыкновенный подробный учебник*;
3) *рядовой задачник*;
4) *учебник, освобожденный от рутины, но дающий общую картину, мотивы, связи, «что зачем».*

До четвертого пункта у системы образования руки не доходили. Конечно, подобная задача иногда ставилась и решалась, но в большинстве случаев — при параллельном исполнении функций обыкновенного учебника. Акценты из-за перегрузки менялись, и намерения со второй-третьей главы начинали дрейфовать, не достигая результата. В виртуальном пространстве так бывает. Аналог объединения гантели с теннисной ракеткой перестает решать обе задачи, хотя это не сразу бросается в глаза.

«Лекции» ставят 4-й пункт своей главной целью. Сопутствующая идея — экономия слов и средств. Правда, на фоне деклараций о краткости и ясности изложения предполагаемое издание около 20 томов может показаться тяжеловесным, но это связано с обширностью математики, а не с перегрузкой деталями.

Необходимо сказать, на кого рассчитано. Ответ «на всех» выглядит наивно, но он в какой-то мере отражает суть дела. Обозримый вид, обнаженные конструкции доказательств, — такого сорта книги удобно иметь под рукой. Не секрет, что специалисты самой высокой категории тратят массу сил и времени на освоение

математических секторов, лежащих за рамками собственной специализации. Здесь же ко многим проблемам предлагается короткая дорога, позволяющая быстро освоить новые области и освежить старые. Для начинающих «короткие дороги» тем более полезны, поскольку облегчают движение любыми другими путями.

В вопросе «на кого рассчитано», — есть и другой аспект. На сильных или слабых? На средний вуз или физтех? Опять-таки выходит «на всех». Звучит странно, но речь не идет о регламентации кругозора. Простым языком, коротко и прозрачно описывается предмет. Из этого каждый извлечет свое и двинется дальше.

Наконец, последнее. В условиях информационного наводнения инструменты вчерашнего дня перестают работать. Не потому, что изучаемые дисциплины чересчур разрослись, а потому, что новых секторов жизни стало слишком много. И в этих условиях мало кто готов уделять много времени чему-то одному. Поэтому учить всему — надо как-то иначе. «Лекции» дают пример. Плохой ли, хороший — покажет время. Но в любом случае, это продукт нового поколения. Те же «колеса», тот же «руль», та же математическая суть, — но по-другому.

Предисловие к десятому то́му

Чего, собственно, ожидать на краю? На грани, где возможное переходит в невозможное, жизнь — в смерть, теорема — в парадокс. По сути — ожидать нечего. Однако, как и в поиске смысла жизни, основную роль здесь играют побочные эффекты.

Том планировалось назвать «Труднорешаемые задачи», но помешала двусмысленность толкования. Одни начинают думать о математических олимпиадах, другие — «где бы найти что-нибудь полегче». А речь-то идет о противоборстве перебора и целенаправленного поиска. Иначе говоря, содержание вращается вокруг знаменитой проблемы «P против NP», и вовлекает в круговорот многое за пределами. Можно ли кардинально избавиться от сложности решения при компактном описании исходных данных? Не вообще избавиться, а там, где перечисление организовано экономно. Ибо почему бы не найти ответ быстро, если данных много, но описание коротко? Проблема на вид проста, но ускользает, и аукается в таких закоулках, что мысль о «неисповедимых путях» обретает дополнительную опору.

Глава 1

Комбинаторные задачи

Если для размышлений над задачей нужны карандаш и бумага, постановка — еще не осознана.

Для удобства дальнейшего разговора, равно как и вообще для ориентации в области *комбинаторного анализа*, целесообразно иметь перед глазами некоторое количество конкретных задач. С точки зрения удобства вхождения в тему полезно также грубое представление о мотивах и целях предполагаемой теории. Наивный ракурс в данном случае дает вполне адекватную картину. При этом плюсы поверхностного взгляда позволяют легче одолеть этап привыкания к новым понятиям.

1.1. Экспоненциальный кошмар

Насчет «кошмара», разумеется, вопрос растяжимый. Как говорится, что немцу смерть, русскому — витамин. Такая же история с комбинаторной плодовитостью, которая практику — кость в горле, теоретику — манна небесная.

Речь идет о простом и хорошо известном явлении. Дискретные задачи часто имеют комбинаторную природу, ибо элементы множества X, среди которых ищутся подходящие варианты, задаются комбинационно. Скажем, в *задаче коммивояжера* [1] данными считаются города и расстояния, а множество X возможных маршрутов — возникает опосредованно. В итоге при нескольких десятках городов маршрутов оказывается больше, чем атомов в Галактике,

[1] *Даны n городов и расстояния r_{ij} между ними. Требуется найти циклический маршрут минимальной длины, заходящий во все города по одному разу.* Это классический оптимизационный вариант *задачи коммивояжера*. Вариант «распознавания»: *существует ли маршрут, проходящий через все города, длина которого, $r_{i_1 i_2} + r_{i_2 i_3} + \ldots + r_{i_n i_1}$, не превосходит r?*

и подобного рода «экспоненциальные взрывы»[2] в значительной мере определяют облик Мироздания. Соответственно, перспектива рассмотрения умопомрачительного числа вариантов настраивает на философский лад.

Положение дел усугубляется тем, что «непрямое» определение множества X превращает необходимость указания даже одного его элемента — в «пытку». Скажем, в *задаче целочисленного линейного программирования* X представляет собой совокупность целочисленных решений системы линейных неравенств

$$\sum_i w_{ij}x_i \leqslant W_j, \quad j = 1, \ldots, m,$$

но найти хотя бы одно решение — серьезная проблема, если не соглашаться на «тупой перебор», требующий в рядовых ситуациях $\sim 10^{100}$ лет.

В той же *задаче коммивояжера* о множестве циклических маршрутов говорится вскользь, как о само собой разумеющейся совокупности, — тогда как задача указания хотя бы одного подходящего маршрута[3] — не проще исходной (в некотором естественном смысле, что уточняется далее). Поэтому задача поиска и перечисления элементов X не является чисто технической, если рассчитывать на эффективные алгоритмы ее решения.

1.2. P- и NP-задачи

В комбинаторной оптимизации приходится чаще всего перебирать чуть ли не все варианты. Чем кончится эпопея поиска эффективных путей сокращения перебора нелегко сказать[4], но уже сегодня колоссальные безрезультатные усилия позволяют характеризовать такого сорта задачи как *труднорешаемые*. В этом океане есть аномалии. Вот один из примеров.

---

[2] Прилагательное «экспоненциальный» в переборных задачах возникает в связи с *формулой Стирлинга* $n! \sim \sqrt{n}\,(n/e)^n$, переводящей факториальные зависимости в экспоненциальные.

[3] Замкнутый маршрут, заходящий во все города по одному разу, называют *гамильтоновым циклом*.

[4] Большинство думает, что ничем.

МИНИМАЛЬНОЕ ОСТОВНОЕ ДЕРЕВО[5] **(МОД)**

В содержательной интерпретации *задача состоит в построении сети дорог минимальной стоимости, связывающей n населенных пунктов* (рис. 1.1), что на академическом языке означает *поиск минимального остовного дерева реберно-взвешенного графа.*

На фоне обычных комбинаторных неприятностей вопрос, на удивление, легко решается так называемым *жадным алгоритмом.* Сначала выбирается самая дешевая дорога. Затем — самая дешевая из оставшихся, при условии, что не образуется *цикла*, и т. д. Если на некотором шаге несколько дорог имеют минимальную стоимость — выбирается любая, не образующая цикла. Обоснование работоспособности алгоритма довольно просто. В итоге удается обойти полный перебор, получая значительный выигрыш. Алгоритм быстро завершает работу за *полиномиальное время*[6].

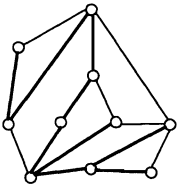


Рис. 1.1

Теория *сложности алгоритмов* ориентируется в основном на изучение времени, необходимого для завершения вычислений, и распадается на два направления. Одно из них — в некотором роде философское — связано с выяснением принципиальных ограничений. Можно или нельзя переборные задачи решать за полиномиальное время? Второе направление, сугубо прагматическое, — объединяет поиск экономных алгоритмов.

«Экономия» чаще подразумевается временна́я. Например, при умножении двух чисел n_1 и n_2 плохой алгоритм $n_1 - 1$ раз выполняет операцию

$$N(k+1) = N(k) + n_2,$$

полагая вначале $N(1) = n_2$. В итоге $N(n_1) = n_1 \cdot n_2$, но время счета, — пропорциональное n_1, — как говорится, зашкаливает. Время же работы алгоритма

[5] *Остовом* (каркасом) *связного графа* называют любую совокупность ребер, соединяющих все вершины, не содержащую *циклов*.

[6] В то время как число деревьев, которые можно построить на n вершинах, равно n^{n-2}. Неизоморфных вариантов меньше [18], но все равно экспоненциально.

«умножения в столбик» пропорционально $\log n_1 \cdot \log n_2$. В первом случае алгоритм *экспоненциальный*, во втором — *полиномиальный* [7].

Рассматриваемая область имеет три отправных точки.

• Во-первых, это понятие самого *алгоритма*, как рецепта достижения результата с помощью однозначной последовательности действий. Такое определение чересчур расплывчато, но более приемлемая дефиниция также оставляет значительную широту толкования: *алгоритм — это программа на любом универсальном языке программирования*. Возможна конкретизация в диапазоне от *машин Тьюринга* [6, т. 6] до какого-либо *алгола*. Конечно, от выбора базы моделирования алгоритма зависит время его работы, но *теория сложности вычислений* ограничивается изучением порядков роста, каковые в естественных предположениях не меняются при переходе от одного языка к другому [8].

• Во-вторых, это *длина входа* — говорят еще *размер входа*, или *длина описания задачи* в битах, т. е. количество двоичных символов, необходимых для задания исходных данных (всех параметров). Удобнее говорить о *длине описания*, куда помещается также длина выхода (ответа).

• В-третьих, это понятие *полиномиального счета*. Если работа алгоритма, решающего задачу с длиной описания x, характеризуется необходимостью выполнения $f(x)$ *элементарных операций*, то алгоритм считается *полиномиальным* в случае

$$f(x) = O(x^k) \quad \textit{при некотором} \quad k > 0. \tag{1.1}$$

Иначе говоря, если существует такая константа $C > 0$, что $f(x) < Cx^k$ при достаточно больших x. Таким образом, полиномиальность

[7] Потому что полиномиальность имеется в виду не по отношению к самим числам, а к их кодировке — позиционной записи в двоичной или другой системе счисления, кроме единичной. Число знаков, необходимое для записи числа n в k-ичной системе, имеет порядок $\log_k n$.

[8] Последний тезис общепринят, но до уровня теоремы — не доходит. Хотя иногда подразумевается именно теоремное звучание. Квантовые компьютеры (глава 10) могут поколебать удобную концепцию.

имеет *асимптотический характер*. Поэтому длину описания и «количество операций» можно измерять с точностью до умножения на положительную константу[9]. Полиномиальные алгоритмы хороши еще тем, что *их композиция снова дает полиномиальный алгоритм*.

Задачи распознавания. Большинство практических задач имеет оптимизационный характер, но пока в теории отдается предпочтение *задачам распознавания*, каковые, по определению, могут иметь только два ответа: «да» или «нет». Потери смысла при переходе к более простым задачам распознавания, вообще говоря, не происходит, но «руки оказываются заняты» несколько меньше, что освобождает ресурсы для теоретических изысканий.

Эквивалентность распознавания и оптимизации понимается в следующем смысле: если *полиномиальный алгоритм* существует для первой задачи, то полиномиально решается вторая, и наоборот. Возьмем, для примера, *задачу коммивояжера*. В одну сторону эквивалентность совсем очевидна. Определение оптимума сразу же решает задачу распознавания о существовании маршрута длины $r_{i_1 i_2} + r_{i_2 i_3} + \ldots + r_{i_n i_1} \leqslant r$. Надо лишь длину оптимального маршрута сравнить с r. *Обратно*. Решая N раз[10] задачу распознавания для различных r в диапазоне от нуля до N, определяем значение оптимума r^*. Те же рассуждения проходят и в общем случае.

1.2.1. Теорема. *Если целевая функция может принимать не более N значений, то существует дихотомическая процедура решения оптимизационной задачи за $\log_2 N$ шагов решения соответствующей комбинаторной задачи распознавания.*

Чтобы найти сам оптимальный маршрут в *задаче коммивояжера*, исключаем ребра по очереди, и заново ищем значение оптимума $\widetilde{r}^*$

[9] Вместо *количества битов* годится количество символов в любом алфавите, кроме единичного. Произвол в определении элементарных операций также весьма велик. Это могут быть арифметические операции, сдвиги головки машины Тьюринга, операторы алголоподобных программ и т. п. Все это не нарушает асимптотики (1.1), а только меняет константу в $f(x) < Cx^k$.

[10] Где N — максимально возможная сумма весов n ребер, которая оценивает сверху длину любого маршрута.

в усеченной задаче. Это позволяет установить принадлежность ребра оптимальному маршруту, каковой находится перебором всех ребер. Поэтому с точностью до полиномиальной эквивалентности асимптотика оптимизации совпадает с асимптотикой распознавания.

Описанный рецепт обеспечивает поиск *удостоверения* непосредственно в задаче распознавания, но, увы, работает не всегда — не во всех задачах. Скажем, в тандеме ПРОСТОЕ ЧИСЛО — СОСТАВНОЕ ЧИСЛО обе части полиномиально разрешимы как задачи распознавания (раздел 1.8), однако *задача факторизации числа*, разложения на множители, считается пока труднорешаемой (глава 6).

Причина фиаско использованного выше метода при поиске сомножителей ($N = N_1 \cdot N_2$) заключена в том, что здесь нет строительных кубиков, которые можно было бы контролируемо изымать [11]. По крайней мере, не ясно, как N можно «усечь», чтобы факторизация усеченного варианта помогла решению исходной задачи.

У каждого ракурса, оптимизации и распознавания, есть преимущества и недостатки. Есть также специфический инструментарий. В задачах распознавания, например, принято говорить о *распознавании языка* L, каковым называют то или иное подмножество слов $L \subset \{\mathcal{A}^n\}$ в алфавите $\mathcal{A}$.

В случае двоичного алфавита $\mathcal{A} = \{0, 1\}$ совокупность $\{\mathcal{A}^n\}$ представляет собой множество всех $(0, 1)$-последовательностей. Языком L может быть, скажем, совокупность квадратов k^2, записанных в двоичной системе, либо множество двоичных чисел, каждое из которых по определенным правилам описывает индивидуальную задачу КОММИВОЯЖЕРА. В результате появляется возможность говорить о распознавании языков безотносительно к конкретным задачам, что способствует рассасыванию аудитории до оптимальных размеров.

1.2.2. Определение. *Совокупность задач распознавания, которые могут быть решены некоторым полиномиальным алгоритмом, называется* ***классом P.***

[11] Как в *задаче коммивояжера* и других комбинаторных задачах.

Концентрация на задачах распознавания — дань традиции. Вместо задач распознавания можно с тем же успехом говорить о распознавании языков, что по сути — то же самое. С учетом приведенного выше замечания в определении 1.2.2 распознавание можно заменить оптимизацией.

1.2.3. ***Класс NP*** *определяется как совокупность* ***полиномиально проверяемых*** *задач распознавания, в которых, если решением является ответ «да», то существует «слово» x полиномиальной длины* [12] *и полиномиальный от x алгоритм, дающий ответ «да».*

Иначе говоря,

1.2.3′. *Язык $L \in$ NP, если существует полиномиально вычислимый предикат R_L такой, что слово (задача) $z \in L$, если*

$$\exists x\colon\ (z, x) \in R_L.$$

Обратим внимание на несимметричность ответов «да» и «нет». Если задача принадлежит NP, то ее *дополнение* (та же задача с ответом «нет»), вообще говоря, не обязано принадлежать NP. Совокупность дополнений к NP-задачам образует *класс* **co-NP**.

Определение 1.2.3 отличается от общепринятого [7], но предлагаемый здесь ракурс рассеивает некоторый туман в связи со стандартным (и вообще говоря, необязательным) использованием «недетерминированных машин Тьюринга». Нюансы будут обсуждаться в последующих главах.

Полиномиальная проверяемость в большинстве случаев — явление тривиальное. Существует ли маршрут длины $\leqslant r$? Если «да», то такой маршрут (слово) можно предъявить, и далее остается убедиться за полиномиальное время, что маршрут удовлетворяет предъявляемым требованиям. Существует ли что-то, обладающее свойством A? «Что-то» предъявляется и проверяется, полиномиально. Такая схема исчерпывает подавляющую часть практических ситуаций.

[12] Полиномиальной от *длины описания задачи*.

Но есть задачи иного сорта. Скажем, необходимо установить, что на заданной сети дорог подходящих циклических маршрутов вообще нет [13]. Принадлежит ли эта задача классу NP — не ясно. Похоже, необходимо перебрать и отсеять все варианты. Аромат полиномиальности не чувствуется, но чем черт не шутит. Возможно, есть «короткий» инвариант, вычисление которого дает ответ «да» за полиномиальное время.

1.3. Центральный вопрос и окружение

Проблема $\mathbf{P} \stackrel{?}{=} \mathbf{NP}$. Вопрос о совпадении или несовпадении классов P и NP до сих пор остается открытым, и это есть одна из крупнейших математических проблем, решив которую можно поправить свое материальное положение [14].

Вопрос $\mathbf{P} \stackrel{?}{=} \mathbf{NP}$ в теории труднорешаемых задач является центральным, но, ускользая и мимикрируя, он не только породил обширную периферию исследований, но и сам приобрел довольно странную форму. Проблема-то сводится к простому и ясному вопросу: «*Существует ли полиномиальный алгоритм для задачи коммивояжера*?». В этом заключена вся проблема, и соответствующий ответ стоит миллион долларов. Но «*коммивояжера*», оказывается, можно заменить в формулировке любой из сотен других равносильных задач — и это уже нетривиальный эффект абстрагирования. Поэтому необозримое множество NP-задач вместо одной — хотя и странный, но весьма полезный трюк.

Рис. 1.2

NP-полнота. Среди NP-*задач* есть в некотором роде универсальные, как говорят, — NP-*полные* (рис. 1.2), каковые *полиномиально эквива-*

[13] Иначе говоря, граф не имеет *гамильтонова цикла*, см. раздел 1.4.

[14] В 2001 г. Математический институт Клея (Кембридж, штат Массачусетс) учредил премии по миллиону долларов за решение каждой из семи проблем тысячелетия, в том числе за «$\mathbf{P} \stackrel{?}{=} \mathbf{NP}$».

лентны друг другу. По определению задача NP-*полна* (NP-*complete*), если к ней полиномиально сводится [15] любая другая NP-задача. Поэтому полиномиальное решение любой NP-*полной задачи* полиномиально решает все остальные NP-*задачи*.

Кое-что имеет смысл сказать загодя о понятиях, которые в рассматриваемой области иногда вызывают недопонимание. Например, часто упоминаемые ***массовые задачи***, т. е. задачи общего вида, каковые становятся *индивидуальными* при конкретизации размерности и параметров. За пределами теории алгоритмов подчеркивать такое различие даже в голову не приходит. Фиксация матрицы A переводит массовое неравенство $Ax > 0$ (семейство неравенств) — в индивидуальное. Зачем, казалось бы, фиксировать очевидное? Поэтому новичок иногда теряется в догадках насчет потайных пружин массовости задач. На самом же деле никакого скрытого смысла нет. Есть лишь резон при изучении работы алгоритмов *на семействах задач* делать упор на зависимости времени работы от размера описания задачи. Бывает наоборот, надо уточнить, что речь идет о конкретной, индивидуальной задаче. Либо промежуточный вариант — работа алгоритма в ситуации, когда один параметр фиксирован (например, размерность), другой — свободен (как бы частичная массовость).

1.4. Задачи на графах

Ассортимент комбинаторных задач насчитывает тысячи наименований. При этом некоторое ощущение «игрушечности» возникает из-за выдвижения на передний план упрощенно-сказочных вариантов, о которых удобнее рассказывать. В то же время такого сорта задачи, но в другом обрамлении, весьма широко распространены. Например, *сортировка и поиск* в массивах данных [11] — необозримая тема с выходом на самые что ни на есть практически важные задачи. Расположение информации в компьютерной памяти с целью

[15] Полиномиальную эквивалентность и полиномиальную сводимость на данном этапе мы оставляем в подвешенном состоянии — см. главу 3. Хотя «издалека» здесь все более-менее ясно, речь идет о возможности преобразования одной задачи в другую за полиномиальное время. Но есть и нюансы.

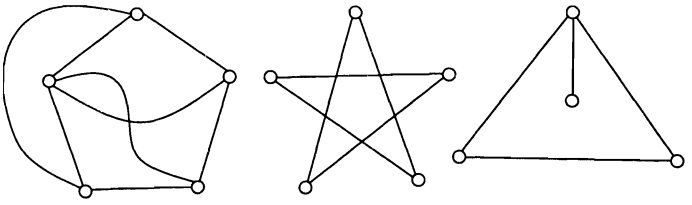


Рис. 1.3

облегчения доступа и скорости счета, составление расписаний, выбор маршрутов связи, расстановка приоритетов, оптимизация дисциплин обслуживания — все это комбинаторные задачи, ежесекундно решаемые *за кадром* в любом работающем компьютере.

Стилизованные постановки выглядят более абстрактно, и довольно часто формулируются как задачи на графах. Предварительно имеет смысл договориться о терминологии.

Множество точек — *вершин* V, — соединенных линиями — дугами, *ребрами* (u, w), где $u, w \in V$, — называют *графом*[16] и обозначают G либо $G(V)$, либо $G(V, E)$, где E обозначает совокупность ребер. Вершины удобно нумеровать, и тогда ребра — это пары целых чисел (i, j). Вершины i и j, соединенные ребром, называют *смежными*, а ребро, выходящее из вершины k (либо входящее)[17] — *инцидентным* вершине k. Граф G однозначно характеризуется *матрицей смежности* $A = [a_{ij}]$, у которой $a_{ij} = 1$, если G содержит дугу (i, j), и $a_{ij} = 0$ — в противном случае. Матрица смежности, разумеется, симметрична (для неориентированного графа).

Последовательность вершин $\{i_1, \ldots, i_n\}$, *соединенных ребрами*, называют *путем*, соединяющим вершины i_1, i_n. Путем, или *цепью*, называют также последовательность ребер

$$(i_1, i_2),\ (i_2, i_3),\ \ldots,\ (i_{n-1}, i_n).$$

Граф, у которого любые две вершины соединены некоторым путем, считается *связным*. Замкнутый путь, начинающийся и заканчивающийся в одной и той же вершине, — это *цикл* (*контур*). У *простого цикла* все ребра различны, а у *элементарного* — нет *подциклов* (циклов, не совпадающих с исходным). *Гамильтонов цикл* — это элементарный цикл, проходящий через все вершины графа. Граф без простых циклов считается *лесом*. Связный лес — *дерево*.

---

[16] О графах можно рассуждать алгебраически — как о заданном подмножестве пар (u, w) из *произведения* $V \times V$. Но это игра с завязанными глазами. Исходный язык графов — наглядные геометрические образы (рис. 1.3), помогающие выбирать пути решения задач.

[17] Различие приобретает смысл в *ориентированных графах*, где «входит» и «выходит» — разные понятия.

Ориентированные графы. Ребра *ориентированного графа* имеют направление, — что на рисунках изображается стрелочками, а при описании пар (i, j) фиксацией порядка. Возможно:

$$(i, j) \in V \times V, \quad \text{но} \quad (j, i) \notin V \times V,$$

хотя не обязательно — в графе могут присутствовать обе дуги (i, j), (j, i).

Кое-что в терминологии для ориентированных графов (*орграфов*) меняется, но логика изменений лежит на поверхности. Под направлением дуги удобно подразумевать допустимое направление движения. Тогда путь в орграфе из i в j — это та же последовательность дуг, по которым можно пройти из i в j, но теперь с учетом разрешенных направлений движения.

Матрица смежности A у *орграфа* — не обязательно симметричная. *Элемент* a_{ij}^k *k-й степени* A^k *равен числу путей в* G, *ведущих из вершины* i *в* j. (**?**) Этот пример говорит о целесообразности выхода в теории графов за пределы чисто геометрических представлений. Впрочем, это и так ясно. Использование разных языков в любой области расширяет кругозор и набор инструментов.

Некоторые понятия будут уточнены по ходу дела. К слову сказать, на терминологии важно не зацикливаться, потому что соблазн оговорить «все» — создает мешанину, в которой никто не хочет разбираться. К тому же некоторый люфт определений обеспечивает гибкость, присущую обыкновенному языку, и создает менее напряженную атмосферу. Скажем, упоминание *реберно-взвешенного* графа без предварительных разъяснений способно навести на мысль, что имеется в виду граф, ребрам которого приписаны веса (числа). Конечно, кто-то скажет, что проще «заранее определить». Но проблема не так проста. Груда мелких определений способна превращаться в монолит, где компоненты уже не видны и никому не интересны. Именно поэтому компьютерные программы стали делать так, чтобы разобраться можно было без инструкций. Подобный стиль имеет свои плюсы и в других областях, в том числе — в математике.

Однако вернемся к нашим баранам. Комбинаторных задач известно много, но среди них есть стержневые, знакомство с которыми необходимо для ориентации в рассматриваемой области.

КОММИВОЯЖЕР (ЗК)

Даны n «городов» (вершин графа) и расстояния r_{ij} (длины ребер) между ними. Требуется найти замкнутый маршрут минимальной длины, заходящий во все города по одному разу. Это оптимизационный вариант *задачи коммивояжера* (**ЗК**). Вариант «распознавания»: *существует ли маршрут, проходящий через все города по одному разу, длина которого,*

$$r_{i_1 i_2} + r_{i_2 i_3} + \ldots + r_{i_n i_1}, \tag{1.2}$$

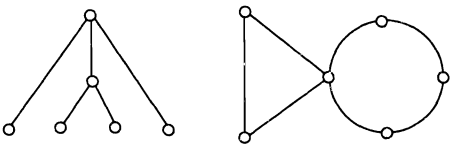
не превосходит r?

Обратим сразу внимание, что «торгово-транспортная» интерпретация не обязательна. В ту же схему укладывается масса других задач. Например, *обработка на станке n деталей*, при условии что r_{ij} — время переналадки станка для обработки j-й детали после i-й. Оптимальная последовательность обработки по критерию минимума суммарного времени переналадок (1.2) — равносильна *задаче коммивояжера.*

КОММИВОЯЖЕР — наиболее знаменитая задача комбинаторной оптимизации, которая долгое время притягивала внимание, стимулировала исследования и вызывала некие «глубинные ощущения». Так или иначе казалось, что ее решение обеспечит фундаментальный прорыв в широком диапазоне дискретного анализа. Смутные предчувствия материализовались впоследствии в теории NP-*полных задач*, о чем речь впереди.

ГАМИЛЬТОНОВ ЦИКЛ

При поиске *на реберно-взвешенном графе гамильтонова цикла минимальной длины* (**ЗК**), — необходимо убедиться, что существует хотя бы один маршрут, обладающий требуемыми свойствами. Граф может оказаться вообще без циклов (рис. 1.4 *а*) или не иметь гамиль-



а) *б)*

Рис. 1.4

тонова цикла (рис. 1.4 *б*). Ситуация, кстати, может быть не столь визуально прозрачна как на рис. 1.4, — и проблема существования гамильтонова цикла, вообще говоря, не менее сложна, чем исходная задача коммивояжера (обе NP-*полны*, см. главу 3).

ИЗОМОРФИЗМ ПОДГРАФУ

Даны два графа G_1, G_2. *Верно ли, что* G_1 *содержит подграф, изоморфный* [18] G_2? При совпадении числа вершин и ребер у G_1 и G_2 задача переходит в **ИЗОМОРФИЗМ ГРАФОВ** — изоморфны ли G_1 и G_2? Пример ИЗОМОРФИЗМА ГРАФОВ изображен на рис. 1.5. Положительный ответ очевиден благодаря геометрической идентичности графов. После «хорошей» деформации граф перестает быть похожим сам на себя.

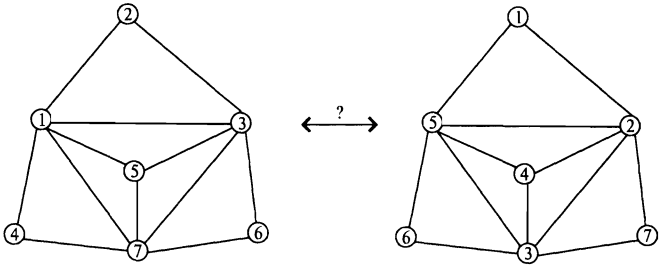


Рис. 1.5

КЛИКА

Существует ли в графе клика, т. е. полный [19] *подграф, с числом вершин не менее* k? На рис. 1.6 клика выделена жирными линиями.

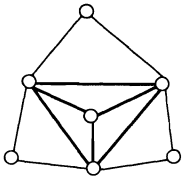


Рис. 1.6

Это вариант «распознавания». В случае «оптимизации» речь идет о нахождении *клики максимального размера.*

Размер максимальной клики называют *плотностью графа.* Иначе говоря, плотность графа — это максимальное число вершин,

[18] То есть совпадающий с G_2 при подходящей перенумерации вершин. *Подграфом* графа G называют любой граф $G' \subset G$, вершины и ребра которого являются вершинами и ребрами графа G.

[19] Граф называется *полным*, если содержит все возможные ребра. *Кликой* часто называют также полный подграф максимального размера.

ВЕРШИННОЕ ПОКРЫТИЕ

Существует ли в графе $G(V, E)$ вершинное покрытие из k вершин? Подмножество $V' \subset V$ называют *вершинным покрытием* графа $G(V, E)$, если каждое ребро G инцидентно некоторой вершине из V'.

КЛИКА, ВЕРШИННОЕ ПОКРЫТИЕ и еще АНТИКЛИКА (подмножество вершин $V' \subset V$, не соединенных в $G(V, E)$ ни одним ребром), — образуют единый комплект в следующем смысле.

1.4.1. Теорема. *Граф G содержит клику размера k в том случае, если дополнение*[20] $\overline{G}$ *имеет антиклику того же размера, либо $\overline{G}$ обладает вершинным покрытием размера $|V| - k$.*

РАСКРАСКА ГРАФА В 3 ЦВЕТА

Можно ли все вершины графа раскрасить в 3 цвета так, чтобы смежные вершины были окрашены в разные цвета? Задача NP-полна и остается NP-полной для графов, все вершины которых имеют степень $\leqslant 4$, а также для *планарных графов*[21].

Если выбор цвета для каждой вершины ограничен двумя цветами, задача решается полиномиально. (**?**)

ПАРОСОЧЕТАНИЕ

Граф G называется *двудольным*, если множество его вершин так распадается на два непересекающихся подмножества V_1 и V_2, что все ребра G соединяют вершины исключительно из разных подмножеств V_1, V_2 (рис. 1.7). Задача состоит в поиске *совершенного паросочетания*, каковым называют взаимно однозначное соответствие между вершинами из V_1 и подмножеством вершин из V_2, при котором соответствующие вершины соединены ребром в G.

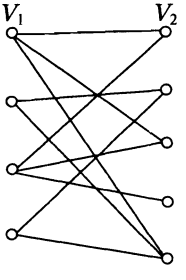


Рис. 1.7

[20] Дополнение $\overline{G}$ имеет то же множество вершин что и G, и содержит те ребра, которых не содержит G.

[21] Граф называют *планарным*, если он может быть нарисован на плоскости без пересечений ребер.

Содержательная привязка задачи имеет широкий диапазон, от распределения проектов V_1 по исполнителям V_2 — ребра G показывают возможности исполнителей, до заключения браков — ребра G показывают, какие женихи V_1 и невесты V_2 подходят друг другу.

Для существования *совершенного паросочетания* необходимо, очевидно, чтобы любые k вершин из V_1 были связаны ребрами в G с $k' \geqslant k$ вершинами из V_2. Это лежащее на поверхности условие является также достаточным (*теорема Холла*), что в определенной степени неожиданно, и не так уж легко доказывается.

Популярен также более общий вариант, известный как *задача о назначениях*, где ребрам приписаны веса, и речь идет о поиске *максимального паросочетания*[22]. Собственно, здесь можно говорить о том же распределении проектов по исполнителям, с тем уточнением, что заданы величины c_{ij}, характеризующие эффективность i-го исполнителя при выполнении j-го проекта.

Задача решается эффективно (за полиномиальное время) так называемым *венгерским методом*, представляющим собой один из вариантов двойственного решения *транспортной задачи* специального вида, где решение, лежащее в вершине допустимого многогранника, автоматически получается целочисленным.

Легкорешаемые задачи, разумеется, представляют интерес, и они будут далее описаны с некоторыми подробностями. Но в данном контексте фокус внимания иной. Акцент делается на тех ситуациях, где перебор не удается кардинально сократить, — по крайней мере, пока. Те же задачи, с которыми удается справиться за полиномиальное время, служат как бы ориентиром. МОД и ПАРОСОЧЕТАНИЕ, конечно, не исчерпывают многообразия *полиномиальных задач*, но составляют его существенную часть, потому что эффективно решаемых задач не так много.

Естественным обобщением ПАРОСОЧЕТАНИЯ является задача ТРЕХМЕРНОЕ СОЧЕТАНИЕ (3-С). *В трехдольном графе*,

[22] Максимальной суммарной эффективности либо минимальной суммарной стоимости.

каждая доля которого содержит n вершин,

$$u \in U, \quad v \in V, \quad w \in W,$$

требуется найти n непересекающихся «троек», каждая из которых содержит по одной вершине из каждой доли. Задача NP-полна.

В оптимизационном варианте каждой «тройке» вершин, содержащей по одной вершине из каждой доли, приписывается вес «тройки», и далее ищутся n непересекающихся «троек», суммарный вес которых максимален.

1.5. Целочисленное программирование

Задача целочисленного программирования

$$f(x) \to \max, \quad g(x) \leqslant 0,$$

получается из обычной задачи на условный экстремум [6, т. 7] разрешением переменным принимать лишь целочисленные значения. Среди таких задач принято выделять определенные частные случаи, несущие на себе следы общих закономерностей.

РЮКЗАК

Имеется n предметов, v_i — стоимость i-го, w_i — его вес. Надо выбрать группу предметов с максимальной суммарной стоимостью при ограниченном суммарном весе, т. е. решить задачу

$$\sum_i v_i x_i \to \max, \quad \sum_i w_i x_i \leqslant W,$$

где x_i может принимать значение 1 *или* 0.

Задача встречается в различных вариациях. В варианте распознавания остаются два неравенства, которые обычно сводят к одному, приравнивая стоимости весам. Неравенство часто заменяют равенством

$$\sum_i w_i x_i = W, \tag{1.3}$$

требуя разрешимости (1.3) либо в целочисленных переменных, либо в (0, 1)-переменных. В первом случае задачу называют **ЦЕЛОЧИСЛЕННЫЙ РЮКЗАК**, во втором — (0, 1)-**РЮКЗАК**. Наконец, частный случай $W = \frac{1}{2}\sum_i w_i$ называют задачей **РАЗБИЕНИЕ**, в силу эквивалентной формулировки: можно ли выделить такое подмножество индексов J, что

$$\sum_{i\in J} w_i = \sum_{i\notin J} w_i.$$

Несмотря на видимую простоту, все перечисленные задачи NP-*полны*.

ЦЕЛОЧИСЛЕННОЕ ЛИНЕЙНОЕ ПРОГРАММИРОВАНИЕ (кратко ЦЛП)

$$\sum_{i=1}^{n} v_i x_i \to \max, \quad \sum_i w_{ij} x_i \leqslant W_j, \quad j = 1, \ldots, m, \tag{1.4}$$

где искомые переменные x_i и коэффициенты v, w, W — принимают целочисленные значения.

Популярна также задача (0, 1)-**ЦЛП**, в которой искомые переменные x_i могут принимать лишь значения 0 или 1. Эта задача является частным случаем ЦЛП, поскольку сводится к форме (1.4) добавлением неравенств $x_i \leqslant 1$, $-x_i \leqslant 0$. Что касается аналогичной взаимосвязи задач ЦЕЛОЧИСЛЕННЫЙ РЮКЗАК и (0, 1)-РЮКЗАК, то ее нет, потому что в постановках «РЮКЗАКОВ» всего одно неравенство, по определению.

Если требование целочисленности x_i снимается, (1.4) трансформируется в обычную задачу **ЛИНЕЙНОЕ ПРОГРАММИРОВАНИЕ (ЛП)**, решение которой лежит в вершине допустимого многогранника

$$\sum_i w_{ij} x_i \leqslant W_j, \quad j = 1, \ldots, m.$$

В схему **ЛП** укладывается

ТРАНСПОРТНАЯ ЗАДАЧА

$$\sum_{i,j} c_{ij}x_{ij} \to \min, \quad \sum_{j} x_{ij} \leqslant a_i, \quad \sum_{i} x_{ij} \geqslant b_j, \qquad (1.5)$$

где переменные $\{x_{ij}\}$ могут быть выстроены в цепочку, заново перенумерованы, — и задача приобретет вид (1.4) [23].

Стандартная легенда формализации (1.5) такова. Имеется m производителей и n потребителей некоторого товара, расположенных в узлах транспортной сети. Если x_{ij} обозначает количество продукта, перевозимого из i-го узла в j-й, a_i — объем производства в i-м узле, b_j — суммарная потребность — в j-м, то (1.5) означает минимизацию общей стоимости перевозок при естественных ограничениях: вывезти из узла нельзя больше, чем там производится; завезти надо не меньше, чем потребляется.

1.6. Логические задачи

Некоторая часть дискретных задач возникает в матлогике [6, т. 6], в связи с чем уместно кое-что напомнить. Логические переменные $x, y, z \ldots$ принимают одно из двух значений $\{1, 0\}$, либо {*истина* (И), *ложь* (Л)}. Для задания логических функций используются *логические связки*:

$^-$(*отрицание «не»*), $\vee$ (*или*), $\wedge$ (*и*), $\to$ (*следует*).

Связку $\vee$ называют *дизъюнкцией*; $\wedge$ — *конъюнкцией*. Функция $x \vee y$ равна 1, если хотя бы одна переменная равна 1, и — нулю в противном случае. Конъюнкция $x \wedge y$ равна 1 только при условии $x = y = 1$, и $x \wedge y = 0$ в остальных случаях.

Переход на знаки «$\cdot$» и «$+$» для обозначения, соответственно, конъюнкции и дизъюнкции имеет определенные психологические преимущества. В булевых переменных $\{1, 0\}$ конъюнкция $\wedge$ — есть в чистом виде обычное умножение $x \cdot y$. Для замены $x \vee y = x + y$ под $x + y$ приходится понимать

$$1 + 1 = 1 + 0 = 0 + 1 = 1, \quad 0 + 0 = 0.$$

Комбинации связок позволяют задавать n-местные *логические функции* $\varphi(x_1, \ldots, x_n)$. Например, $\varphi(x, y, z) = (\bar{x} + y) \cdot \overline{(y \to z)}$.

Любая n-местная логическая функция $\varphi(x_1, \ldots, x_n)$ может быть записана с помощью *конъюнкций, дизъюнкций и отрицаний* либо в *дизъюнктивной нормальной*

[23] Чего не стоит делать, ибо это разрушает структуру, помогающую решению задачи.

форме (ДНФ), т. е. в виде суммы произведений [24] переменных или их отрицаний [25], либо в *конъюнктивной нормальной форме* (КНФ), в которой любая функция представляется в виде произведения сумм переменных или их отрицаний. Например,

$$\varphi(x, y, z) = (\bar{x} + y + z) \cdot (\bar{x} + \bar{y} + z) \cdot (x + \bar{y} + \bar{z}) \cdot (x + y + z) = z.$$

Существование КНФ легко устанавливается. Логическую функцию $\varphi(x_1, \ldots, x_n)$, заданную в вершинах куба $[0, 1] \times \ldots \times [0, 1]$, можно представить в виде произведения

$$\varphi(x_1, \ldots, x_n) = \prod_{k=1} \psi_k(x_1, \ldots, x_n), \tag{1.6}$$

где каждая функция $\psi_k(x_1, \ldots, x_n)$ принимает значение 0 в k-й вершине куба, и значение 1 — в остальных вершинах. Понятно, что $\psi_k(x_1, \ldots, x_n)$ представимы в виде суммы (дизъюнкции) переменных x_i или их отрицаний $\bar{x}_j$. Например, $x_1 + \bar{x}_2 + x_3 = 0$ только в вершине $\{0, 1, 0\}$. Запись (1.6), конечно, не экономна, и число сомножителей может быть значительно уменьшено. Более подробно логическая техника рассматривается в [6, т. 6].

Задача ВЫПОЛНИМОСТЬ (коротко ВЫП или SAT) [26]

Дана n-местная логическая функция $\varphi(x_1, \ldots, x_n)$ в конъюнктивной нормальной форме, т. е. в виде произведения сумм переменных или их отрицаний — конъюнкции дизъюнкций литералов. Например,

$$\varphi(x_1, \ldots, x_n) = (\bar{x}_1 + x_5 + x_8) \cdot (\bar{x}_2 + \bar{x}_4) \cdot (x_2 + \bar{x}_7 + \bar{x}_8).$$

Существует ли набор значений переменных $\{x_1, \ldots, x_n\}$, при котором $\varphi(x)$ выполнима, т. е. $\varphi(x_1, \ldots, x_n) = 1$?

То же самое часто излагают иначе. Имеется m дизъюнкций

$$D_1(x), \ldots, D_m(x).$$

Требуется установить существование набора литералов $x = \{x_1, \ldots, x_n\}$, при котором истинны все [27] $D_j(x)$, т. е. $\forall j : D_j(x) = 1$. От такой интерпретации удобно отталкиваться при формулировке задачи **max-ВЫП**: *найти $x = \{x_1, \ldots, x_n\}$, при котором истинно максимальное число дизъюнкций $D_j(x)$.*

---

[24] Например, $\varphi(x, y, z) = x \cdot y \cdot z + x \cdot \bar{y} \cdot z + \bar{x} \cdot y \cdot \bar{z} = x \cdot z + \bar{x} \cdot y \cdot \bar{z}$.

[25] Переменную или ее отрицание называют *литералом*.

[26] **SAT** — западное обозначение задачи, от satisfiability (англ.).

[27] Разумеется, $\varphi(x) = D_1(x) \ldots D_m(x)$.

Легко проверяется, что любая дизъюнкция *полиномиально* преобразуется в произведение дизъюнкций, содержащих по три *литерала*[28]. Поэтому ВЫПОЛНИМОСТЬ полиномиально эквивалентна задаче 3-ВЫП, в которой все дизъюнкции содержат по три *литерала*.

В теории сложности алгоритмов ВЫПОЛНИМОСТЬ играет роль «главной» задачи, что до некоторой степени является результатом исторической случайности. Просто *Кук* [23], заложивший основы теории NP-полноты, «так распорядился», начав исследование именно из этого пункта. Надо признать, однако, что для такого выбора были естественные основания, на которых пока нет смысла останавливаться. Но вот вариации ВЫПОЛНИМОСТИ заслуживают внимания, поскольку мыслить бывает удобнее в других категориях.

Задание n-местной логической функции $\varphi(x_1, \ldots, x_n)$ равносильно указанию вершин единичного куба $[0, 1]^n$, в которых $\varphi = 1$, и вершин — в которых $\varphi = 0$. Иначе говоря, для задания $\varphi(x)$ множество вершин куба $[0, 1]^n$ необходимо разбить на два подмножества, для чего достаточно перечислить элементы одного из подмножеств. О вершинах можно говорить, так же как о $(0, 1)$-последовательностях, или двоичных числах. Например,

$$\{0, 1, 1, 0, 1, 0, 0, 0, 1\} \quad \Leftrightarrow \quad 0\,1\,1\,0\,1\,0\,0\,0\,1 = 2^7 + 2^6 + 2^4 + 1.$$

Бесхитростное перечисление вершин (чисел), само собой, неэкономно. Экономии позволяет добиться следующий трюк. В n-разрядном двоичном числе фиксируются цифры в каких-то k разрядах, остальные — произвольны. Это сразу выделяет 2^{n-k} чисел[29]. Например, одна строчка вида

$$*\,1\,*\,0\,*\,*\,*\,*\,1,$$

где $*$ можно заменять на любую цифру 1 или 0, — сразу охватывает 2^{9-3} вершин (двоичных чисел), а несколько строчек типа

$$\begin{array}{l} 0\,*\,*\,0\,*\,1, \\ *\,1\,0\,1\,*\,*, \\ *\,1\,1\,1\,0\,*, \\ 1\,0\,0\,*\,*\,*, \\ *\,*\,*\,*\,1\,0, \\ 0\,*\,*\,0\,*\,0, \\ *\,0\,0\,0\,*\,0, \end{array} \tag{1.7}$$

[28] Пока мы обходимся без доказательств.

[29] Что равносильно выделению $(n-k)$-мерной грани куба $[0, 1]^n$.

при небольшой длине описания покрывают большое множество двоичных чисел (вершин куба, (0, 1)-последовательностей), но посчитать сколько — трудно, иначе NP-задачи легко решались бы.

Легко видеть, что строчки (1.7) взаимно однозначно описывают дизъюнкции по типу

$$0 * * 0 * 1 \quad \Leftrightarrow \quad x_1 + x_4 + \bar{x}_6.$$

ИСЧЕРПАНИЕ ВЕРШИН КУБА (ИВК)

Дана совокупность строчек длины n вида (1.7). *Покрывают ли они в указанном выше смысле все вершины куба*[30] $[0, 1]^n$?

Дополнение ИВК, сводящееся к выяснению существования на кубе непокрытой вершины[31], — является эквивалентом задачи ВЫПОЛНИМОСТЬ.

Принадлежность дополнения ИВК классу NP очевидна. Если непокрытая вершина есть, ее можно предъявить и легко проверить. Принадлежность NP самóй задачи ИВК — разумеется, вопрос открытый.

1.7. (0, 1)-матрицы

Многие задачи на графах удобно формулировать с помощью *матриц смежности*. КЛИКА, например, есть вопрос *о существовании у симметричной* (0, 1)*-матрицы A главной подматрицы*[32] *размера $k \times k$, у которой все внедиагональные элементы равны* 1. Положительный ответ означает возможность приведения одинаковой перестановкой строк и столбцов матрицы A к блочному виду

$$\begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix},$$

[30] Все двоичные числа или все (0, 1)-последовательности длины n.

[31] Неисчерпанной (0, 1)-последовательности, отличающейся хотя бы в одном разряде от любой строчки описания вида (1.7).

[32] *Подматрицей* матрицы A называют матрицу, элементы которой стоят на пересечении выбранных строк и выбранных столбцов матрицы A. Если номера строк и столбцов одинаковы — подматрицу называют *главной*.

где у квадратной подматрицы A_{11} все внедиагональные элементы $a_{ij} = 1$.

Так же легко трансформируется ИЗОМОРФИЗМ ПОДГРАФУ. *Даны две* (0, 1)*-матрицы* A_1, A_2. *Верно ли, что* A_1 *содержит подматрицу, перестановочно подобную*[33] A_2? При совпадении размерностей A_1 и A_2 задача переходит в эквивалент ИЗОМОРФИЗМА ГРАФОВ.

Что касается ГАМИЛЬТОНОВА ЦИКЛА, то здесь не все так просто. Воспользуемся некоторыми понятиями из линейной алгебры [6, т. 3]. Положительная матрица A называется *неразложимой*, если одинаковой перестановкой строк и столбцов она не может быть приведена к виду

$$\begin{bmatrix} A_{11} & A_{12} \\ 0 & A_{22} \end{bmatrix},$$

где A_{11} и A_{22} квадратные матрицы. Иными словами, матрица $A = [a_{ik}]$ неразложима, если не существует такого подмножества индексов J, что $a_{ik} = 0$ для всех $i \in J$, $k \notin J$.

Неразложимость $A = [a_{ij}]$ равносильна существованию *дорожки невырожденности*, каковой называют последовательность ненулевых элементов

$$a_{i_1 i_2}, a_{i_2 i_3}, \ldots, a_{i_k i_1},$$

где среди индексов $i_1, i_2, \ldots, i_k$ имеются все числа $1, 2, \ldots, n$. Вопрос о существовании у (0, 1)-матрицы дорожки невырожденности, в которой все индексы $i_1, i_2, \ldots, i_k$ разные, — равносилен ГАМИЛЬТОНОВУ ЦИКЛУ[34].

Если бы не последняя оговорка (насчет неповторяющихся индексов), задача (о существовании циклического маршрута, заходящего во все города, но не обязательно по одному разу, как на рис. 1.4 *б*) решалась бы проверкой неравенства $(I + A)^{n-1} > 0$, влекущего за собой неразложимость $A \geqslant 0$ [6, т. 3]. Причем, *если*

[33] То есть «подматрица» одинаковой перестановкой строк и столбцов переводится в A_2.

[34] Можно было бы сказать еще иначе. Граф имеет гамильтонов цикл, если его матрица смежности после обнуления некоторых элементов совпадает с матрицей циклической перестановки. Напомним, *матрицей перестановки* называют матрицу, у которой в каждом столбце и каждой строке — ровно один элемент равен 1, остальные нули. Умножение на такую матрицу слева — переставляет строки, справа — столбцы. Например,

$$P = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix} \quad \Rightarrow \quad P \begin{bmatrix} a_{1:} \\ a_{2:} \\ a_{3:} \end{bmatrix} = \begin{bmatrix} a_{1:} \\ a_{3:} \\ a_{2:} \end{bmatrix}.$$

Матрица циклической перестановки не имеет единиц на диагонали.

матрица $A \geqslant 0$ *неразложима, то спектральный радиус* $\rho(A) > 0$ *является ведущим собственным значением* A *алгебраической кратности* 1, *которому отвечает строго положительный собственный вектор. Других положительных собственных значений и векторов у* A *нет.* Пример свидетельствует о наличии связей между комбинаторными и спектральными свойствами матриц.

1.8. Простые и составные числа

Задача **СОСТАВНОЕ ЧИСЛО (СЧ)** заключается в выяснении возможности *факторизации*[35] целого N. Существуют ли такие $N_1 > 1$ и $N_2 > 1$, что $N = N_1 N_2$? Дополнительная задача **ПРОСТОЕ ЧИСЛО (ПЧ)** состоит в выяснении простоты N.

Обе задачи принадлежат классу P, но это стало ясно лишь в результате титанических усилий, см. главу 6. Однако даже сейчас, когда проблема вроде бы решена, СЧ остается белой вороной в ряду комбинаторных задач. Дело заключается в том, что СЧ полиномиальна как задача распознавания, но остается труднорешаемой как задача *факторизации*. Другими словами, вопрос: «число N простое или составное?» — может быть решен за полиномиальное[36] число шагов. Но если N окажется составным, $N = N_1 N_2$, то нахождение N_1 и N_2 требует (пока) экспоненциального по $\lfloor \log N \rfloor$ счета. Иначе говоря, задача полиномиальна, но поиск *удостоверения* фактически требует перебора, что для P-класса нехарактерно.

Выбросить неприятность из головы и забыть здесь не удается, потому что на факторизации чисел свет в некотором роде сошелся клином, ибо на тяжести поиска злополучных сомножителей стоит вся современная криптография, над которой дамокловым мечом нависает перспектива создания *квантового компьютера*, см. главу 10.

Однако специфика на этом не заканчивается. Неожиданной особенностью задачи ПЧ оказывается непривычная разновидность *удостоверения*. Вместо обычного «того, что ищется» здесь *удостоверением* является доказательство полиномиальной длины. Собственно, до того как полиномиальность была установлена, под вопросом

[35] Разложения на множители.

[36] Относительно длины описания N, т. е. относительно $\lfloor \log N \rfloor$.

была даже принадлежность ПЧ классу NP, потому что не ясно было, что можно предъявить в качестве удостоверения, кроме перебора всех вариантов. Однако ПЧ $\in$ NP было установлено [37] до того, как выяснилась полиномиальность ПЧ [21].

Конечно, проблемы с удостоверением для ПЧ не так уж нежданны-негаданны. Подобные осложнения характерны для всех задач из co-NP, в том числе для задач дополнительных к P-задачам. Другое дело что к дополнению МОД (имеется в виду вариант распознавания МОД) удостоверение конструируется более-менее

Еще одна особенность ПЧ — низкая эффективность полиномиального алгоритма, из-за чего практические вычисления в данном случае ориентируются на *вероятностные алгоритмы* (глава 8) и надеются на — *квантовые* (глава 10).

1.9. Комбинаторные механизмы

Вся Вселенная сделана из сотни химических элементов. А с помощью короткого алфавита написаны и Библия, и Уголовный кодекс. Такова сила комбинаторики, плодоносящая разнообразием. В теории алгоритмов это создает трудности.

Схематично решаемые задачи состоят в поиске оптимума на некотором дискретном множестве X. Либо в проверке существования варианта $x \in X$, обладающего неким данным свойством. Проблема заключается в том, что элементы X задаются как «*молекулы из комбинаторных атомов*» [38]. И если X с самого начала непосредственно описано — это не комбинаторная задача.

Скажем, X — подмножество вершин куба $[0, 1]^n$, в которых логическая функция $\varphi(x_1, \ldots, x_n)$ равна 1, причем $\varphi = 0$ в остальных вершинах. Элементы X можно дать списком — и тогда задача перестает быть комбинаторной, и даже перестает быть «задачей», поскольку исчезает изюминка. В нормальное русло ситуацию возвращает использование последовательностей типа (1.7), каковые служат как раз «комбинаторными атомами», с помощью которых экспоненциальный список X экономно упаковывается. Иначе говоря, в задаче ВЫП «комбинаторными атомами» служат дизъюнкции.

---

[37] *Pratt V.* Every prime has a succinct certificate // SIAM J. Comp. 1975. **4**. 214–220.

[38] Как комбинации некоторого «алфавита» $\mathcal{A}$.

В ЛИНЕЙНОМ ПРОГРАММИРОВАНИИ аналогичный трюк состоит в использовании линейных неравенств $Ax \leqslant b$. Игра идет на вершинах допустимого многогранника, но сам многогранник описывается с помощью «комбинаторных атомов» $\sum_i a_{ij} x_i \leqslant b_j$.

Как ни банально сказанное, но именно «стенографические секреты» описания условия задачи оказываются тем внутренним механизмом, без понимания которого невозможно эффективное решение [39]. В ряде случаев, если не всегда, имеет смысл начинать с реорганизации данных и переосмысливания условия.

Есть также другая сторона медали. Множества комбинаций бывают надуманы. Комбинаторика, так сказать, как фантазия исследователя. Задачу сортировки числового массива, например, трудно считать комбинаторной. Океан перестановок, конечно, можно притянуть за уши, чтобы потом гордиться найденной иголкой в стоге сена. Но эти перестановки играют роль не более чем декораций. На дифференцирование тоже можно смотреть как на выбор подходящей функции $f'(x)$ из континуума возможных.

Другой пример — ЛИНЕЙНОЕ ПРОГРАММИРОВАНИЕ. Экспоненциальное множество вершин допустимого многогранника естественно возникает при геометрическом осмысливании задачи, и сохраняет свое значение при рассмотрении симплекс-метода. Но для *метода эллипсоидов* (глава 5) соответствующее множество теряет смысл, не представляет интереса. И разговоры о том, что метод ловко справляется с лавиной вариантов, — не более чем пиар-акция. Дескать, на задачу можно так посмотреть, что эффективность просто удивительна. Но на задачу в контексте *метода эллипсоидов* более естественно смотреть с другой стороны, где комбинаторики просто нет. Хотя, конечно, жалко. Да и не вся правда сосредоточена на другой стороне медали.

[39] Перебор — не что иное, как бессилие перед головоломкой.

Глава 2

Инструменты и декорации

Успех детективного расследования часто зависит от широты взгляда, при котором в поле зрения попадают «не относящиеся к делу» обстоятельства.

Благороднее застрелиться,
чем паразитировать на тонкостях.

В главе приводится краткая информация о вычислимых функциях, и вообще об «алгоритмической идеологии», что для дальнейшего чтения не имеет принципиального значения в техническом отношении. Но психологически, понятийно, и даже философски, — это важная отправная точка. Более подробно тема излагается в [6, т. 6] — ниже дается выжимка, чтобы в процессе драки не отвлекаться на изучение айкидо.

2.1. Вычислимые функции

Целочисленную функцию $f(n)$ целочисленного аргумента [1] называют *вычислимой*, если существует *алгоритм*, вычисляющий значения $f(n)$, если они (значения) определены.

Процедура вычислений отождествляется с программой работы компьютера, представляющей собой запись алгоритма на некотором языке в некотором алфавите $\mathbb{A}$,

$$P = a_1 a_2 \dots a_N, \quad \textit{все} \quad a_j \in \mathbb{A}. \tag{2.1}$$

Под a_j в (2.1) чаще подразумевают операторы [2] из набора $\mathbb{A}$.

---

[1] Аргумент может быть векторным, $n = \{n_1, \dots, n_k\}$. Функцию $f(n_1, \dots, n_k)$ в этом случае называют *k-местной. Включение запятых в алфавит с последующей перекодировкой цифрами — позволяет все функции считать одноместными.*

[2] Цикла, присвоения, перехода и проч.

Программа применяется к различным *входным словам*. Если входные и выходные данные кодируются числами, программа определяет *вычислимую функцию*. С точностью до педантизма к такому определению сводятся все подходы.

Запись программы вычислений в виде (2.1) позволяет *все вычислимые функции перенумеровать*,

$$f_1(n), \ldots, f_k(n), \ldots . \tag{2.2}$$

Сначала перечисляются все программы, состоящие из одного оператора (буквы), потом из двух, потом из трех и т. д.

Почти все разговоры о вычислимых функциях сопровождаются определенной путаницей, связанной с двойственностью функции и программы, ее вычисляющей[3]. В большинстве случаев речь идет о программах, как например в (2.2). Нумеруются таким образом алгоритмы, но изгнать функции из теории неудобно, иначе потеряется возможность в простых ситуациях типа $f(n) = n^n$ обходиться без упоминания конкретных программ. Поэтому о вычислимой функции говорят как о функции в обычном понимании, но за кадром подразумевается наличие алгоритма. В результате каждая функция получает бесконечное количество номеров в перечислении (2.2).

Упразднить функции, оставив лишь алгоритмы, нельзя и по более серьезным причинам. Скажем, *невычислимая функция*[4]

$$h(n) = \begin{cases} f_n(n) + 1, & \textit{если значение } f_n(n) \textit{ определено}, \\ 0, & \textit{если значение } f_n(n) \textit{ не определено}, \end{cases} \tag{2.3}$$

это уже не программа, хотя выглядит «как». Таким образом, в результате (2.3) возникает нечто, имеющее «описание», но не поддающееся алгоритмизации. Обсуждение противоречий увело бы здесь слишком далеко (см. [6, т. 6]).

Отмеченная аномалия, несмотря на миниатюрность, имеет фундаментальный характер и много лиц. Каждый маленький поворот порождает новую истину.

---

[3] Той же размытостью страдает данное в начале раздела определение вычислимой функции.

[4] В предположении противного $h(n) = f_p(n)$ при некотором p. Но этого не может быть, так как $h(p) \neq f_p(p)$, если значение $f_p(p)$ определено, и $h(p)$ определено, если $f_p(p)$ не определено.

Скажем, идея избавления от неопределенных функций по той же причине оказывается несостоятельной. Никакие меры не могут предотвратить зацикливание некоторых программ на некоторых входах, — и любые усилия в этом направлении бессмысленны. Допустим, например, что к перечислению (2.2) тем или иным способом удалось допустить только всюду определенные функции[5] f_n. Тогда функция

$$g(n) = f_n(n) + 1$$

не вычислима, так как при $n = 1$ отличается от f_1, при $n = 2$ — от f_2, и так далее. С другой стороны, она вычислима в любом разумном смысле, поскольку к вычисляемому значению $f_n(n)$ всегда можно прибавить 1. Если же какие-то функции в перечислении (2.2) определены не при всех n, конструкция $g(n) = f_n(n) + 1$ к противоречию не приводит.

Существование невычислимых функций демонстрирует, как далеко может заходить труднорешаемость. Конечно, тема лучше чувствуется в содержательных разветвлениях типа *проблемы останова* или *тождества слов*. Интуитивно значимо выглядят также *теоремы Гёделя*, несмотря на свою чрезвычайную общность. Обо всем этом относительно подробная информация имеется в [6, т. 6].

2.2. Перечислимые множества

Одна из стандартных вычислительных задач — перечисление элементов множества, каковое считается *перечислимым*, если существует эффективная[6] процедура порождения его элементов. Множество *разрешимо*, если существует эффективная процедура для выяснения принадлежности любого n этому множеству.

Вот более конструктивное определение.

2.2.1. Определение. *Множество X перечислимо, если представляет собой область значений либо область определения вычислимой функции.*

[5] То есть программы, останавливающиеся на любом входе n.

[6] Характеристика «эффективная» в данном случае подразумевает существование алгоритма, дающего результат при любом n. Эффективность в контексте NP-проблематики означает полиномиальность счета.

◀ Это не сразу увязывается с предыдущим. Если X — область значений $f(n)$, то как порождать его элементы? Процедура «зависнет» на первом же n, при котором значение $f(n)$ не определено. Выход из положения состоит в следующем. Пусть $P(n, m)$ обозначает реализацию m шагов работы программы по вычислению значения $f(n)$. Пары чисел (n, m) упорядочиваются стандартным образом, после чего программе дается возможность последовательно работать по m шагов, вычисляя $f(n)$, в избранном порядке. Понятно, все значения $f(n)$ будут рано или поздно перечислены.

Кроме того, определение содержит в себе теорему: *область значений любой вычислимой функции всегда есть область определения другой вычислимой функции, и наоборот*. Устанавливается это с помощью той же процедуры $P(n, m)$, вычисляющей $f(n)$ на протяжении m шагов. Если вычисление $P(n, m)$ приводит к определенному результату, n объявляется значением функции $g(k)$, где k — номер пары (n, m). В результате область определения f становится областью значений g. Обратный трюк еще проще. ▶

2.2.2. Теорема Поста. *Для разрешимости X необходимо и достаточно, чтобы X и его дополнение $\overline{X}$ были перечислимы.*

◀ *Необходимость*. Если программа P определяет, принадлежит n множеству X или нет, то ее последовательная работа на $n = 0, 1, 2, \ldots$ разбивает натуральный ряд на два списка X и $\overline{X}$.

Достаточность. Если P перечисляет X, а Q — $\overline{X}$, то попеременная работа программ P и Q рано или поздно любое n внесет в один из списков X или $\overline{X}$, что дает разрешающий алгоритм. ▶

2.2.3. Теорема. *Существует перечислимое, но неразрешимое множество положительных целых чисел*[7].

◀ Пусть $S_1, S_2, \ldots$ — эффективное перечисление всех перечислимых множеств[8]. Заметим, если бы все S_n были разрешимы, то и все дополнения $\overline{S}_n$ входили бы в перечисление $S_1, S_2, \ldots$ (теорема 2.2.2).

Образуем множество D из тех номеров n, которые принадлежат S_n. Таким образом,

$$n \in D \cap S_n \quad \Leftrightarrow \quad n \in D \cup S_n,$$

откуда следует невозможность[9] $D \cap S_n = \varnothing$ и в то же время $D \cup S_n = \mathbb{N}$. Это означает, что D не совпадает ни с одним $\overline{S}_n$, т. е. не перечислимо, а значит (теорема 2.2.2) и неразрешимо. ▶

---

[7] Для вычислимости $f(x)$ необходима и достаточна перечислимость графика, т. е. множества пар $\{x, f(x)\}$.

[8] Существование перечисления перечислимых множеств следует из наличия перечисления (2.2) вычислимых функций, области значений которых и являются множествами $S_1, S_2, \ldots$.

[9] Непустота D при любой организации перечисления $\{S_n\}$ следует хотя бы из того, что $\{S_q = \mathbb{N}\}$ при каком-то q.

После определенной дозы алкоголя P- и NP-задачи начинают ассоциироваться с разрешимыми и перечислимыми множествами. И тогда теорема 2.2.3 подталкивает к заключению $P \neq NP$, а теорема 2.2.2 — к равенству $P = co\text{-}NP \cap NP$.

2.3. Неразрешимые проблемы

Существование невычислимых функций, теорема 2.2.3 и многие другие неразрешимые проблемы алгоритмического характера — имеют общий корень, питающий в том числе *теоремы Гёделя* [10]:

- *Какова бы ни была совокупность аксиом в арифметике, если она непротиворечива, существует такое утверждение A, что ни A, ни его отрицание ($\neg A$) — не доказуемы.*
- *Если непротиворечивая теория T содержит в себе арифметику, то непротиворечивость T недоказуема в T.*

Вариации и короткие доказательства есть в [6, т. 6]. В данном контексте теоремы Гёделя интересны тем, что обращают внимание на универсальную значимость алгоритмического подхода. Не только распознавание образов, игры и прочие интеллектуальные штучки — кодируются и вычисляются, — но и математические изыскания. На доказательства теорем, в частности, можно смотреть как на комбинаторные задачи. Из аксиом и правил вывода требуется построить цепочку, ведущую от предположений к утверждению теоремы. При ограничении длины цепочек получается типичная NP-задача. Если доказательство (цепочка) найдено, оно легко проверяется. Но как найти? Обязателен ли экспоненциальный перебор?

Шоковая составляющая первой теоремы Гёделя состояла в том, что при снятии «ограничения на длину цепочек» успех поиска доказательства или опровержения вообще не гарантирован. Думалось-то наоборот, долго ли коротко ли, но при конечном наборе строительных кубиков (аксиом и правил) результат в принципе обеспечен. А вышло по-другому. Теорем оказалось больше, чем доказательств. Точнее говоря, перечислимое множество «концов цепочек» оказалось неразрешимо.

[10] Перевернувшие в свое время математическое мировоззрение.

Для расширения горизонтов приведем некоторые дополнительные результаты [6, т. 6].

2.3.1. Теорема. *Какова бы ни была непротиворечивая теория, содержащая арифметику, существует не имеющий положительных корней полином* $Q(x_1, \ldots, x_k)$, *отсутствие у которого целых положительных корней недоказуемо.*

2.3.2. Теорема. *Множество* $\mathcal{P}$ *неразрешимых*[11] *полиномов* $P(x)$ *— неперечислимо (тем более, неразрешимо).*

Может показаться, что арифметику могло бы «спасти» расширение аксиоматики. Увы.

2.3.3. Теорема. *Арифметика неаксиоматизируема, даже при включении в систему бесконечного, но перечислимо задаваемого множества аксиом.*

2.3.4. Теорема. *Существуют системы аксиом, «порочность» которых принципиально нельзя обнаружить.*

Неразрешимых алгоритмических проблем слишком много, но все они вертятся вокруг одной Космической воронки. Увидеть в разнообразии единство помогает абстрагирование от специфики. Вот один из таких результатов.

2.3.5. Теорема Райса. *Любое нетривиальное свойство вычислимых функций алгоритмически неразрешимо*[12].

Алгоритмически неразрешимыми оказываются, в частности, *конечность или бесконечность множества значений* $f(x)$, *периодичность*, *ограниченность*, *порядок роста*. Особое значение для дальнейшего будет иметь *неразрешимость проблемы существования полиномиального алгоритма*, вычисляющего $f(x)$.

2.4. Машины Тьюринга

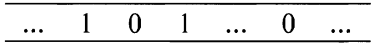
Изучение алгоритмической вычислимости и разрешимости исторически начиналось с машин Тьюринга, которые до сих пор играют роль общепринятого инструмента в теории алгоритмов.

---

[11] Полином считается *неразрешимым* в случае неразрешимости диофантова уравнения $P(x) = 0$, $x = \{x_1, \ldots, x_n\}$.

[12] Свойство считается нетривиальным, если имеются функции, обладающие этим свойством и — не обладающие.

Машина Тьюринга представляет собой конечный автомат, способный читать и писать на бесконечной ленте. Точнее говоря, лента разбита на ячейки, приспособленные для записи букв из некоторого алфавита $A = \{a_1, \ldots, a_m\}$, например, из $\{0, 1\}$:



Время дискретно. В каждом такте автомат обозревает содержимое текущей ячейки, стирает и записывает в эту ячейку новую букву, после чего переходит к соседней ячейке, слева или справа, и меняет свое внутреннее состояние.

Формализованно процесс выглядит так: t — время, $i(t)$ — номер обозреваемой ячейки в момент t, $x(t)$ — входной сигнал автомата (буква, которую он читает в момент t), $x(t+1)$ — выходной сигнал (буква, которую автомат записывает в ячейку вместо $x(t)$), $q(t)$ — внутреннее состояние автомата, число которых предполагается конечным, наконец, $d(t)$ — направление сдвига головки.

Машина M характеризуется тремя функциями F, G и H, определяющими динамику «вычислений»,

$$\begin{cases} x(t+1) = F[x(t), q(t)], \\ q(t+1) = G[x(t), q(t)], \\ i(t+1) = i(t) + d(t) = i(t) + H[x(t), q(t)], \end{cases} \tag{2.4}$$

которые показывают: F — «что в ячейку писать», G — «в какое состояние переходить», H — «куда сдвигаться, влево или вправо». При этом среди внутренних состояний машины $q_1, \ldots, q_n$ выделяется начальное — q_1, в котором машина приступает к работе, и конечное — q_n, попадая в которое, машина останавливается. Если речь идет о задаче распознавания, в состоянии q_n может быть предусмотрена печать символа «да».

В каждом такте работу машины можно описывать также с помощью возможных переходов

$$q_i a_j \to q_k a_l L \quad \textit{или} \quad q_i a_j \to q_k a_l R, \tag{2.5}$$

что равносильно (2.4). В соответствии с (2.5) машина, находясь в состоянии q_i, если видит перед собой a_j, — стирает a_j, пишет a_l, переходит в состояние q_k

и сдвигается влево (Left) или вправо (Right). Список (2.5), задающий сразу все функции F, G и H, может оформляться в виде таблицы, имеющей строки

| q_i | a_j | a_l | , | q_i | a_j | q_k | , | q_i | a_j | L *или* R | .

Богатые возможности машины Тьюринга устанавливаются достаточно просто. Сначала конструируются машины, выполняющие простейшие арифметические и логические операции. Затем выясняется возможность работы таких машин в комбинации друг с другом, что порождает более сложные машины. Те, в комбинации друг с другом, порождают еще более сложные машины. И так далее. Так или иначе, общими усилиями было осознано, что для любого алгоритма, приходящего в голову, существует реализующая машина. Но *тезис Тьюринга*[13]: «*Любой алгоритм, являющийся таковым с интуитивной точки зрения, реализуем машиной Тьюринга*», конечно, может быть справедлив только в «религиозном смысле», поскольку *интуитивное* понимание алгоритма — явление внематематическое.

Универсальная машина. Машины Тьюринга — в силу конечности описания каждой — можно *эффективно перечислить*,

$$M_1, \ldots, M_n, \ldots .$$

Восстановление функций F, G и H по номеру n оказывается при этом достаточно простой операцией, выполнимой некоторой машиной U, на которую — после восстановления M_n — может быть возложена имитация работы M_n на любом входном слове k. В этом случае U называется *универсальной машиной Тьюринга*,

$$U(n, k) = M_n(k).$$

Универсальной машиной подобного типа является современный компьютер, где индивидуальный подход к задачам обеспечивается разными программами. Правила изменения внутренних состояний могли бы, конечно, быть запаяны внутрь конструкции, — но тогда для каждой задачи требовался бы свой компьютер. В универсальном варианте перестройка под задачу производится вводом информации извне. Похожая картина наблюдается в ситуации с универсальной

[13] Используемый, когда возникает необходимость утверждать существование машины Тьюринга для функции, алгоритмическая вычислимость которой ясна из «посторонних» соображений, проистекающих из опыта.

машиной Тьюринга. Ее подробное описание [15], несмотря на принципиальную ясность вопроса, имеет определенный смысл, поскольку некоторые детали выглядят проблематично. Например, может ли U, имея фиксированное число внутренних состояний, моделировать работу машин Тьюринга M_n, имеющих большее число состояний? Может. Нехватка состояний компенсируется записями в памяти (на ленте), что иллюстрируется примером обычного компьютера.

Примеры и замечания

Машины легче конструировать при использовании *единичного кода*, в котором число N записывается в виде последовательности N единиц с возможными пропусками — пустыми ячейками, $\varnothing$. Числа отделяются друг от друга оговоренным знаком, например, $*$.

• Для сложения двух чисел N_1 и N_2 достаточно, чтобы машина убрала вторую звездочку в *машинном слове*

$$\underbrace{*\,1\,\varnothing\,1\,\ldots\,1}_{N_1\ \text{единиц}} * \underbrace{1\,1\,\ldots\,\varnothing\,1}_{N_2\ \text{единиц}}.$$

Задача решается следующим образом. Пусть M начинает работу с левой единицы, и в ее списке (2.5) присутствуют операции

$$q_1 1 \to q_1 1 R, \quad q_1 \varnothing \to q_1 \varnothing R.$$

Тогда M будет двигаться вправо, ничего не меняя [14], пока не дойдет до $*$. Здесь в ее списке (2.5) должна присутствовать команда $q_1 * \to q_2 \varnothing L$, приводящая к стиранию звездочки и переходу в новое состояние q_2, которое может быть заключительным. В модернизированном варианте можно организовать возврат головки в исходное положение (к левой единице)

$$q_2 1 \to q_2 1 L, \quad q_2 \varnothing \to q_2 \varnothing L, \quad q_2 * \to q_3 * R,$$

где q_3 — конечное состояние машины в новом исполнении.

• **Преобразование единичного кода в двоичный.** Для определения четности числа N машине требуются два состояния, которые бы она попеременно меняла после чтения каждой единицы, двигаясь все время вправо. Наткнувшись на обозначение конца N, она бы печатала 0 или 1 (после $*$) — в зависимости от состояния.

Для перевода N из единичного кода в двоичный — описанный счетчик необходимо усложнить. Например, так. Сначала пересчитываются N единиц и пишется 0 при четном N и 1 — при нечетном. Это дает последний разряд двоичного кода N. При пересчете каждая вторая единица стирается, а также стирается последняя единица, если после нее нет «второй». Оставшиеся единицы

[14] Поскольку все время стирает единицу и снова ее записывает.

в количестве [15] $[N/2]$ пересчитываются аналогичным образом, что дает следующий разряд двоичного кода N. И так далее — до исчерпания всех исходных единиц.

Утрясти детали не так сложно, располагая несколькими дополнительными состояниями. Принципиального решения требуют два вопроса. Стирание четных единиц и сдвиг вправо уже полученных разрядов двоичного кода N перед получением следующего разряда (для освобождения места).

• Решение только что упомянутых вопросов значительно упрощает следующее общее соображение. Если задача распадается на две подзадачи, которые надо решить последовательно, то машины Тьюринга M_1 и M_2 для подзадач можно конструировать независимо, и включать M_2 после завершения работы M_1. Это выглядит как связка двух машин, но может рассматриваться как единая машина Тьюринга при отождествлении конечного состояния M_1 и начального M_2, в результате чего возникают единые таблицы функций F, G и H, характеризующие работу блок-схемы в целом. Следовательно, композиция машин Тьюринга — снова машина Тьюринга. Такого рода консолидация в единую машину возможна и при сложном ветвлении алгоритмов в зависимости, например, от последней напечатанной цифры.

• **Поиск информации** [16]. Допустим, информация на ленте записана в виде слов W_j и признаков S_j (*адресная память*). Пары S_jW_j могут быть отделены друг от друга двумя звездочками, а признаки от слов — одной. Где-то на ленте записан желаемый признак S. Машина ищет сначала ближайшую пару S_jW_j, потом — следующую, и каждый раз челночными проходами осуществляет поразрядное сравнение S и S_j.

• **Многомерные состояния.** Конструируя машину Тьюринга, удобно рассуждать, имея в распоряжении *разнородные* состояния. В следующем, например, исполнении. Имеется набор k *опций*, которые можно включать и выключать. Включена опция q_5 — головка движется влево, q_2 — стирание, q_8 — заполнение нулями, q_3 — поиск информации и т. д.

В результате совокупное внутреннее состояние будет характеризоваться двоичным числом 10 ... 0101 (1 в j-м разряде — опция q_j включена, 0 — выключена).

Другое удобное послабление: добавление в алфавит новых символов по мере размышлений. Этим допустимо пользоваться, поскольку любой алфавит впоследствии всегда можно перевести (опять же машиной Тьюринга) в двоичный код.

• Иногда возникает затруднение при переходе от алфавита $\{1, 0, *\}$ — к чисто двоичному. Трудность — психологическая. Думается, что кодировать надо лишь

[15] Квадратные скобки обозначают целую часть числа.

[16] Блок-схемы алгоритмов описывают работу системы *по управлению* — к работе какого блока переходить. При этом необходимо указать, где и как искать информацию.

звездочку, — и тогда все перемешается. Кодировать надо все. Например,

$$0 \leftrightarrow 00, \quad 1 \leftrightarrow 11, \quad * \leftrightarrow 01.$$

Лента в результате становится «блочной» — ячейки объединяются в пары.

2.5. Полиномиальные алгоритмы

Теория сложности алгоритмов имеет дело с задачами, в которых «вычислимость» не играет зримой роли. Неприятностей, однако, и так хватает. Длина вычислений как естественная мера трудоемкости алгоритмов — обнаруживает фундаментальную неуловимость. Любой счет можно ускорить. Факт, конечно, тривиальный. Какова бы ни была машина Тьюринга, существует другая машина с бóльшим алфавитом, которая кодирует группы символов одним символом, за счет чего быстрее считает. Разумеется, с точностью до оговорок. На этом пути есть ограничения. Существуют функции $f(x)$, длина вычисления $N(x)$ которых не может быть сделана меньше $\sqrt{N(x)}$. В то же время есть функции другого типа: если программа вычисляет $f(x)$ за $N(x)$ шагов, то существует другая программа, вычисляющая ту же функцию за $M(x)$ шагов, причем $M(x)^{M(x)} \leqslant N(x)$, т. е. ускорение больше экспоненциального (*теорема ускорения Блюма*[17]). Палитру дополняет следующий простой результат (см. [6, т. 6]):

2.5.1. *По любой всюду определенной функции g можно построить всюду определенную функцию f, принимающую значения* 0 *или* 1, *вычисление которой невозможно менее чем за $g(x)$ шагов для бесконечного числа значений x.*

Как уже отмечалось[18], *полиномиальное время работы алгоритма* предполагает требуемое число операций $f(x)$ для завершения работы программы равным *О-большому* от x^k, где x — длина описания

[17] Теорема показывает, что попытки связать понятие сложности алгоритмов с длиной вычислительных программ — сопряжены с принципиальными трудностями. Но ситуация не совсем безнадежна, см. раздел 3.7.

[18] Постепенное вхождение в проблематику идет по виткам спирали, на которых старые сведения начинают выглядеть по-новому. Причина большей частью заключена не в изложении, а в восприятии.

задачи, т. е.

$$f(x) = O(x^k) \quad \textit{при некотором} \quad k > 0.$$

Иначе говоря, существует такая константа $C > 0$, что $\boxed{f(x) < Cx^k}$ при достаточно больших x. Это влечет за собой существование положительных констант C_1 и C_2, таких что $f(x) < C_1 x^k + C_2$ при любом x.

Длина описания задачи — еще говорят «*длина входа*» — также определяется с точностью до умножения на положительную константу, и представляет собой «величину», пропорциональную числу символов в записи условия задачи, — при той или иной кодировке данных. Соответствующее место изложения нередко сопровождается многословными пояснениями, диалектически наводящими густой туман. Больший эффект дает концентрация на главном. *Длиной описания задачи* можно считать *количество информации* (число *битов*, т. е. двоичных знаков алфавита $\{0, 1\}$), необходимое для описания задачи. При переходе, например, от двоичной к десятичной кодировке «*длина описания*» уменьшится в $K \sim \log_2 10$ раз [19].

Исключается из рассмотрения *нерациональное* кодирование, — типа единичной записи, при которой числу n сопоставляется n единиц, 11...1. Избежать бесполезных дебатов можно, жестко ориентируясь на *порядок числа битов*, каковой легко оценивается в рамках любой системы счисления.

Граф с n вершинами можно задать с помощью $(0, 1)$-*матрицы смежности*, содержащей n^2 двоичных знаков. Длина описания $\sim n^2$. Если граф реберно-взвешенный, то *длиной описания* будет суммарное число двоичных [20] знаков в записи весов r_{ij} ребер. При ограниченности весов, $r_{ij} \leqslant M$, длина описания будет не более $n^2 \log_2 M \sim n^2$. Причем, как будет видно из дальнейшего, для выяснения полиномиальности алгоритма важен именно порядок *длины описания*, а не ее точное значение.

Что касается *полиномиальности* самого счета, то люфт возможного толкования здесь весьма широк, а скрупулезность определений

[19] А число десятичных знаков после умножения на $\log_2 10$ даст приблизительное количество битов.

[20] Или других — при иной кодировке.

несущественна и даже вредна. С одной стороны, думается, конечно, что понятие алгоритма должно быть строго очерчено. Но как только выбирается что-либо конкретное типа *машины Тьюринга*, за этим тут же возникает шлейф излишней конкретики. Теория на 90 % превращается в обсуждение «вторичных признаков», и лес за деревьями уже не так хорошо виден. В то же время конкретизация важна при определенном психологическом устройстве восприятия, но целесообразна лишь в том случае, когда в душе не возникает тревоги о деталях. Машины Тьюринга, разумеется, допустимы в качестве технологической базы — при ментальной готовности игнорировать их описание. В этом случае «технология» маячит за кадром, а руки развязаны для операций на более высоком уровне абстрагирования.

Конечно, удобнее не привязываться к конкретике, понимая под алгоритмом некую более-менее рациональную программу вычислений, написанную на каком-либо универсальном языке программирования. При этом необходимое число операций для завершения вычислений, разумеется, чувствительно к «языку», — но полиномиальный характер зависимости времени счета от длины описания сохраняется. Машина Тьюринга, например, считает гораздо медленнее обычного компьютера — скажем, в миллион раз. Но именно «в миллион», независимо от размерности решаемой задачи, — полиномиальность счета остается без изменения.

2.6. Вычислительная сложность

Вычислительными ингредиентами алгоритмической практики служат элементарные операции с массивами числовых данных, арифметические действия и другие стандартные манипуляции. Все это является объектом пристального внимания при разработке программного обеспечения, и здесь имеются свои маленькие и большие хитрости, которые в местах сгущения образуют самостоятельные теории [21)]. Несмотря на внешнюю простоту возникающие при этом задачи требуют определенной изобретательности. Даже в мелочах. Матрицу A, скажем, не надо возводить в 33-ю степень, умножая

[21)] См. [11], где рутинные задачи сортировки и поиска поднимаются до уровня искусства.

саму на себя 33 раза. Рациональнее схема

$$A^2 = A \cdot A, \quad A^4 = A^2 \cdot A^2, \quad \ldots, \quad A^{32} = A^{16} \cdot A^{16},$$

и, наконец, $A^{33} = A^{32} \cdot A$. В этом случае алгоритм вычисления A^n становится полиномиальным по n.

Многие проблемы вычислительного толка бывает труднее понять, чем решить. Неискушенный взгляд вполне одобрительно, например, созерцает формулу

$$a_n = \frac{1}{\sqrt{5}}\left[\left(\frac{1+\sqrt{5}}{2}\right)^n - \left(\frac{1-\sqrt{5}}{2}\right)^n\right] \tag{2.6}$$

для вычисления n-го числа Фибоначчи,

$$a_{n+1} = a_n + a_{n-1}, \quad a_1 = a_2 = 1. \tag{2.7}$$

И только программист видит преимущества совсем другого подхода, состоящего в перезаписи (2.7) в виде

$$\begin{bmatrix} a_{n+2} \\ a_{n+1} \end{bmatrix} = A \begin{bmatrix} a_n \\ a_{n-1} \end{bmatrix}, \quad \text{где} \quad A = \begin{bmatrix} 2 & 1 \\ 1 & 1 \end{bmatrix},$$

после чего a_n вычисляется с помощью возведения матрицы A в n-ю степень. Казалось бы — схоластическая выдумка. Но подсчет A^n при целочисленной матрице дает неоспоримые преимущества по сравнению с (2.6), где ошибки округления мешают воспарить над задачей.

Долгое время казалось, что при перемножении двух n-значных чисел можно рассчитывать на трудоемкость порядка n^2, но не менее — *гипотеза Колмогорова*. Гипотетический минимум $\sim n^2$ обеспечивало проверенное тысячелетним опытом «умножение в столбик». Однако в методе «Divide and Conquer» (1960) число необходимых битовых операций неожиданно оказалось пропорциональным $n^{\log_2 3} \sim n^{1,6}$. Находка послужила источником различных технологий *быстрых вычислений*[22].

При широте замаха, для которой все полиномиальные алгоритмы на одно лицо, сказанное, разумеется, не впечатляет, и призвано лишь дать намек о существовании важной и достаточно интересной области алгоритмизации утилитарного назначения, лежащей вне

[22] См.: http://www.ccas.ru/personal/karatsuba/alg.htm либо *Карацуба А. А.* Сложность вычислений // Тр. МИАН им. Стеклова. 1995. **211**. С. 169–183.

избранного здесь направления. Вместе с тем необходимо отметить, что вычислительные рецепты нередко из приземленных плоскостей поднимаются ввысь, откуда видны философские пропасти. Причем сюрпризы обнаруживаются в самых неожиданных местах. Кто бы мог предположить, например, что N-й знак в двоичном разложении числа π

$$\pi = \ldots, \varepsilon_1 \ldots \varepsilon_N \ldots, \quad \text{все } \varepsilon_j \in \{0, 1\},$$

можно определять, не вычисляя предыдущих? Никто. Но потом выяснилось обратное[23]. Задача распознавания *«равно ли ε_N единице?»* — оказалась по трудоемкости далеко не такой, как можно было бы ожидать.

2.7. Задачи верхнего уровня

Коллекцию комбинаторных задач распознавания удобно пополнить следующим экспонатом.

МАШИНА ТЬЮРИНГА (МТ)

Дана машина Тьюринга M, работающая на входном слове $x = \{v, w\}$. Существует ли такое v заданной длины, не превосходящей длины w, что M на слове $x = \{v, w\}$ заканчивает вычисление с результатом «да» за время меньшее

$$p(|x|) = C_1|x|^k + C_2 \, ? \tag{2.8}$$

Модуль здесь обозначает длину x. Входное слово делится на удостоверение v и описание задачи w. В КОММИВОЯЖЕРЕ, например, w_{ij} — веса ребер, v — кольцевой маршрут. Как правило, v и w полиномиально равносильны по длине, при этом вместо (2.8) можно использовать полином $C_1|v|^k + C_2$ с другими константами, что позволяет просто говорить о полиномиальности решающего алгоритма по отношению к v или x. Это имеет место, например, в задаче ВЫПОЛНИМОСТЬ, — где речь идет о поиске

[23] *Bailey D. H., Borwein P. B., Plouffe S.* On the Rapid Computation of Various Polylogarithmic Constants // Math. of Computation. 1997. **66**. P. 903–913.

набора $v = \{v_1, \ldots, v_n\}$, при котором истинны все дизъюнкции $D_1(v), \ldots, D_m(v)$, — если на рассматриваемой совокупности задач число дизъюнкций m растет полиномиально по n. Но если $m \sim n^{\ln n}$, то асимптотической длиной описания задачи следует считать w. В любом случае, однако, можно говорить об ограниченности длины вычислений полиномом $p(|x|)$.

Заметим, что ограниченность времени счета полиномом (2.8) снимает возможные поползновения в сторону неопределенности *проблемы останова*. Вместе с тем присутствие ограничения (2.8) с пользой напоминает о том, что неявно всегда фигурирует.

Задача МТ, как легко сообразить, в некотором роде главная — NP-*полная* — потому что любая NP-задача полиномиально проверяема, по определению. А это означает, что для вычислительного прибора (в частности, для машины Тьюринга) можно написать программу, которая справится с проверкой за полиномиальное время. Поэтому совокупность машин Тьюринга, с будильником (2.8), либо совокупность программ для *универсальной машины*, охватывает все NP-задачи.

Заменяя машину Тьюринга другим вычислительным инструментом, получаем новую «главную» задачу. Новую — описательно. По сути, это будет опять некий NP-*полный* вариант. Как бы, опять двадцать пять. Но у разнообразия форм свои резоны, что временами веско подтверждается. Так или иначе, наряду с МТ имеют смысл и другие задачи, где универсальный характер весь на виду. Скажем, **АЛГОРИТМ МАРКОВА** работает по той же схеме, что и МТ:

Дан нормальный алгоритм Маркова M, работающий на входных словах x. Существует ли такое x заданной длины, $|x| = n$, что M на слове x заканчивает вычисление с результатом «да» за время меньшее (2.8).

Напомним, **нормальные алгоритмы Маркова** [6, т. 6] опираются на системы подстановок типа

$$\begin{cases} ab \Rightarrow c, \\ cc \Rightarrow bbca, \\ cb \Rightarrow bcb, \end{cases} \tag{2.9}$$

что порождает преобразования слов в некотором алфавите $\mathbb{A} = \{a, b, c, \ldots\}$.

В *нормальном алгоритме* слова просматриваются слева направо, подстановки перебираются в заданном порядке, и применяется первая возможная. При отсутствии разрешенной подстановки алгоритм останавливается. Машина Тьюринга представляет собой частный случай нормального алгоритма, преобразуя слово, записанное на ленте, с помощью собственного списка команд, который тривиальными ухищрениями приводится к подстановкам вида (2.9). В то же время нормальные алгоритмы реализуются на подходящих машинах Тьюринга — и потому являются универсальным инструментом для вычислений.

Объектами анализа могут быть и другие NP-задачи аналогичного толка, основанные на *продукциях Поста*, *ассоциативных исчислениях*, или *системах Туэ* [6, т. 6], — с замыканием ответа «да» на *эквивалентность слов*.

Преобразования одних слов в другие удобно мыслить как поиск в бесконечном лабиринте, или бесконечном графе, в котором вершины соответствуют разным словам, а связывающие ребра имеются в тех случаях, когда одно слово получается из другого с помощью одной подстановки. Неразрешимым ситуациям будут отвечать графы, имеющие сколь угодно длинные минимальные пути, связывающие слова априори ограниченной длины. И такие графы могут быть заданы конечными описаниями, а помещение соответствующих NP-задач в прокрустово ложе полиномиального счета возможно ограничением длины допустимых путей.

Глава 3

Проблема $\mathbf{P} \stackrel{?}{=} \mathbf{NP}$

Есть два вида истины — тривиальная, которую отрицать нелепо, и глубокая, для которой обратное утверждение — тоже истина.

Нильс Бор

Болезнь так вплетена, что неотличима от здоровой ткани.

Главная алгоритмическая проблема по накалу страстей, конечно, $\mathbf{P} \stackrel{?}{=} \mathbf{NP}$. Но это идеологически, философски, эмоционально. Практическая ориентация выдвигает на передний план совсем другие темы: вероятностные алгоритмы, параллельный счет, квантовые компьютеры и т. п. Тем не менее начать имеет смысл с попытки углубиться в «жерло вулкана».

3.1. Класс P

Как уже отмечалось, совокупность полиномиально разрешимых задач распознавания (P-*задач*) считается *классом* P. Примерами P-задач служат ПАРОСОЧЕТАНИЕ и МОД.

Еще один важный пример P-задачи 2-ВЫП, т. е. ВЫПОЛНИМОСТЬ, в которой все дизъюнкции содержат по два *литерала*. Полиномиальное решение дает так называемый *метод резолюций*. Суть проста. В наборе $\{x_1, \ldots, x_n\}$ логических переменных берем какое-то x_i и полагаем равным чему-нибудь, например, $x_i = 1$. Далее в «туго завязанной ситуации» процесс разворачивается следующим образом. Какие-то дизъюнкции — из тех, в которые x_i входит, — остаются пока невыполненными (НВ). Это, собственно, те дизъюнкции, в которые x_i входит с чертой, т. е. с отрицанием, — и там парные иксы однозначно определяются. Скажем, в $x_7 + \bar{x}_i$ необходимо $x_7 = 1$, иначе дизъюнкция останется невыполненной, а переменных больше нет. В итоге группа НВ-дизъюнкций определяет значения

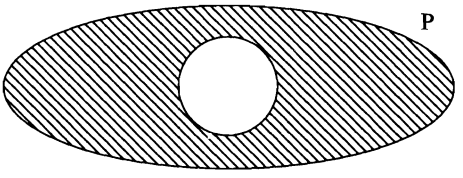


Рис. 3.1

некоторой совокупности иксов. При этих значениях возникает новая группа НВ-дизъюнкций и т. п. Если процесс не заканчивается благополучно, надо вернуться в начало и положить $x_i = 0$. Если на каком-то шаге НВ-дизъюнкций нет, а переменные не исчерпаны, — какому-нибудь из «нетронутых иксов» присваивается какое-либо значение, и машина движется дальше. Легко видеть, что в рутинной толще алгоритма ветвление отсутствует — трудоемкость полиномиальна.

В определении класса P есть одна тонкость. Так или иначе, подразумевается, что P-задачам поставлены в соответствие полиномиальные решающие алгоритмы, и потому *класс* P можно трактовать как множество полиномиальных алгоритмов распознавания. Но по *теореме Райса* не всякий полиномиальный алгоритм доказуемо полиномиален. Поэтому P-задачи (рис. 3.1) делятся на те, где полиномиальность доказуема (белый круг на рис. 3.1), и те, где полиномиальность принципиально невозможно установить (заштрихованная область на рис. 3.1). Учет этого обстоятельства несколько преображает проблему $\mathbf{P} \stackrel{?}{=} \mathbf{NP}$.

3.1.1. Теорема. *Множество полиномиальных алгоритмов распознавания перечислимо, но неразрешимо.*

◀ Неразрешимость класса P вытекает из *теоремы Райса*. Далее, заметим, что всякий полиномиальный алгоритм характеризуется существованием трех положительных констант k, C_1 и C_2, таких что длина вычисления $N(x) < C_1 x^k + C_2$ при любой длине входа x. Пусть

$$M_1, \ldots, M_n, \ldots$$

перечисление всех машин Тьюринга.

Четверке $\{M_i, k, C_1, C_2\}$ будем сопоставлять машину Тьюринга M_j, работа которой совпадает с M_i, если вычисление M_i укладывается в число шагов $N(x) \leqslant C_1 x^k + C_2$, и M_j выдает нуль, если M_i не успевает завершить работу за $C_1 x^k + C_2$ шагов. Перечисляя четверки $\{M_i, k, C_1, C_2\}$, получаем требуемое перечисление полиномиальных алгоритмов распознавания. ▶

3.2. Класс NP

О задаче говорят как о *труднорешаемой* [1], если она не решается полиномиальным алгоритмом. Наличие феномена очевидно. Во-первых, это неразрешимые задачи (глава 2). Во-вторых, это задачи с экспоненциальным, например, размером ответа. В-третьих, это легко конструируемые задачи распознавания. Скажем, если $f_1, \ldots, f_n, \ldots$ — перечисление полиномиальных алгоритмов, гарантируемое теоремой 3.1.1, то функция «$h(n) = $ *анти-*$f_n(n)$», т. е. $h(n) = 0$ в случае $f_n(n) = 1$, и наоборот, — описывает неполиномиальную задачу распознавания.

В качестве труднорешаемых на подозрении находится часть NP-*задач*, в которых, напомним, если решением является ответ «да», то *существует* «слово» x полиномиальной длины [2] и полиномиальный от x алгоритм, дающий ответ «да». Если же решением является ответ «нет», ничего не предполагается. Такова подавляющая часть комбинаторных задач, встречающихся на практике. Их совокупность называется *классом* NP (определение 1.2.3). Таким образом, класс NP — это совокупность *полиномиально проверяемых* задач распознавания. Разумеется, $\mathbf{P} \subset \mathbf{NP}$, но равны ли P и NP, — вопрос на миллион долларов.

В большинстве практических задач *удостоверением* служит «то, что ищется». В КОММИВОЯЖЕРЕ предъявляется подходящий маршрут x, и далее проверяется, меньше ли его длина заданного r. В задаче СОСТАВНОЕ ЧИСЛО предъявляется $x = \{N_1, N_2\}$ и проверяется равенство $N = N_1 \cdot N_2$. Представление о ситуации кардинально меняет задача ПРОСТОЕ ЧИСЛО. Предъявить, на первый взгляд, нечего. Но в [21] [3] предъявляется в качестве x полиномиальной длины доказательство (полиномиально проверяемое). Доказательство в качестве подсказки — несколько раскрепощает мысль и преображает NP-проблематику еще в одном направлении.

---

[1] Термин не вполне устоялся. О труднорешаемости часто говорят лишь за пределами NP-класса, см. далее.

[2] Полиномиальной от *длины входа*. Входное слово x называют также *подсказкой* или *удостоверением*.

[3] См. также *Agrawal M., Kayal N., Saxena N.* Primes in P. 2002 — по адресу http://www.cse.iitk.ac.in/news/primality.pdf.

Что касается данного выше определения *класса* NP, то оно отличается от общепринятого [7], опирающегося на понятие недетерминированного вычислительного процесса, откуда и происходит буква N в обозначении класса. Традиционный ракурс выглядит так: обычная машина Тьюринга M (либо другое универсальное вычислительное устройство) дополняется *оракулом — блоком угадывания*, — способным мгновенно генерировать *удостоверение* x полиномиальной длины[4], которое затем «полиномиально проверяется» и дает ответ «да». Тандем машины M с оракулом называют *недетерминированной машиной Тьюринга*. Понятно, что такая машина решает NP-задачи за полиномиальное время[5], и класс NP можно определить как множество задач, полиномиально решаемых недетерминированными машинами. Никакого влияния на NP-совокупность и на понимание сути проблемы $\mathbf{P} \stackrel{?}{=} \mathbf{NP}$ — смена определения не оказывает. Поэтому при изучении предмета в отсутствие хронической тяги к «углубленному пониманию», — о недетерминированных машинах лучше не упоминать. *Существует ли полиномиальный алгоритм решения, скажем, задачи* ЦЛП? — вот собственно и вся проблема $\mathbf{P} \stackrel{?}{=} \mathbf{NP}$, но об этом речь впереди.

Гроссмейстерский трюк в определении *недетерминированной машины Тьюринга* N-MT заключается в переопределении самого вычислительного процесса. Неопределенность допускается на каждом шаге — жесткий регламент (2.5) работы машины заменяется неопределенным ветвлением:

$$q_i a_j \to \begin{cases} q_k a_l L(R), \\ \ldots\ldots\ldots \\ q_p a_s R(L), \end{cases} \tag{3.1}$$

и задача объявляется решенной, если результат достигается на некоторой ветви. При этом возникает впечатление о несравненно бóльших возможностях такой машины, но это иллюзия — иначе *тезису Тьюринга* [6, т. 6] была бы грош цена. Моделирование

[4] Если таковое существует.

[5] Благодаря всезнающему оракулу.

«решающей ветви» недетерминированного процесса (3.1) на обычной *машине с подсказкой* довольно просто. Например, машина N-MT сначала генерирует подсказку, после чего детерминированным образом ее проверяет.

Вероятностная машина Тьюринга (BMT) работает как и *недетерминированная* — с той разницей, что ветвление в (3.1) определяется с помощью подбрасывания монеты. Если результаты «подбрасывания» записать на специальной ленте, то работу BMT можно свести к работе детерминированной машины Тьюринга с двумя лентами.

Нельзя сказать, что понятие недетерминированной машины вообще излишне. Не говоря о том, что это хороший барьер для школьников, которые бы хлынули добывать миллион, решая проблему $\mathbf{P} \stackrel{?}{=} \mathbf{NP}$, это еще и полезный инструмент исследования. В теории алгоритмов для задач удобно иметь вычислительные устройства, а без оракула приходится, изучая класс NP, навешивать кванторы существования, что выводит анализ за рамки принятого стандарта. Ничего страшного, конечно, однако трансформация определений меняет виденье, что иногда плодоносит[6)].

Так или иначе, недетерминированные машины ставят исследование на другие рельсы, и по ходу дела возникают нюансы, изучение которых обеспечивает работой многие подразделения ученых, без чего успешное развитие науки вообще немыслимо.

3.3. Сводимость и NP-полнота

Конечно, может оказаться $P = NP$, но мнение большинства склоняется в пользу второй альтернативы $P \neq NP$ (рис. 3.2), и в этом предположении ведется основная часть исследований. Если вдруг

[6)] Показательный пример — аналитическая механика, где вариационные принципы, равносильные по сути трем законам *Ньютона*, рождают в итоге совсем другую науку [6, т. 2]. Понятия вообще полезно слегка шевелить, выясняя возможности получения больших выгод малыми средствами. *Лебег*, например, чуть-чуть пошевелив определения [6, т. 5], решил проблему введения *меры*, над которой безуспешно бились великие умы.

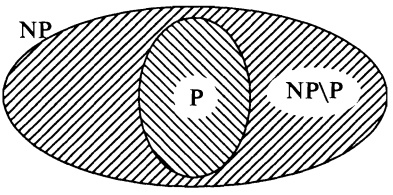


Рис. 3.2

выяснится совпадение P и NP — теория NP-задач лопнет как воздушный шарик. Недетерминированные машины можно будет заменить детерминированными — без потерь в быстроте счета, точнее говоря, с сохранением полиномиальности вычислений. Труднорешаемые задачи станут P-задачами.

Однако пока усилия направлены на выявление структуры класса NP при условии $P \neq NP$. Особое внимание уделяется изучению подмножества так называемых NP-полных задач, базирующихся на понятии полиномиальной сводимости задач.

3.3.1. Определение. *Функция f_1 полиномиально сводится к f_2, — пишут $f_1 \propto f_2$, — если существует полиномиально вычислимая функция f, такая что $f_1(x) = f_2(f(x))$.*

Полиномиальную сводимость 3.3.1 называют *сводимостью по Карпу*[7]. Если $f_1 \propto f_2$ и $f_2 \in P$, то и $f_1 \in P$. Очевидно также

$$f_1 \propto f_2, \quad f_2 \propto f_3 \quad \Rightarrow \quad f_1 \propto f_3.$$

А в случае $f_1 \propto f_2$ и $f_2 \propto f_1$ функции f_1, f_2 считаются *полиномиально эквивалентными*, $f_1 \sim f_2$. Легко видеть, что отношение «$\propto$» задает полуупорядоченность, «$\sim$» — эквивалентность, в результате чего множество NP разбивается на классы полиномиально эквивалентных подмножеств[8], полуупорядоченных в смысле «$\propto$». Структура соответствующего разбиения NP-совокупности — отдельная тема. Очевидно, пожалуй, лишь одно. Внизу иерархии находится класс

[7] Ситуация опять-таки сопровождается дихотомией доказуемости. Функция f в определении 3.3.1 может оказаться доказуемо полиномиальной либо — недоказуемо. Для теоретической надстройки это еще один источник неприятностей. Сводимость 3.3.1 была введена в статье [24].

[8] В случае P = NP эти классы сливаются в один, P.

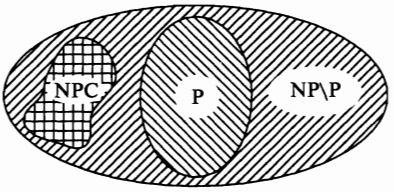


Рис. 3.3

P — «минимально сложных» задач. Верхняя ступень иерархии — NP-полные задачи.

3.3.2. Определение. *Задача называется* ***NP-полной****, или* ***универсальной переборной****, если к ней полиномиально сводится любая другая NP-задача.*

Понятно, что *все* NP-*полные задачи полиномиально эквивалентны друг другу*, — их множество называют NPC-*классом*[9]. Существует ли NPC-*класс*, на первый взгляд, не вполне ясно, — вопрос, тем не менее, решается совсем просто (см. следующий раздел). В предположении $\mathrm{P} \neq \mathrm{NP}$ оказывается $\mathrm{NP}\backslash\mathrm{NPC} \neq \mathrm{P}$ (рис. 3.3), — но это уже второстепенные детали, хотя их не так легко обосновать.

3.4. Универсальная переборная задача

Принято думать, что существование универсальной переборной задачи — сложная проблема. На самом деле — это очевидная вещь. Правда, исторически первой NP-полной задачей стала ВЫПОЛНИМОСТЬ[10], где универсальная природа несколько замаскирована. Поэтому более удобна другая отправная точка. Факт «существования» сразу следует из определения класса NP как совокупности полиномиально проверяемых задач. Последнее означает наличие для любой NP-задачи вычислительного инструмента (рис. 3.4), способного полиномиально по входу $x = \{v, w\}$ — за время меньшее $p(|x|)$, где $p(\cdot)$ заданный полином, — проверять удостоверение v в комплекте с описанием задачи w с помощью алгоритма A.

[9] Буква C в NPC от англ. *complete* (*полный*).

[10] См. далее *теорему Кука*.

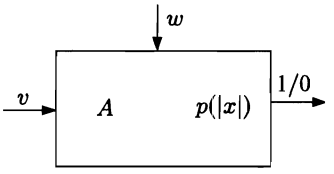


Рис. 3.4

Поэтому NP-*полной* (*универсальной переборной*) является, например, задача МАШИНА ТЬЮРИНГА: *существует ли такое* v, *что машина* M *на слове* $x = \{v, w\}$ *заканчивает вычисление с результатом «да» за время меньшее* $p(|x|)$? Задача в точности охватывает весь класс NP, в силу того что машина Тьюринга является универсальным вычислительным инструментом [11]. Таким образом, факт существования универсальной переборной задачи становится очевидным, что решает философскую часть проблемы. Неясным остается вопрос о том, насколько широк класс NP-полных задач, и входят ли туда менее абстрактные и более привычные постановки типа КОММИВОЯЖЕРА.

3.5. Теорема Кука

Теорема Кука устанавливает NP-полноту ВЫПОЛНИМОСТИ, трансформируя тем самым проблематику в область задач, которые можно «пощупать».

Полиномиальная сводимость ВЫПОЛНИМОСТИ (ВЫП) к МАШИНЕ ТЬЮРИНГА (МТ) не требует доказательства, ибо к МТ (в силу ее NP-полноты) полиномиально сводится любая NP-задача. Обратное легко устанавливается [12]. В первую очередь описание работы машины Тьюринга (2.4) переводится на язык логических переменных $x_{i\lambda}^t$, $q_{i\mu}^t$:

[11] Другая сторона медали здесь связана с полиномиальностью счета, уводящей теорию в малоинтересную дискуссию рутинного характера. Если машина Тьюринга М терпит фиаско, можно предположить, что с задачей «полиномиально справляется» другой инструмент. Но при разумном кодировании и рациональном соответствии вычисление на М будет также полиномиальным [7], что обсуждать невыгодно, ибо связано с неуклюжестью. Да и проколоться можно в связи с *квантовыми вычислениями*.

[12] Но подробные объяснения превращают доказательство в головоломку.

• $x_{i\lambda}^t = 1$, если в момент времени t в i-й ячейке ленты записан символ λ. В противном случае $x_{i\lambda}^t = 0$.

• $q_{i\mu}^t = 1$, если в момент времени t машина просматривает i-ю ячейку, находясь в состоянии μ. В противном случае $q_{i\mu}^t = 0$.

Далее переменные $x_{i\lambda}^t$, $q_{i\mu}^t$ могут быть вытянуты в цепочку, перенумерованы и обозначены вектором $z = \{z_1, \ldots, z_N\}$, после чего работа машины M может быть описана логической функцией $\varphi_M(z)$, где

$$\varphi_M(z) = \varphi_M(z_1, \ldots, z_N) = 1, \tag{3.2}$$

если значениям z отвечают: подходящее начальное слово на ленте (часть массива $x_{i\lambda}^0$), «вычисления» идут в соответствии с процедурой (2.4) и приводят в итоге к результату «да». В противном случае $\varphi_M(z) = 0$.

Задача МТ тем самым сводится к выяснению разрешимости уравнения (3.2). А поскольку любая логическая функция $\varphi_M(z)$ может быть представлена в *конъюнктивной нормальной форме*, т. е. в виде произведения сумм *литералов* [13], — то разрешимость (3.2) есть не что иное, как ВЫПОЛНИМОСТЬ. Не вполне ясным остается лишь полиномиальный характер описания этой задачи по отношению к МТ, но вопрос легко снимается.

Во-первых, число переменных N полиномиально по длине описания задачи, ибо полиномиально время работы машины в задаче МТ [14] (раздел 2.7). Во-вторых, число дизъюнкций в КНФ-представлении $\varphi_M(z)$ полиномиально по N (меньше $C \cdot N^k$ при подходящих константах), хотя это, пожалуй, нуждается в пояснении:

◄ Переход к КНФ-представлению $\varphi_M(z)$ может быть осуществлен в два этапа. Сначала $\varphi_M(z)$ записывается в виде произведения

$$\varphi_M(z) = \varphi_M^1(z) \ldots \varphi_M^s(z), \tag{3.3}$$

[13] Иначе говоря, в виде *конъюнкции дизъюнкций*. Напомним, *литералом* называют логическую переменную или ее отрицание. См. раздел 1.6.

[14] Поскольку речь идет о проверке удостоверений для NP-задач.

где логические функции $\varphi_M^j(z)$ имеют естественную содержательную интерпретацию. В первую очередь, разумеется, часть сомножителей (3.3) должна отражать «программную» часть (2.4) работы машины. Например,

$$\varphi_M^j(z) = \sum_{\nu=F(\lambda,\mu)} x_{i\nu}^{t+1} \cdot x_{i\lambda}^t \cdot q_{i\mu}^t \tag{3.4}$$

истинно, т. е. $\varphi_M^j(z) = 1$, лишь в том случае, если машина работает в соответствии с правилом F — «что в ячейку писать». Аналогично могут быть описаны остальные правила (2.4).

Но этого мало. Логические переменные «свободны друг от друга», и могут принимать значения, удовлетворяющие (3.4), но допускающие присутствие в одной ячейке нескольких символов либо одновременную обработку нескольких ячеек, чего машина Тьюринга не имеет права делать. Такого сорта возможности предотвращает добавление в (3.3) сомножителей типа [15)]

$$\varphi_M^k(z) = \prod_{i,t,\nu\neq\lambda} \left(\bar{x}_{i\nu}^t + \bar{x}_{i\lambda}^t\right). \tag{3.5}$$

В том же духе решается проблема пребывания машины только в одном состоянии, а также корректного начала работы (в состоянии q_1 из крайне левого положения) и завершения вычислений в состоянии q_n.

На следующем этапе каждый сомножитель в (3.3) представляется в конъюнктивной нормальной форме. В случае (3.5) преобразование не требуется. Короткая ДНФ-запись (3.4) преобразуется в

$$\varphi_M^j(z) = \prod_{\nu\neq F(\lambda,\mu)} \left(\bar{x}_{i\nu}^{t+1} + \bar{x}_{i\lambda}^t + \bar{q}_{i\mu}^t\right). \tag{3.6}$$

Длина записи (3.6) больше, но оценка сверху все же полиномиальная, $\leqslant N^2$. Таковы же оценки и для других сомножителей в (3.3), каковых имеется конечное число независимо от размерности решаемой задачи. ►

[15)] Если $x_{i\nu}^t = x_{i\lambda}^t = 1$ при некоторых $\nu \neq \lambda$, — произведение (3.5) обнуляется, что исключает возможность присутствия двух разных символов в одной ячейке.

Таким образом, задача ВЫПОЛНИМОСТЬ полиномиально эквивалентна МТ, и потому — NP-полна.

3.6. Класс NPC

В обойму NPC попадает довольно много популярных задач. Понимание этого факта было главным достижением *Кука—Карпа—Левина* (см. разделы 3.5, 3.7), в результате которого вскрылся обманчивый характер большого разнообразия прикладной комбинаторики. ВЫПОЛНИМОСТЬ, ЗК, ЦЛП, ИЗОМОРФИЗМ ПОДГРАФУ, КЛИКА, ГАМИЛЬТОНОВ ЦИКЛ, — все это оказалось полиномиально эквивалентными лицами универсальной переборной задачи. Список NP-полных задач сегодня насчитывает сотни наименований, и его увеличение уже перестало радовать. Тем не менее первые шаги в этом направлении имеют смысл в общеобразовательных целях.

3.6.1. *Задача ЦЛП NP-полна* ($\in$ *NPC*).

◀ Легко видеть, что ЦЛП $\in$ NP, и потому, в силу *теоремы Кука*, ЦЛП полиномиально сводится к задаче ВЫП. Обратно. Любая дизъюнкция равносильна линейному неравенству. Рецепт сопоставления:

$$x_1 + \bar{x}_3 + x_7 = 1 \quad \Leftrightarrow \quad x_1 + (1 - x_3) + x_7 \geqslant 1,$$

плюс добавление неравенств $x_i \leqslant 1$, $-x_i \leqslant 0$.

Поэтому ВЫПОЛНИМОСТИ отвечает совокупность линейных целочисленных неравенств (трудоемкость преобразования линейна). Иначе говоря, ВЫП полиномиально сводится к ЦЛП [16]. ▶

Использованная схема доказательства стандартна. Если некоторая задача T $\in$ NP, то она, очевидно, сводится полиномиально к любой NP-полной задаче, и для обоснования NP-полноты T надо лишь установить полиномиальную сводимость к T хотя бы одной задачи из NPC, NP-полнота которой уже доказана.

[16] Из доказательства видно, что NP-полна также (0, 1)-ЦЛП.

3.6.2. *Задача 3-ВЫП NP-полна* [17].

◄ Дизъюнкция, содержащая более трех литералов, $D_j = \zeta_1 + \ldots + \zeta_n$, может быть заменена произведением $n - 2$ дизъюнкций:

$$D_j = (\zeta_1 + \zeta_2 + \theta_1)(\theta_1 + \theta_2 + \zeta_3)(\theta_2 + \theta_3 + \zeta_4) \ldots (\theta_{n-3} + \zeta_{n-1} + \zeta_n),$$

где $\theta_1, \ldots, \theta_{n-3}$ — вспомогательные литералы.

«Усложнить» ситуацию $n < 3$ «до трех» — еще легче. ►

3.6.3. *Задача КЛИКА* $\in$ *NPC.*

◄ Логической функции в КНФ-представлении

$$\varphi(\cdot) = D_1 \ldots D_k, \quad D_j = (\lambda_{j1} + \ldots + \lambda_{jk_j}), \tag{3.7}$$

сопоставим граф G, вершинами которого будут все литералы λ_{js}, каковые попарно соединим ребрами, если только они принадлежат разным дизъюнкциям D_j, и «не противоположны», т. е. «не x и $\bar{x}$». Следовательно, ребра соединяют такие пары литералов из разных скобок D_j, которым одновременно можно придать значение 1, *истинно*. Тем самым *клике* (полному подграфу) с k вершинами в G соответствует k литералов по одному из разных $D_1, \ldots, D_k$, которым одновременно можно придать значение *истинно*, — в результате чего формула (3.7) будет выполнена.

Таким образом, задача ВЫПОЛНИМОСТЬ сводится к КЛИКЕ, легко видеть — полиномиально. ►

В силу *теоремы* 1.4.1 NP-полны также АНТИКЛИКА и ВЕРШИННОЕ ПОКРЫТИЕ.

Список задач NPC может быть значительно продолжен. Одно лишь перечисление заняло бы немало страниц. Поэтому ограничимся упоминанием нескольких популярных вариантов.

3.6.4. *NPC принадлежат задачи: ТРЕХМЕРНОЕ СОЧЕТАНИЕ, РЮКЗАК, РАЗБИЕНИЕ, ИЗОМОРФИЗМ ПОДГРАФУ.*

Особое положение в этом ряду занимает, пожалуй, следующий результат.

[17] Иначе говоря, ВЫПОЛНИМОСТЬ полиномиально может быть сведена к задаче об одновременной выполнимости дизъюнкций, содержащих ровно по три литерала.

3.6.5. *ГАМИЛЬТОНОВ ЦИКЛ и КОММИВОЯЖЕР* $\in$ *NPC.*

Дело в том, что в отличие от предыдущих утверждений доказательство теоремы 3.6.5 существенно более громоздко и неудобно как для изложения, так и для восприятия. Изящные конструкции есть в [7, 17], но для проникновения в NP-проблематику тратить силы на их освоение необязательно [18]. Здесь важен сам факт нетривиальной полиномиальной эквивалентности задач, которые возникли вдали друг от друга. Когда сводимость доказывается в две строчки, там можно думать о совпадении природы и отличиях формы. Но когда совсем непохожие задачи, инициированные на вид совсем разными причинами, удается связать «равенством», — есть над чем задуматься.

Удивительно вообще, что все задачи, которые «не попадают» в класс P, оказываются в основном NP-полными. Множество NP\P в предположении NP$\neq$P не пусто, но попасть в него почему-то весьма трудно. Почти все независимо возникающие труднорешаемые задачи соединены ниточкой полиномиальной эквивалентности с NPC. В чем причина? Простейшим объяснением было бы NP = P.

3.7. Подход Левина

Теория NP-полных задач была инициирована работой *Кука* [23], и приобрела современный вид под влиянием *Карпа* [24]. Не обошлось, как всегда, без предыстории, западная часть которой описана в [7]. О российских истоках информация есть у *Разборова* [19].

Особого упоминания заслуживает работа *Левина* [13], в которой та же самая теория была рождена в ином одеянии. Вот едва ли не дословное воспроизведение результатов [13].

3.7.1. *Функции* $f(n)$ *и* $g(n)$ *называются* ***сравнимыми****, если при некотором* k

$$f(n) \leqslant [g(n)+2]^k, \quad g(n) \leqslant [f(n)+2]^k.$$

[18] Другое дело, если возникает потребность доказать NP-полноту новой задачи, и не хватает примеров для подражания.

[19] *Разборов А. А.* $\mathbf{P} \stackrel{?}{=} \mathbf{NP}$ или проблема перебора: взгляд из 90-х http://www.mi.ras.ru/razborov/phasis.ps.

3.7.2. *Задачей* ***переборного типа*** *называется задача вида «по данному x найти какое-нибудь y длины, сравнимой с длиной x, такое что выполняется $A(x, y)$», где $A(x, y)$ — какое-нибудь свойство, проверяемое алгоритмом, время работы которого сравнимо с длиной x. Существует ли такое y —* ***квазипереборная задача.***

Кук [23] в качестве NP-полных выделил 6 задач, *Карп* [24] расширил список до 21 пункта. Список *Левина*:

Задача 1. Заданы списком конечное множество и покрытие его 500-элементными подмножествами. Найти подпокрытие заданной мощности (выяснить существует ли оно).

Задача 2. Таблично задана частичная булева функция. Найти заданного размера ДНФ, реализующую эту функцию в области определения.

Задача 3. Выяснить, выводима или опровержима данная формула исчисления высказываний (равна ли константе данная булева формула).

Задача 4. Даны два графа. Найти гомоморфизм одного на другой.

Задача 5. ИЗОМОРФИЗМ ПОДГРАФУ.

Задача 6. Рассматриваются матрицы из целых чисел от 1 до 100 и некоторое условие о том, какие числа в них могут соседствовать по вертикали и какие по горизонтали. Заданы числа на границе и требуется продолжить их на всю матрицу с соблюдением условия.

3.7.3. Теорема. *Если вообще существует какая-нибудь массовая задача переборного (квазипереборного) типа, неразрешимая за время меньшее $f(n)$ при длине аргумента, сравнимой с n, то этим же свойством обладают задачи* 1–6.

Иначе говоря, *задачи* 1–6 *являются универсальными переборными*, — в следующем смысле.

3.7.4. *Пусть $A(x, y)$ и $B(x, y)$ определяют соответственно переборные задачи A и B. Задача A сводится к B, если есть три алгоритма $r(x)$, $p(y)$, $s(y)$, работающие за время, сравнимое с длиной аргумента, такие что*

$$A(x, p(y)) \equiv B(r(x), y) \quad \text{и} \quad A(x, y) \equiv B(r(x), s(y)).$$

Задача, к которой сводится любая задача перебора, называется ***универсальной.***

Вот собственно и вся работа [13]. Не считая малополезного с виду добавления: *для произвольной массовой переборной задачи* $A(x, y)$ *существует алгоритм, решающий ее за время, оптимальное с точностью до умножения на константу и прибавления величины, сравнимой с длиной* x. Результат, тем не менее, весьма примечателен, ибо показывает, что негативная роль *теорем Блюма об ускорении* [20] (раздел 2.5) не так велика.

3.8. Полиномиальное раздутие

Если мы не знаем «A равно B или не равно», — можно подействовать на оба элемента неким преобразованием Θ и проверить равенство $\Theta(A)$ и $\Theta(B)$. Трансформация может оказаться легче, удобнее. Либо сложнее, но глубже.

Последнее как раз и происходит с P- и NP-задачами. Проблема $\mathbf{P} \stackrel{?}{=} \mathbf{NP}$ сама по себе заключена в стерильном вопросе «имеет ли задача ВЫП [21] полиномиальный алгоритм решения». Вместо этого рассматривается более сложная на вид задача о совпадении классов P и NP, где NP получается *полиномиальным раздутием* ВЫПОЛНИМОСТИ по описанному уже рецепту охвата всех задач, полиномиально сводящихся к ВЫП. Ситуация размывается. Вместо «точечного равенства $A = B$» приходится изучать совпадение множеств $\Theta(A)$ и $\Theta(B)$. Но именно такое «усложнение» раскрепощает исследование, включая новые степени свободы. Многосторонний взгляд на этом пути не решает пока проблемы, но проливает дополнительный свет и приносит побочные плоды, каковые постепенно становятся главными. Весьма вероятно, что в рассматриваемой области ничего стоящего не останется, кроме полиномиальной сводимости и NP-полноты, не считая дразнящего неразрешимостью вопроса $\mathbf{P} \stackrel{?}{=} \mathbf{NP}$.

---

[20] С точки зрения попыток связать сложность алгоритмов с длиной вычислительных программ.

[21] Либо какая угодно другая NP-полная задача.

Пример изучения NP-задач демонстрирует возможности трансформации проблем в более сложные, но и более информативные одежды. Подобный трюк заслуживает общефилософского осмысления и широкой проверки в разных вариантах. «Раздувать» задачи можно, разумеется, не только полиномиально.

3.9. Будет ли решена проблема $\mathbf{P} \stackrel{?}{=} \mathbf{NP}$

Не так давно проводился опрос по поводу $\mathbf{P} \stackrel{?}{=} \mathbf{NP}$. Интересно, что среднестатистическое мнение насчет срока решения проблемы было где-то в районе 2050 г.[22], что символично. Когда математик прогнозирует решение задачи через 50 лет, это означает, что он не видит подходов к ее решению. Что касается прогнозов насчет «да» или «нет», то 61 из 100 сказали «$P \neq NP$», 9 — «$P = NP$», 22 — затруднились с ответом, а 8 — предположили неразрешимость в рамках *аксиоматики* ZFC [6, т. 6].

Большинство сторонников «$P \neq NP$» опираются, главным образом, на интуитивное ощущение существования легко- и трудно-решаемых задач, что, безусловно, факт — причем не интуитивный, а вполне доказуемый. МОД и ЦЛП — принципиально разные по сложности задачи, и в главе 7 указана *мера их трудоемкости*[23]. Но это ничего не говорит в пользу «$P \neq NP$», потому что полиномиальных задач вполне может быть несколько категорий, и такие P-задачи, как МОД и ЛП — яркое тому подтверждение. Обе полиномиальные, но далеко не одного поля ягоды. Поэтому гипотеза «$P = NP$» ничему не противоречит.

Аргумент «$P \neq NP$, потому что многие не сумели доказать обратное» — не выдерживает критики. Не только потому, что многие не сумели доказать ни то, ни другое. Обоснование полиномиальной разрешимости NP-задач вообще не там искали — поэтому «$P = NP$», можно сказать, еще девственная проблема. Вопрос подробно рассматривается в главе 7. Суть дела в двух словах

[22] Не считая особого мнения *Дональда Кнута*: задача будет решена либо в 2048, либо в 4096 г.

[23] Плотность графа многогранника задачи.

сводится к тому, что NP-полные задачи имеют высокую *плотность графа* своего многогранника, тогда как полиномиальные — низкую, а комбинаторные алгоритмы оказываются *алгоритмами прямого типа*, которые в принципе не могут избавить от перебора при экспоненциальных плотностях. Поэтому, если уж доказывать «P = NP», то рецепты надо искать за пределами комбинаторной идеологии, до чего руки пока не доходят[24)].

Таким образом, в пользу «P ≠ NP» нет ни одного аргумента. За исключением чисто эмоционального заблуждения в связи с существованием задач, которые «сложнее простых». КОММИВОЯЖЕР, например, против СОРТИРОВКИ. Спору нет — сложнее, причем доказуемо сложнее, аргументированно. Однако трудоемкость может очень сильно расти в пределах класса P, что, собственно, и происходит при переходе от элементарных P-задач к ЛИНЕЙНОМУ ПРОГРАММИРОВАНИЮ или ПРОСТОМУ ЧИСЛУ. Философский вопрос о полиномиальности ЛП и ПЧ решается положительно, однако для ЛП вместо *алгоритма Хачияна* используется все же переборный симплекс-метод, а для ПЧ вместо AKS-*алгоритма* — вероятностные манипуляции.

В пользу «P = NP» аргументы, наоборот, есть. Они консолидированно представлены в главе 7. Если же говорить «по верхам», то это: обоснование возможности экспоненциального роста плотности графа задачи ЛП, которая все же остается в классе P; теорема 7.4.1 о полиномиальной глубине ЛРД для любой задачи, в том числе NP-полной; наконец, понимание того, что прежние поиски общего полиномиального алгоритма были холостым выстрелом, — и надо искать в другом направлении. Последнее особенно вдохновляет, потому что исследовательские тропы, оказывается, не затоптаны. Но гарантий успеха, разумеется, нет.

[24)] Попытки для ЛИНЕЙНОГО ПРОГРАММИРОВАНИЯ и ПРОСТОГО ЧИСЛА сразу дали положительный результат — главы 5, 6.

Глава 4

Анатомия переборных задач

Путь к себе — есть путь отказа,
Колдовство.
Ох, и спрятано, зараза,
Естество.

Исследования вокруг $\mathbf{P} \stackrel{?}{=} \mathbf{NP}$ давно перешли в хроническую форму. Гипертрофированные очертания, неадекватное количество новых понятий, — все это говорит о запущенности болезни, в чем полезно отдавать себе отчет, чтобы созерцать процесс, не поддаваясь частностям.

4.1. Проблема co-NP$\stackrel{?}{=}$NP

Если задача T состоит в выяснении того, существует ли элемент $x \in X$, обладающий свойством $\mathcal{B}$, то *дополнение задачи* T^c заключается в установлении того, что ни один элемент $x \in X$ не обладает свойством $\mathcal{B}$. Совокупность дополнений NP-задач образует *класс* co-NP.

Вообще говоря, не ясно, совпадает ли co-NP с классом NP. Трудно себе представить, например, эффективную подсказку для дополнения к КЛИКЕ. Похоже, необходимо перебрать и отсеять все варианты. Хотя, кто знает... Но так или иначе, проблема co-NP$\stackrel{?}{=}$NP остается нерешенной. Разумеется, *дополнения к* P-*задачам не выводят из класса* P — рис. 4.1, — т. е. co-P = P, потому что если алгоритм не выдает в P-задаче «да» за полиномиальное время, значит — «нет» [1]. Возможное равенство co-NP = NP не исключает NP $\neq$ P. Подозрение co-NP $\bigcap$ NP = P — не доказано.

[1] Для этого, правда, надо знать полиномиальную оценку.

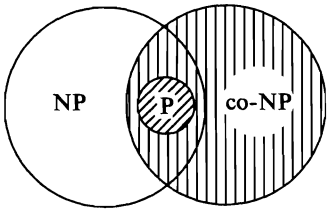


Рис. 4.1

Как и в случае $P \stackrel{?}{=} NP$, проблема $co\text{-}NP \stackrel{?}{=} NP$ сводится к проверке всего одной задачи.

4.1.1. Теорема. *Если существует такая NP-полная задача T, что $T^c \in NP$, то NP=co-NP.*

◀ Любая задача $A \in NP$ полиномиально преобразуется в T. То же самое преобразование полиномиально переводит A^c в T^c. ▶

4.2. Сильная NP-полнота

В постановках некоторых комбинаторных задач фигурируют числа, благодаря разбросу которых задачи оказываются NP-полны, а при ограничениях диапазона — переходят из NP в P. Задача ЦЕЛОЧИСЛЕННЫЙ РЮКЗАК, где речь идет о разрешимости

$$\sum_{k=1}^{n} w_k x_k = W, \quad x_k \in \mathbb{N}, \tag{4.1}$$

обычно служит иллюстрацией. Коэффициенты в (4.1) также предполагаются целыми.

Сопоставим задаче ориентированный граф G с множеством вершин

$$V = \{0, 1, 2, \ldots, W\},$$

соединенных ребрами e_{ij} лишь в том случае, если $i < j$ и $j - i = w_k$, т. е. разница номеров вершин $j - i$ совпадает хотя бы с одним весом w_k из (4.1). Каждому w_k будет отвечать, таким образом, $W - w_k + 1$ ребер. Всего ребер будет $O(nW)$ — нас устраивает грубая оценка.

4.2.1. *Задача* (4.1) *разрешима в томм случае, когда в графе G существует путь из* 0 *в W.*

◀ Результат, в принципе, очевиден. Если такой путь существует, то его длина — если ребрам приписать веса $|e_{ij}| = |j - i| = w_k$ — будет равна как раз величине (4.1), где x_k — число ребер длины w_k, входящих в этот путь. Обратное также ясно. ▶

Алгоритм поиска соответствующего пути довольно прост. Пометим вершины G, из которых можно добраться в W за один шаг, индексом 1, и обозначим их множество через J_1. Далее пометим индексом 2 вершины, из которых можно добраться в J_1 за один шаг, и обозначим их множество через J_2. Продолжая процесс, $J_1, J_2, \ldots$, на каком-то шаге $p \leqslant W$ получим либо $0 \in J_p$, либо вершина 0 так и останется «нетронутой». В случае $0 \in J_p$ берем любое ребро, ведущее из 0 в одну из вершин $u \in J_{p-1}$, потом из u в J_{p-2}, и так пока не доберемся до W. На поиск уходит $O(nW)$ шагов.

Поскольку длина описания задачи (4.1) $L \sim n \log W$, то итог $O(nW) = O(n \cdot 2^{\log W})$ все же «не полиномиальный». Однако если W ограничено полиномом от L, то временна́я трудоемкость рассмотренного алгоритма оказывается полиномиальной — и по этой причине такого сорта алгоритмы (и задачи) называют *псевдополиномиальными*. Хитрость, таким образом, невелика. Сужение NP-полной задачи попадает в класс P, что неудивительно. Но речь в данном случае идет о сужении за счет ограничения чисел, входящих в описание задачи. В поле зрения попадают, таким образом, только числовые задачи. Но КОММИВОЯЖЕРУ, например, никакие ограничения на веса ребер не помогают — задача ГАМИЛЬТОНОВ ПУТЬ NP-полна, как и сама ЗК. Такие задачи — не имеющие псевдополиномиальных алгоритмов решения — называют *сильно* NP-*полными*. Сильно NP-полны КЛИКА, ВЫПОЛНИМОСТЬ, ВЕРШИННОЕ ПОКРЫТИЕ, ИЗОМОРФИЗМ ПОДГРАФУ — и другие NP-полные задачи, в постановках которых нет чисел.

Задача ЦЛП тоже *сильно* NP-*полна*, хотя и числовая, — ибо заключение ее коэффициентов даже в прокрустово ложе $\{0, 1\}$ оставляет задачу NP-полной (включающей в себя эквивалент ВЫПОЛНИМОСТИ). Псевдополиномиальных задач не так много[2)],

[2)] Помимо РЮКЗАКОВ некоторые задачи составления расписаний и еще отдельные аномалии [7].

но тема популярна в своем негативе. Для NP-задач принято доказывать невозможность их псевдополиномиального решения.

Если внимательно присмотреться, то понятие *сильной* NP-*полноты* несколько расползается, потому что сузить поле зрения строго до «задач с числами» — трудно. Помимо длин, весов и коэффициентов в задачах есть вершины, ребра, переменные, неравенства, — ограничения на число которых могут приводить к соскальзыванию из NP в P, что, так или иначе, допустимо характеризовать как псевдополиномиальность. Стандартный пример КЛИКА в графе с числом вершин N и максимальной степенью вершины ρ, решаемая за время N^{ρ}, что — полиномиально, если ρ ограничено.

4.3. Кратчайший путь

Ряд популярных комбинаторных задач сопровождается в литературе досадной неразберихой. Таковы задачи о *кратчайшем пути*, *максимальном потоке*, *минимальном разрезе* и некоторые другие. В одном месте изучаются полиномиальные алгоритмы для поиска кратчайшего пути, в другом — устанавливается NP-полнота той же самой задачи. Источник видимого противоречия заключается в одинаковых названиях для разных задач. При этом незначительные на вид вариации постановок переводят задачи из одной категории в другую.

Возьмем, для примера, КРАТЧАЙШИЙ ПУТЬ: *дан реберно-взвешенный граф с двумя выделенными вершинами* s *и* d; *найти простой* (*без циклов*) *путь из* s *в* d *минимальной длины*. На самом деле это не задача, а скелет, — потому что решение зависит от неоговоренных обстоятельств. Во-первых, неясно, идет ли речь об ориентированном или неориентированном графе[3]. Во-вторых, надо бы уточнить требования к весам ребер x_{ij}. Вопрос, навскидку, не стоит выеденного яйца. Однако *при положительных весах* x_{ij} *задача решается полиномиально, тогда как при отрицательных* —

[3] Но это, в конце концов, не так принципиально, поскольку в неориентированном случае под каждой дугой можно подразумевать две противоположно ориентированные.

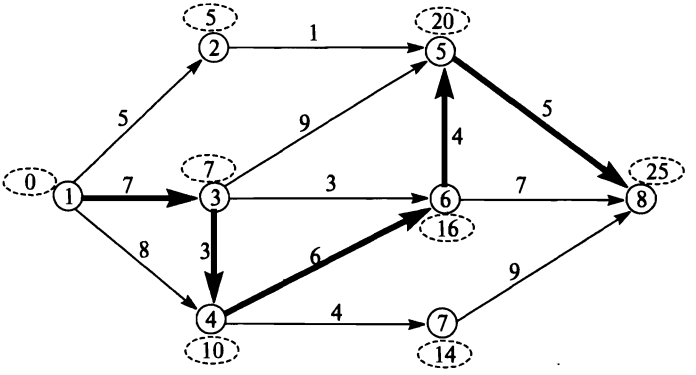


Рис. 4.2

является NP-полной. Можно сказать еще по-другому: *при положительных весах ребер задача о кратчайшем пути — полиномиальна, о длиннейшем пути — NP-полна.*

Феномен в некотором роде бьет по мозгам, потому что есть привычка к равноценности задач на минимум и на максимум. Но если не оставаться у поверхности фактов, причины аномалии вполне тривиальны. Досконально объяснять, правда, долго — проще разобраться самостоятельно, отправляясь от какой-либо простой ситуации. Скажем, от примера, который приводился в [6, т. 7] по поводу сетевых методов. Задача в данном контексте выглядит так. Для определения самого длинного пути из вершины ① в ⑧ (рис. 4.2) поступают следующим образом[4]. Исходной вершине ① приписывается *потенциал* 0. Потенциалы остальных вершин (обведены пунктиром) вычисляются последовательно, сложением веса входящих ребер с потенциалами вершин, из которых эти ребра выходят, и выбором максимальной суммы. Подходящая последовательность рассмотрения вершин обязательно найдется, как легко убедиться, *если в графе нет циклов.* Потенциал последней вершины ⑧ равен длине оптимального пути. Для определения самого длиннейшего пути, который выделен жирным на рисунке, граф просматривается в обратном направлении [6, т. 7], что представляет собой стандартную процедуру *динамического программирования.*

Алгоритм полиномиален ($\sim n^2$), но работает только при условии, если в графе нет циклов. В противном случае при расстановке потенциалов не удается «сдвинуться с места». При этом ясно, что препятствия возникают не только для данного алгоритма. Задача

[4] Числа на дугах задают веса ребер.

вообще становится крайне неудобной (NP-полной) — см. далее теорему 4.3.1.

Если же ищется кратчайший путь, ситуация кардинально меняется. Фактически работает тот же алгоритм с заменой «максимальных сумм» минимальными. Незначительная коррекция дает полиномиальный *алгоритм Дейкстры*, работающий и при наличии в графе циклов, но в предположении неотрицательности весов, $x_{ij} \geqslant 0$. *Алгоритм Флойда—Уоршелла* работает на основе повторения операции замены [5)]

$$\forall\, i, j \neq k : x_{ij} := \min\{x_{ij},\ x_{ik} + x_{kj}\}$$

и справляется с задачей за время $\sim n^3$ даже в присутствии отрицательных весов, но при отсутствии циклов отрицательной длины. Подробности есть в [17], но здесь стоит заметить, что минимальное усилие мысли делает проблему абсолютно прозрачной, тогда как письменные объяснения лишь затемняют суть.

Об NP-полноте задачи в общем случае свидетельствует следующий результат.

4.3.1. Теорема. *КОММИВОЯЖЕР полиномиально сводится к задаче ДЛИННЕЙШИЙ ПУТЬ* [6)].

◀ Пусть реберно-взвешенный граф G имеет n вершин, x_* — минимальная длина ребра из всех x_{ij}, x^* — максимальная, $x^* - x_* > 0$, и речь идет о существовании гамильтонова цикла в G длины не меньше $K > nx_*$.

Выделим произвольно вершину номер 1, и дополним граф $(n+1)$-й вершиной, соединенной с остальными. Ребрам полученного графа G' припишем веса: $z_{ij} = x_{ij} - x_* + n(x^* - x_*)$ в пределах G; $z_{1(n+1)} = 0$, наконец,

$$z_{i(n+1)} = x_{1i} - x_* + n(x^* - x_*), \quad i \neq 1.$$

Положим

$$K' = (x^* - x_*)n^2 - nx_* + K.$$

Легко видеть, что путь из первой в $(n+1)$-ю вершину в G' может быть не меньше K' лишь в том случае, когда он проходит через все вершины. ▶

[5)] В результате выполнения которой для всех k элементы x_{ij} становятся равными кратчайшим путям из i в j.

[6)] Иначе говоря, ДЛИННЕЙШИЙ ПУТЬ не проще КОММИВОЯЖЕРА. При доказательстве удобнее рассматривать задачи распознавания. Поэтому под «ДЛИННЕЙШИМ ПУТЕМ» подразумевается существование пути длины не меньше заданного K'.

4.4. Максимальный поток и минимальный разрез

Вот классический вариант задачи о максимальном потоке в сети. На графе выделены две вершины: s — *источник*, d — *сток*. Поток из s в d, разветвляясь в вершинах, «течет» по ребрам из i-й вершины в j-ю. Каждое ребро характеризуется *пропускной способностью* $\sigma_{ij} \geqslant 0$. Поток по ребру $x_{ij} \geqslant 0$ обязан удовлетворять условию $x_{ij} \leqslant \sigma_{ij}$, причем в каждой вершине, за исключением s и d, предполагается выполненным «*закон Кирхгофа*»:

$$\sum_j x_{ij} = \sum_k x_{ki}, \quad i \neq s,\ d, \tag{4.2}$$

т. е. суммарный поток, входящий в вершину i, равен — выходящему. В силу (4.2) поток $\sum_j x_{sj}$, выходящий из s, совпадает с потоком $\sum_i x_{id}$, входящим в d.

Задача о максимальном потоке, таким образом, имеет вид:

$$\begin{aligned} &\sum_j x_{sj} \to \max, \\ &\sum_j x_{ij} = \sum_k x_{ki} \quad (k \neq s,\ d), \\ &0 \leqslant x_{ij} \leqslant \sigma_{ij}, \end{aligned} \tag{4.3}$$

и представляет собой частный случай задачи ЛП (глава 5) в двухиндексной записи. Двойственная задача ЛП в данном случае имеет естественную содержательную интерпретацию, и вовлекает в рассматриваемую проблематику еще одну задачу (*о минимальном разрезе*). *Разрезом* C_S, или *пропускной способностью сечения*[7] S, называется суммарная пропускная способность ребер, входящих в S и выходящих из $\bar{S}$.

Переформулировка пп. 5.1.1, 5.1.2 приводит к следующему результату.

4.4.1. *Величина любого допустимого потока не превышает минимального разреза C_S. Максимальный поток равен минимальному разрезу.*

Полиномиальный алгоритм поиска максимального потока опирается на простую схему. Сначала берется любой допустимый поток, например нулевой. Затем проверяется, существует ли путь из s в d, состоящий из ненасыщенных ребер

[7] *Сечением* S в сети называется любое разбиение множества вершин на два непересекающихся подмножества S и $\bar{S}$, при условии $d \in S$, $s \in \bar{S}$.

(ребро (i, j) ненасыщенно, если $x_{ij} < \sigma_{ij}$). Если «нет» — алгоритм останавливается, поток максимален. Если «да» — в пути из ненасыщенных ребер все потоки увеличиваются на величину $\min\{\sigma_{ij} - x_{ij}\}$, где минимум берется по ребрам избранного пути, после чего алгоритм возвращается к предыдущему пункту.

Задача о максимальном потоке при дополнительном ограничении целочисленности переменных решается полиномиально точно так же. Это до некоторой степени удивительно, поскольку общая задача линейного программирования в целочисленном варианте является NP-*полной*. Причина заключается в следующем. Дискретная задача (4.3) как непрерывная ЛП — в силу своей специфики имеет решение в целочисленной вершине допустимого многогранника, что открывает ей дорогу в класс P.

Люфт постановки (4.3) довольно велик, поэтому вокруг рассмотренной задачи есть много вариаций с аналогичными названиями, но совсем другими характеристиками. Большинство потоковых задач и задач о разрезах NP-полны [7], причем таковыми их делают совсем незначительные на вид усложнения. Вот два простейших примера.

Целочисленная *задача о минимальном разрезе*, вернее, ее вариант распознавания, становится NP-полной (даже при равенстве всех весов ребер) при введении ограничения на число вершин сечения: $|S|, |\bar{S}| \leqslant R$. И только в случае $R = |V|$ есть гарантия полиномиальной разрешимости. А вариант распознавания *задачи о максимальном разрезе* NP-полон, опять-таки даже при равенстве всех весов ребер, но уже без каких бы то ни было ограничений на величину $|S|$.

4.5. Теория матроидов и жадный алгоритм

Труднорешаемые задачи от легкорешаемых отличаются, по крайней мере тем, что уже почти сорок лет не поддаются усилиям вскрыть их подлинную сущность. Критерий, конечно, эмоциональный. Более конструктивный индикатор описан в главе 7, а на объективное различие нельзя надеяться, пока не решена проблема $\mathrm{P} \stackrel{?}{=} \mathrm{NP}$. Тем не менее отдельные попытки увидеть разницу в частных секторах

интерпретаций представляют интерес, ибо продвигают к пониманию — хотя бы иллюзорно. Один из таких секторов — *теория матроидов*, в рамках которой возникают подходящие иллюзии насчет признаков P-класса.

Матроиды были введены *Уитни*[8] как обобщение понятия линейной независимости. По определению ими (*матроидами*) называют пару $(E, \mathcal{M})$, где E — непустое конечное множество, $\mathcal{M}$ — семейство его *независимых* подмножеств, удовлетворяющих двум характеристическим условиям: **(i)** любое подмножество независимого множества независимо; **(ii)** если $A, B \in \mathcal{M}$, причем B содержит на один элемент больше чем A, то существует такой элемент $e \in B$, $e \notin A$, что $A \cup \{e\} \in \mathcal{M}$.

Максимальное независимое множество называется *базой матроида*. Все базы, как выясняется, имеют одинаковое число элементов[9]. Возможно также эквивалентное определение матроида через семейство его баз как пары $(E, \mathcal{B})$, где E — непустое конечное множество, $\mathcal{B}$ — семейство *баз*, удовлетворяющих двум характеристическим условиям: **(i)**° любое собственное подмножество базы не является базой; **(ii)**° если $B_1, B_2 \in \mathcal{B}$ — базы, и $e \in B_1$, то существует такой элемент $f \in B_2$, что $(B_1 - \{e\}) \cup \{f\}$ — тоже база.

Не независимые подмножества E называют *зависимыми*. Минимальное по включению зависимое множество считается *циклом*. Возможно эквивалентное определение матроида (и цикла) как пары $(E, \mathcal{C})$, где E — непустое конечное множество, $\mathcal{C}$ — семейство его *циклов*, удовлетворяющих двум условиям: **(j)** никакой цикл не содержит в качестве подмножества другой цикл; **(jj)** если $C_1, C_2 \in \mathcal{C}$ — различные циклы, каждый из которых содержит элемент e, то в $C_1 \cup C_2$ существует цикл, не содержащий e.

> *О матроиде чаще всего говорят просто как о семействе подмножеств, не упоминая E, молчаливо подразумеваемое.*

В контексте дискретной оптимизации основной интерес представляет эффективность *жадного алгоритма* в рамках следующей

---

[8] *Whitney H.* On the abstract properties of the linear dependence // Amer. J. Math. 1935. **35**. 509–533.

[9] *Условие* **(ii)** *в предположении* **(i)** *равносильно требованию: любые два максимальных независимых множества имеют одинаковую мощность.* Аналогия с линейной независимостью вполне прозрачна. Любые два базиса в $\mathbb{R}^n$ содержат одинаковое число векторов.

задачи: *дано непустое конечное множество* E, *семейство его подмножеств* $\mathcal{M}$ *и неотрицательная функция* $w(x)$ *на* E; *требуется найти множество* $A \in \mathcal{M}$, *на котором*

$$w(A) = \sum_{x \in A} w(x) \tag{4.4}$$

достигает максимума.

Жадный алгоритм в описанной ситуации действует примитивно. Сначала выбирается самый большой элемент $x \in E$, обеспечивающий наибольшее значение $w(x)$. Затем — самый большой из оставшихся, при условии что его добавление не выводит из $\mathcal{M}$. Если на некотором шаге несколько элементов имеют максимальный вес — выбирается любой, не выводящий из $\mathcal{M}$. В результате образуется последовательность множеств

$$A_1 \subset A_2 \subset \ldots \subset A_N = A, \tag{4.5}$$

заканчивающаяся некоторым A.

4.5.1. Теорема. *Если* $(E, \mathcal{M})$ — *матроид, то для любой функции* $w(x) \geqslant 0$ *жадный алгоритм находит множество* $A \in \mathcal{M}$, *обеспечивающее максимум функции* (4.4) [10].

◀ Схема обоснования элементарна. Допустим, алгоритм строит в итоге базу $A_N = \{e_1, \ldots, e_N\} \subset \mathcal{M}$, причем $w(e_1) \geqslant \ldots \geqslant w(e_1)$, а решением служит база $\{f_1, \ldots, f_N\} \subset \mathcal{M}$, где f_j также упорядочены по убыванию $w(f_j)$.

Этого не может быть, поскольку *база, построенная жадным алгоритмом, доминирует над любой другой базой в смысле* $\boxed{w(e_j) \geqslant w(f_j)}$. Действительно, предположим противное. Пусть $j+1$ является первым индексом, для которого $w(e_{j+1}) < w(f_{j+1})$. Если в качестве множеств A и B, фигурирующих в формулировке условия **(ii)**, взять $A = \{e_1, \ldots, e_j\}$ и $B = \{f_1, \ldots, f_{j+1}\}$, то в силу **(ii)** алгоритм был обязан выбрать e_{j+1} так, что $w(e_{j+1}) \geqslant w(f_{j+1})$. ▶

Теорема 4.5.1 охватывает некоторое количество разнородных P-задач. Разнородными они выглядят, правда, пока не осознана их матроидная природа. Что касается полиномиальности числа шагов

[10] *И наоборот, если* $(E, \mathcal{M})$ — *не матроид, то найдется такая функция* $w(x) \geqslant 0$, *что жадный алгоритм приведет к неоптимальному множеству* $A \in \mathcal{M}$. Доказательство представляет собой несложное упражнение.

по выбору e_j, то она очевидна. Остается вопрос о вычислительной сложности проверок «разрешенного добавления элементов e_j» (не выводят ли они из $\mathcal{M}$), но он обычно легко решается [11].

Теория, как всегда, сильна ветвлением на частности.

Любому связному графу G можно сопоставить матроид $M(G)$, взяв в качестве E множество ребер G, а в качестве баз — ребра *остовных деревьев* [12]. Теорема 4.5.1 в этом случае свидетельствует об эффективности жадного алгоритма при решении задачи МИНИМАЛЬНОЕ ОСТОВНОЕ ДЕРЕВО. К определению $M(G)$ можно подойти с другой стороны, выбирая *циклы графа* G в качестве *циклов матроида* $M(G)$. По этой причине $M(G)$ называют *циклическим матроидом*.

На графе G конструируется также другой, двойственный матроид $M^*(G)$, называемый *коциклическим*, или *матроидом разрезов*. При определении $M^*(G)$ в качестве *циклов матроида* берутся *разрезы графа* G.

Разумеется, возникает мысль, не исчерпывают ли эти примеры всю глубину теории? В смысле, не существует ли для любого матроида изоморфный графический эквивалент? Не всегда. На тривиальных исключениях в данном контексте нет смысла останавливаться, но есть и нетривиальные области приложений [1].

Взаимодействие матроидов с теорией трансверсалей порождает большую серию комбинаторных результатов. *Трансверсалью*, или *системой различных представителей* на паре $(E, \mathcal{S})$, где E — непустое конечное множество, $\mathcal{S} = \{S_1, \ldots, S_n\}$ — семейство его непустых подмножеств [13], называется подмножество $\mathcal{T}$ множества E, состоящее из n различных элементов $e_j \in S_j$. Трансверсаль любого подсемейства семейства $\mathcal{S}$ называется *частичной трансверсалью* $\mathcal{S}$.

4.5.2. *Семейство частичных трансверсалей на заданной паре* $(E, \mathcal{S})$ *образует матроид (матроид трансверсалей).*

Иначе говоря, матроид образует пара $(E, \mathcal{T})$, где $\mathcal{T}$ конструируется с помощью $\mathcal{S}$. Это *теорема Эдмондса—Фалкерсона.*

---

[11] Например, при построении МИНИМАЛЬНОГО ОСТОВНОГО ДЕРЕВА на каждом шаге добавления нового ребра фиксируем подгруппу соединенных друг с другом вершин $V' \subset V$, после чего на следующем шаге исключаем ребра, обеими концами лежащие в V'. Полиномиальность такого контроля не вызывает сомнений.

[12] Граф G необязательно должен быть связным, но тогда вместо деревьев надо говорить о лесах.

[13] Не обязательно различных.

Жадный алгоритм в применении к матроиду трансверсалей полиномиально решает задачу *максимального паросочетания*.

4.6. Вариации МОД

Сейчас уже не то время, когда длинные списки NP-полных задач могли представлять общеобразовательный интерес. Такой этап прошел, и теперь соответствующие перечни интересны в основном узким специалистам. Ниже приводится несколько близких вариантов задачи МОД с другой целью. Интересно, как очень простая P-задача при «незначительных» шевелениях улетает сразу в NPC.

Напомним, МОД состоит в *поиске минимального остовного дерева реберно-взвешенного графа*. В варианте распознавания речь идет о существовании остовного дерева, суммарный вес ребер которого больше R. Задача решается полиномиально, причем быстро и просто. Следующие ее модификации NP-полны.

- Каждое ребро графа характеризуется весами двух типов, условно, красного и синего (длина дороги и стоимость). Существует ли остовное дерево, суммарный красный вес ребер которого $> R_r$, синий — $> R_b$? Иначе говоря, при добавлении второго критерия задача из P-оазиса возносится в самую сложную часть NP-класса.
- Существует ли остов, у которого степени всех вершин не превосходят заданного $K \geqslant 2$?
- Существует ли остов, у которого не менее K вершин степени 1?

Еще десяток задач в этой же окрестности есть в [7]. Странная история получается. Самая простая P-задача, но шаг вправо, шаг влево — и задача покидает (если покидает) P-территорию, причем оказывается не где-то «вблизи» от P-класса, а сразу в эмпиреях NP-полноты.

4.7. PSPACE-задачи

Беспросветная ситуация с проблемой «P или NP» заставляет искать выход из положения в перпендикулярных направлениях, из-за чего

возникает масса новых понятий. В основном, это издержки поиска. Циники, конечно, причины видят в защитах диссертаций [14], но тут главный корень в естественных «бомбежках по территориям». Плохо лишь, что воронки от случайных попаданий потом мешают свободно двигаться. А до уборки виртуального пространства руки не доходят.

Одна из напрашивающихся категорий переборных задач — PSPACE-*класс*, охватывающий задачи, решаемые с полиномиально ограниченной памятью. На вид, это не менее важно, чем полиномиально ограниченное время. Память, как и время, естественный вычислительный ресурс, и ею вполне логично интересоваться. Но класс PSPACE оказывается чересчур широк, в него входят P, NP, co-NP и другие задачи [15]. Разговор, конечно, теряет смысл в случае $P = PSPACE$, что в принципе не исключено, однако неравенство $P \neq PSPACE$ [16] выглядит более правдоподобно чем $P \neq NP$, — хотя не доказано.

Вокруг PSPACE есть своя маленькая теория, но довольно безжизненная. И дело не в том, что полиномиальные ограничения памяти малосущественны для практики (до них редко доходит), а в том, что разведка в направлении PSPACE ничего особенно не дала для «P или NP». А ведь главная надежда в такого рода прощупываниях — на побочные эффекты.

Тем не менее класс PSPACE устоял, оказавшись на перепутье магистральных дорог. Во-первых, обнаружились параллели с *интерактивными доказательствами* (разделы 6.8, 8.2) [17], где основной класс IP неожиданно совпал с PSPACE. Во-вторых, полиномиальные ограничения по памяти стали косвенно возникать то там,

[14] Хотя в этом тоже нет ничего плохого.

[15] NP и co-NP входят в PSPACE, ибо все NP-варианты можно перебрать в рамках полиномиальной памяти, очищаемой после каждой проверки.

[16] Возможное в том числе при $P = NP$.

[17] Довольно мощное и весьма интересное направление исследований, имеющее массу приложений, особенно в криптографии.

то здесь, — что подогрело интерес к структуре класса PSPACE. PSPACE-*полной* [18] оказалась, в частности, задача

БУЛЕВА ФОРМУЛА С КВАНТОРАМИ (БФК)

$$\Phi = (Q_1 x_1)(Q_2 x_2) \dots (Q_n x_n)\varphi(x_1, \dots, x_n), \qquad (4.6)$$

где Q_j *обозначают* $\forall$ *или* $\exists$, *и речь идет об истинности* Φ.

Если функция φ в (4.6) задана в форме КНФ, то БФК — это ВЫПОЛНИМОСТЬ с кванторами.

Ситуация в целом перекликается с NP-проблематикой. Вместо БФК PSPACE-полной задачей номер один естественно было бы считать МАШИНУ ТЬЮРИНГА (раздел 2.7), но теперь уже не с *полиномиальным будильником* (2.8), а с *полиномиальным сторожем*, который следит за памятью и, например, выключает машину, печатая 0, как только число использованных ячеек ленты достигает значения $p(|x|)$. Далее путь к БФК пролегает через аналог *теоремы Кука*.

Стандартный источник PSPACE-задач — игры типа шахмат, где входом служит начальная позиция, а вопрос относится к существованию выигрывающей стратегии у одного из игроков. «Тетрис», «сапер», «крестики-нолики», — все это в теории алгоритмов из несолидных развлечений трансформируется в респектабельные задачи. Имеет смысл также вынести на авансцену задачу **ТЕОРЕМА** о существовании доказательства у того или иного логического утверждения, иначе говоря, о разрешимости БФК. Если исходно о логике не упоминать, то подчеркивается универсальный характер ТЕОРЕМЫ, и создается впечатление об охвате комбинаторными задачами всей математики. Об этом уже говорилось в разделе 2.3, где за основу брались NP-задачи. Теперь же кажется правильнее — брать PSPACE-задачи, потому что теоремы почти всегда используют кванторы. Но что «правильнее» — так просто не скажешь. Зависит

[18] PSPACE-*полная задача* — по аналогии с NP-полной — определяется как универсальная PSPACE-задача, к которой полиномиально сводится любая другая задача из PSPACE-класса.

от теории и ее аксиоматики [6, т. 6], позволяющей доказывать некоторые утверждения с кванторами с помощью коротких цепочек рассуждений. Поэтому ТЕОРЕМА в данном контексте не вполне хороша, но ее присутствие напоминает все же о комбинаторном духе, пронизывающем всю математику.

4.8. Схемная сложность

Близкая по направленности к PSPACE теория связана с так называемой *схемной сложностью*, где логические функции

$$f(x_1, \ldots, x_n)$$

рассматриваются с точки зрения их схемной реализации в какомлибо базисе типа «*не*, *и*, *или*». Тем самым комбинаторным задачам распознавания сопоставляются схемы вычисления, формулы, по характеристикам которых (количество элементов, иерархическая глубина) делаются попытки судить о сложности задач. Замысел выглядит многообещающе, но в итоге получаются от жилетки рукава.

В русле соответствующей теории возникает еще один *класс* P/poly, определяемый как совокупность предикатов $f(x_1, \ldots, x_n)$, представимых в некотором логическом базисе формулой полиномиальной сложности (по числу используемых элементов базиса). Сопутствующий вопрос: куда отнести трудоемкость поиска соответствующей формулы — включить в определение «класса» или вынести за скобки? Так или иначе, главная теорема здесь: «$P \subset P/poly$», — что устанавливается совсем просто, и открывает дорогу к новому тупику. Оптимистически настроенные специалисты, дай им бог удачи, тоже есть, но перевешивающий вопрос в том, как не заразить этим оптимизмом остальную часть аудитории.

4.9. Интерактивные протоколы

Модное и перспективное направление в информатике — *интерактивные доказательства*, где речь идет о решении дискретных

задач двумя участниками («Verifier — Prover», «Алиса — Боб», «Оракул — Вычислитель») в рамках упорядоченного диалога. Собственно, вариант *интерактивной системы* уже возникал при определении NP-класса, где машина Тьюринга дополнялась *оракулом* — блоком угадывания, способным генерировать *удостоверение*, которое затем «полиномиально проверялось». Тандем машины с оракулом был по сути *интерактивной системой — недетерминированной машиной Тьюринга*, которая в силу притянутой за уши терминологии решала NP-задачи за полиномиальное время. Но это особого удивления не вызывало, потому что протокол взаимодействия машины с оракулом не порождал каких-либо интересных и неожиданных явлений. Именно поэтому в определении NP-задач мы отдавали предпочтение обыкновенной полиномиальной проверяемости, не наводящей тень на плетень, как недетерминированные вычисления.

Но причуды, бывает, дают поразительные результаты — и здесь как раз тот случай. На пресное взаимодействие двух сторон при решении NP-задачи [19] можно взглянуть шире. Сообща решать задачу позволительно в более изобретательной манере, рассматривая процесс как игру двух лиц, скажем, *Оракула* с *Вычислителем*. Устанавливая те или иные *правила игры* — иначе говоря, *регламент*, *протокол*, — можно добиваться разных целей, как оказывается, довольно неожиданных. Вот пример.

Оракулу $\mathcal{O}$ известен изоморфизм φ графов G_0 и G_1. Но он посылает *Вычислителю* $\mathcal{V}$ граф [20] $H = \psi(G_0)$, где ψ — некий другой изоморфизм, не равный φ. $\mathcal{V}$ бросает монетку, и просит изоморфизм либо $H \sim G_0$, либо $H \sim G_1$. В первом случае $\mathcal{O}$ посылает ψ, во втором — $\varphi \cdot \psi^{-1}$. Таких партий разыгрывается N штук.

Если φ — действительно изоморфизм, $G_0 \sim G_1$, то все проверки будут положительны. Если φ — блеф, с вероятностью 2^{-N} хотя бы одна проверка обнаружит «липу» [21]. При этом вероятность ошибки 2^{-N} может быть сделана сколь угодно малой выбором N независимо от размерности задачи.

При непонимании «зачем это нужно» — сценарий кажется вычурным и скучным. А трюк на самом деле выдающийся по кон-

[19] Одна сторона предъявляет решение, вторая — тупо его проверяет.

[20] Либо $H = \psi(G_1)$.

[21] Та проверка, в которой $\mathcal{V}$ попросит $H \sim G_1$, тогда как $H = \psi(G_0)$.

трасту незамысловатости устройства и уникальности эффекта. Дело в том, что *Оракул* убедил *Вычислителя* в $G_0 \sim G_1$, так и не огласив самого изоморфизма φ. В другом изложении: *Оракул* доказал *Вычислителю* знание изоморфизма φ. Доказал, но по каналу связи не послал никакой информации, которая бы позволила «вражеской» стороне узнать что-либо о φ. Поэтому, если φ пароль, диалог можно вести даже в открытую, — что служит примером *системы с нулевым разглашением*, и представляет собой революционный шаг в криптографии (раздел 6.8).

В зависимости от решаемой задачи целесообразные способы организации подобного взаимодействия сильно разнятся [22]. Совокупность задач, которые могут быть полиномиально решены с помощью *интерактивных доказательств* (*взаимодействий*), образует *класс* IP.

О доказательствах здесь говорят потому, что игра всегда протекает в рамках одной и той же схемы: *Оракул* доказывает, *Вычислитель* проверяет. Конкретные ситуации отличаются друг от друга содержательными оттенками, вслед за которыми часто меняется терминология. Скажем, в рассмотренном выше примере, термин «*Оракул*», вообще говоря, не совсем правомерен, и тандем типа «*Алиса — Боб*» выглядит более уместным. Ибо приготовить задачу об изоморфизме двух графов любой размерности, а потом перенумеровывать вершины — по силам ученику средней школы. Мистические способности, и даже просто особые вычислительные, здесь не требуются.

А вот интерактивные схемы для задач из co-NP (глава 8) требуют запредельных вычислительных способностей у одной из сторон, и там понятие *Оракула* уже к месту. На Западе, как правило, отдают предпочтение паре «*Verifier — Prover*».

В том случае, когда одна из сторон обязана иметь чудодейственные вычислительные таланты, анализ выходит вроде бы за пределы реального, и не имеет смысла. Но это не совсем так, ибо решению

[22] См. главу 8.

практических задач действительно не помогает, но работает как теоретический инструмент, поскольку возникают альтернативные определения классов. Например, оказывается

$$\mathrm{IP} = \mathrm{PSPACE},$$

что отчасти неожиданно[23], и позволяет изучать PSPACE-задачи с позиции интерактивных схем. Тем самым полиномиальные ограничения памяти трансформируются в полиномиальную трудоемкость по времени, а это вскрывает некоторые особенности класса PSPACE общего характера.

Особые надежды связываются с так называемым *классом* PCP (***P**robabilistically **C**heckable **P**roof System*), представляющим собой особую разновидность интерактивных систем — раздел 8.3. Знаменитая PCP-*теорема* устанавливает равенство

$$\mathrm{NP} = \mathrm{PCP},$$

позволяя тем самым обозревать NP-класс с другой точки зрения, что порождает определенные надежды.

4.10. Релятивизация и оракулы

В теории алгоритмов *оракул* $\mathcal{O}$ — это «черный ящик», мгновенно выдающий по запросу машины M значения функции $\mathcal{O}(x)$. Тандем машины с оракулом $M^{\mathcal{O}}$ представляет собой новый вычислительный прибор, обладающий иногда более широкими возможностями.

Феномен отчасти мистический. В теории широко используется, например, оракул, решающий *проблему останова* машины Тьюринга, что до некоторой степени нарушает этикет, ибо такая возможность допускается в той же области, в которой была обоснована принципиальная неразрешимость *проблемы останова* [6, т. 6]. Однако, как говорится, собака лает — караван идет. Плохо ли,

[23] Включение IP $\subset$ PSPACE очевидно, поскольку *Вычислитель*, перед каждой новой полиномиальной проверкой может стирать память. Обратное включение PSPACE $\subset$ IP устанавливается несколько сложнее.

хорошо ли, но понятие оракула позволяет различать оттенки в невидимой области — по крайней мере, создает такую иллюзию.

Возможность обращения к оракулу $\mathcal{O}(x)$, как говорят, *релятивизует* теорию алгоритмов. Возможности счета расширяются, но это никак не влияет на классические результаты о вычислимых функциях, теперь уже «$\mathcal{O}$-*вычислимых*». Возникают $\mathcal{O}$-*универсальные функции*; $\mathcal{O}$-*неразрешимые* и $\mathcal{O}$-*перечислимые множества* и т. п.[24] При этом не меняются не только результаты, но и доказательства. Собственно, результаты не меняются как раз потому, что остаются работоспособны все старые схемы рассуждений. Но это в общей теории алгоритмов, где сложность вычислений не принимается в расчет.

При изучении временно́й трудоемкости ситуация другая. Оракулы, не затрагивающие вычислимости функций как таковой, способны деформировать сложностные классы и ломать некоторые типы рассуждений. Не все, конечно. Например, *метод диагонализации* [6, т. 6] устойчив к использованию оракулов — и если бы с его помощью удалось доказать $\mathrm{P} \neq \mathrm{NP}$, то это означало бы также $\mathrm{P}^{\mathcal{O}} \neq \mathrm{NP}^{\mathcal{O}}$ для любого оракула[25]. Аналогично (бы)

$$\mathrm{P} = \mathrm{NP} \quad \Rightarrow \quad \mathrm{P}^{\mathcal{O}} = \mathrm{NP}^{\mathcal{O}}.$$

Поэтому на изучение оракульных вариантов $\mathrm{P}^{\mathcal{O}} \stackrel{?}{=} \mathrm{NP}^{\mathcal{O}}$ возлагались большие надежды. Но тут, как снег на голову, выяснилось[26] *существование таких оракулов* $\mathcal{A}$ *и* $\mathcal{B}$, *что*

$$\mathrm{P}^{\mathcal{A}} = \mathrm{NP}^{\mathcal{A}}, \tag{4.7}$$

$$\mathrm{P}^{\mathcal{B}} \neq \mathrm{NP}^{\mathcal{B}}, \tag{4.8}$$

откуда сразу стало ясно, что ни диагональный метод, ни другие рассуждения, *устойчивые к релятивизации*, в рассматриваемой области не проходят. Проблема $\mathrm{P} \stackrel{?}{=} \mathrm{NP}$ требует иных мер.

---

[24] Если $\mathcal{O}(x)$ решает проблему останова, то проблема $\mathcal{O}$-останова — снова неразрешима.

[25] Здесь $\mathrm{P}^{\mathcal{O}}$ — полиномиальный класс задач распознавания при вычислениях на оракульной машине $M^{\mathcal{O}}$, а $\mathrm{NP}^{\mathcal{O}}$ — совокупность полиномиально проверяемых на $M^{\mathcal{O}}$ задач.

[26] *Baker T., Gill J., Solovay R.* Relativizations of the $\mathrm{P} \stackrel{?}{=} \mathrm{NP}$ question // SIAM J. Comp. 1975. **4**. 431–442.

◀ Для обоснования (4.7), на первый взгляд, годится оракул, решающий любую задачу. Тогда все вычисляется, вроде бы, за один шаг, и $NP^{\mathcal{A}} = P^{\mathcal{A}}$. Но оракул, способный решить любую задачу, внутренне противоречив, как *множество всех множеств* [6, т. 6]. Тем не менее идея в своей основе разумна. Оракул имеет смысл выбрать самый мощный, но не переступая ту грань, за которой все мешается в одну кучу. Таков любой оракул, решающий какую-нибудь PSPACE-полную задачу, например, БФК. В этом случае $NP^{\mathcal{A}} \subset PSPACE$, потому что полиномиальная [27] проверяемость с «БФК-оракулом» выполнима с полиномиальной памятью без оракула. А включение $PSPACE \subset P^{\mathcal{A}}$ имеет место, поскольку с «оракулом БФК» за полиномиальное время можно решить любую PSPACE-задачу. Таким образом,

$$NP^{\mathcal{A}} \subset PSPACE \subset P^{\mathcal{A}},$$

обратное же включение $P^{\mathcal{A}} \subset NP^{\mathcal{A}}$ справедливо вообще для любого оракула, что доказывает (4.7). Обоснование (4.8) более громоздко. ▶

Разумеется, (4.7) имеет смысл доказывать в случае $NP \neq P$, иначе и так ясно. Если $NP = P$, то в качестве оракула $\mathcal{A}$ достаточно взять какую-нибудь бесполезную функцию типа $1 + 0 = 1$. А (4.8) можно не доказывать, если $NP \neq P$, поскольку опять-таки выручает бесполезный оракул. Поэтому релятивизация классов может трансформировать равенства в неравенства, и наоборот. Это касается не только пары (P, NP), но и других взаимоотношений типа (NP, co-NP), $(P, NP \bigcap \text{co-NP})$.

Поначалу открытие возможностей (4.7) и (4.8) вселяло оптимизм по поводу $P \stackrel{?}{=} NP$, но постепенно стало ясно, что приумножение знаний здесь происходит только в понимании неработоспособности диагонального метода. Но это, между прочим, и так, как на ладони. Что же касается хоть какого-то света на первичный вопрос $P \stackrel{?}{=} NP$, то — никакого.

4.11. Рамсеевские задачи

На фоне расхожих представлений об окружающей действительности какие-нибудь черные дыры во Вселенной оказываются чудовищной аномалией, которая в голову не укладывается. Таковы некоторые задачи во вселенной комбинаторных явлений.

Среди любых шести человек всегда найдутся трое, либо знакомые друг с другом, либо незнакомые. Другими словами, в любом

[27] По времени.

графе с шестью вершинами всегда есть либо 3-клика, либо 3-антиклика[28]. Это *теория Рамсея* в миниатюре.

Философски приукрашенная идея *Рамсея* смотрится более внушительно: *всякая достаточно большая структура содержит регулярную подструктуру заданного размера.* Конкретные результаты без претензий на многозначительность — менее впечатляющи, но по сути остаются фундаментальными. Например: *если число n объектов в некоторой совокупности S достаточно велико и каждые два объекта связаны одним из m отношений, то по любому априори заданному k можно указать подмножество $K \subset S$ мощности k, в котором все объекты связаны отношением одного типа.*

В проекции на граф G с n вершинами это означает, что *при любом k и достаточно большом n в графе всегда есть либо k-клика, либо k-антиклика.* И даже более того:

4.11.1. *По любым k и l можно указать такое N, что в графе с $n \geqslant N$ вершинами всегда есть либо k-клика, либо l-антиклика. Минимальное*

$$N = R(k, l) \tag{4.9}$$

называют числом Рамсея[29].

Числа Рамсея с большим трудом поддаются вычислению, ибо целенаправленных алгоритмов (скажем, рекуррентных) практически нет, а перебор сталкивается с астрономическими масштабами уже на первых шагах. Пока известно кот наплакал:

$$R(3,3) = 6, \quad R(3,4) = 9, \quad R(3,5) = 14, \quad R(3,6) = 18,$$
$$R(3,7) = 23, \quad R(3,8) = 28, \quad R(3,9) = 36, \quad R(4,4) = 18.$$

Еще имеется несколько оценок:

$$R(4,5) \in [25,28], \quad R(4,6) \in [34,45], \quad R(5,5) \in [38,55],$$
$$R(5,6) \in [38,94], \quad R(6,6) \in [102,178].$$

---

[28] Иначе говоря, в любом графе с шестью вершинами либо в его дополнении всегда есть 3-клика. Перебор требует рассмотрения $\sim 2^{15}$ вариантов.

[29] Если между парами вершин возможны m типов отношений, а не только «есть ребро, нет ребра», число Рамсея зависит от трех параметров и обозначается $R = R(k, l, m)$. В (4.9) $m = 2$.

Сами числа R не так уж велики [30], но трудоемкость их вычисления чрезвычайна (гипотетически). Задача распознавания

$$«R(k,l) \textit{ больше } n?» \tag{4.10}$$

входит в co-NP, но не является NP-задачей, по-видимому. Утверждение, возможно, не вполне корректно [31]. Под «(4.10) $\in$ co-NP» подразумевается эквивалентность (4.10) задаче из co-NP, ибо для положительного ответа на вопрос (4.10) достаточно указать такой n-граф, граф G с n вершинами, чтобы он не имел k-клики, а его дополнение — l-клики. Граф в качестве *удостоверения* выглядит приемлемо, но для проверки G надо перебрать все его k-подграфы, и все l-подграфы у дополнения G, что является дополнительной задачей к проблеме:

$$«\textit{Содержит ли } n\textit{-граф } G \textit{ } k\textit{-клику или } l\textit{-антиклику}?». \tag{4.11}$$

Задача (4.11) вносит определенную интригу, ибо для решения в случае $n \geqslant R(k,l)$ не требует вычислений, но таблица $R(k,l)$ известна лишь Всевышнему, если известна. Собственно, никакой такой интриги в самой задаче (4.11) нет, а есть предположительная неприятность в одном из ее срезов:

$$«\textit{Содержит ли } n\textit{-граф } G \textit{ } \lambda(n)\textit{-клику или } \lambda(n)\textit{-антиклику}», \tag{4.12}$$

где функция $\lambda(n)$ полиномиально вычисляется, и

$$n = \lambda^{-1}(k) \geqslant R(k,k). \tag{4.13}$$

В случае (4.13) задача (4.12) имеет ответ «да» $\forall n$, и, таким образом, принадлежит классу P. Однако если неравенство (4.13) недоказуемо, то нельзя утверждать, что алгоритм, всегда дающий ответ «да», решает задачу — и принадлежность (4.12) классу P принципиально остается под вопросом.

Другой потенциальный вариант — существование полиномиально вычислимой функции $\lambda(\cdot)$, которая удовлетворяет неравенству (4.13), но ее невозможно конкретно указать. Возникает иная метаморфоза: в λ-семействе задач (4.12) существует полиномиально разрешимая задача, но какая именно — принципиально неизвестно.

Реальных парадоксов мы, конечно, не получили, но сослагательное наклонение здесь имеет свои мотивы (см. раздел 3.1). Проблема $\mathrm{P} \stackrel{?}{=} \mathrm{NP}$ нуждается в углубленном рассмотрении понятий P- и NP-классов. И тут плоды фантазии противодействуют окостенению представлений.

---

[30] К перечисленному можно добавить общее неравенство $R(k,l) \leqslant C_{k+l-2}^{k-1}$.

[31] Потому что дополнительная к (4.10) задача «$R(k,l) \leqslant n$?», похоже, также не принадлежит NP.

Глава 5

Линейное программирование

Философски выражаясь, экстремальные задачи нужны не для того, чтобы их решать. Они существуют, дабы служить позицией и направлять мысль. Однако рассказывать приходится — как будто все наоборот. Иногда и жить так приходится.

Полиномиальный алгоритм *Хачияна* [20] для задачи ЛП не только решил наболевший вопрос, но и взорвал инерционные представления об NP-проблематике. Решение было достигнуто при игнорировании комбинаторной природы задачи, что обозначило принципиально новый поворот умонастроений, и угасающие было надежды (насчет $\mathbf{P} \stackrel{?}{=} \mathbf{NP}$) всколыхнулись вновь.

5.1. Постановка задачи

Общепринятая постановка задачи линейного программирования,

$$\langle c, x \rangle \to \max, \quad Ax \leqslant b, \quad x \geqslant 0, \tag{5.1}$$

характеризуется линейностью *целевой функции*

$$\langle c, x \rangle = \sum_j c_j x_j$$

и ограничений

$$\sum_j a_{ij} x_j \leqslant b_i.$$

Матрица $A = [a_{ij}]$ имеет размер $m \times n$.

Широко распространены и другие варианты постановки ЛП, сводящиеся друг к другу некоторыми ухищрениями. В (5.1) ограничение $x \geqslant 0$ иногда не фигурирует.

Сами числа R не так уж велики [30], но трудоемкость их вычисления чрезвычайна (гипотетически). Задача распознавания

$$«R(k,l) \textit{ больше } n?» \tag{4.10}$$

входит в co-NP, но не является NP-задачей, по-видимому. Утверждение, возможно, не вполне корректно [31]. Под «(4.10) $\in$ co-NP» подразумевается эквивалентность (4.10) задаче из co-NP, ибо для положительного ответа на вопрос (4.10) достаточно указать такой n-граф, граф G с n вершинами, чтобы он не имел k-клики, а его дополнение — l-клики. Граф в качестве *удостоверения* выглядит приемлемо, но для проверки G надо перебрать все его k-подграфы, и все l-подграфы у дополнения G, что является дополнительной задачей к проблеме:

$$«\textit{Содержит ли } n\textit{-граф } G \textit{ } k\textit{-клику или } l\textit{-антиклику?}». \tag{4.11}$$

Задача (4.11) вносит определенную интригу, ибо для решения в случае $n \geqslant R(k,l)$ не требует вычислений, но таблица $R(k,l)$ известна лишь Всевышнему, если известна. Собственно, никакой такой интриги в самой задаче (4.11) нет, а есть предположительная неприятность в одном из ее срезов:

$$«\textit{Содержит ли } n\textit{-граф } G \textit{ } \lambda(n)\textit{-клику или } \lambda(n)\textit{-антиклику}», \tag{4.12}$$

где функция $\lambda(n)$ полиномиально вычисляется, и

$$n = \lambda^{-1}(k) \geqslant R(k,k). \tag{4.13}$$

В случае (4.13) задача (4.12) имеет ответ «да» $\forall n$, и, таким образом, принадлежит классу P. Однако если неравенство (4.13) недоказуемо, то нельзя утверждать, что алгоритм, всегда дающий ответ «да», решает задачу — и принадлежность (4.12) классу P принципиально остается под вопросом.

Другой потенциальный вариант — существование полиномиально вычислимой функции $\lambda(\cdot)$, которая удовлетворяет неравенству (4.13), но ее невозможно конкретно указать. Возникает иная метаморфоза: в λ-семействе задач (4.12) существует полиномиально разрешимая задача, но какая именно — принципиально неизвестно.

Реальных парадоксов мы, конечно, не получили, но сослагательное наклонение здесь имеет свои мотивы (см. раздел 3.1). Проблема P $\stackrel{?}{=}$ NP нуждается в углубленном рассмотрении понятий P- и NP-классов. И тут плоды фантазии противодействуют окостенению представлений.

---

[30] К перечисленному можно добавить общее неравенство $R(k,l) \leqslant C_{k+l-2}^{k-1}$.

[31] Потому что дополнительная к (4.10) задача «$R(k,l) \leqslant n?$», похоже, также не принадлежит NP.

Глава 5

Линейное программирование

Философски выражаясь, экстремальные задачи нужны не для того, чтобы их решать. Они существуют, дабы служить позицией и направлять мысль. Однако рассказывать приходится — как будто все наоборот. Иногда и жить так приходится.

Полиномиальный алгоритм *Хачияна* [20] для задачи ЛП не только решил наболевший вопрос, но и взорвал инерционные представления об NP-проблематике. Решение было достигнуто при игнорировании комбинаторной природы задачи, что обозначило принципиально новый поворот умонастроений, и угасающие было надежды (насчет $\mathbf{P} \stackrel{?}{=} \mathbf{NP}$) всколыхнулись вновь.

5.1. Постановка задачи

Общепринятая постановка задачи линейного программирования,

$$\langle c, x \rangle \to \max, \quad Ax \leqslant b, \quad x \geqslant 0, \tag{5.1}$$

характеризуется линейностью *целевой функции*

$$\langle c, x \rangle = \sum_j c_j x_j$$

и ограничений

$$\sum_j a_{ij} x_j \leqslant b_i.$$

Матрица $A = [a_{ij}]$ имеет размер $m \times n$.

Широко распространены и другие варианты постановки ЛП, сводящиеся друг к другу некоторыми ухищрениями. В (5.1) ограничение $x \geqslant 0$ иногда не фигурирует.

Варианты с минимизацией $\langle c, x\rangle$ ничего нового в постановку не привносят, ибо сводятся к максимизации после замены $\langle c, x\rangle$ на $\langle -c, x\rangle$. Каноническая модификация

$$\langle c, u\rangle \to \max, \quad Bu = b, \quad u \geqslant 0 \tag{5.2}$$

получается из (5.1) добавлением фиктивных параметров $z_i \geqslant 0$, таких что $\sum\limits_j a_{ij}x_j + z_i = b_i$. В переменных $u = \begin{bmatrix} x \\ z \end{bmatrix}$ задача (5.1) принимает вид (5.2).

Вариант распознавания: *существует ли для данного* $K \geqslant 0$ *вектор из* $X = \{x\colon Ax \leqslant b,\ x \geqslant 0\}$, *такой что* $\langle c, x\rangle \geqslant K$? Таким образом, распознавание в данном случае — есть *задача* ЛН (ЛИНЕЙНОЕ НЕРАВЕНСТВО) о разрешимости системы неравенств

$$Ax \leqslant b, \quad x \geqslant 0, \quad \langle c, x\rangle \geqslant K. \tag{5.3}$$

Требование положительности икса в (5.3) можно загнать в общий список. Например, $Ax > 0$, $x > 0$ равносильно задаче о разрешимости $Bz > 0$ с матрицей $B = \begin{bmatrix} A \\ I \end{bmatrix}$. К однородному неравенству от $Ax \geqslant b$ легко перейти:

$$\begin{bmatrix} a_{11} & \dots & a_{1n} & -b_1 \\ a_{21} & \dots & a_{2n} & -b_2 \\ \dots & \dots & \dots & \dots \\ a_{m1} & \dots & a_{mn} & -b_m \\ 0 & \dots & 0 & 1 \end{bmatrix} \cdot x > 0.$$

◄ Если последнее неравенство имеет решение $\{x_1, \dots, x_{n+1}\}$, то решением является и

$$\left\{\frac{x_1}{x_{n+1}}, \dots, \frac{x_n}{x_{n+1}}, 1\right\}. \quad \blacktriangleright$$

Пониманию ЛП способствует геометрический взгляд, согласно которому каждое из ограничений

$$\sum_j a_{ij}x_j \leqslant b_i, \quad x_j \geqslant 0,$$

задачи (5.1) определяет в $\mathbb{R}^m$ сдвинутое полупространство, пересечение которых дает *многогранник допустимых решений*, ограниченный или неограниченный, возможно пустой. Линейная целевая

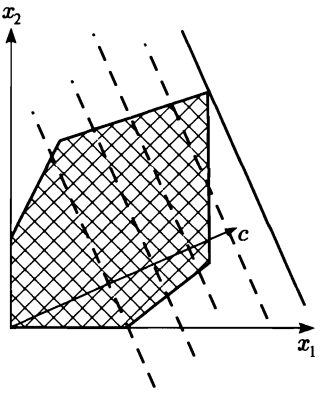


Рис. 5.1

функция $\langle c, x\rangle$ растет при движении точки x вдоль c. Поверхностями постоянного уровня, на которых функция $\langle c, x\rangle$ принимает одно и то же значение, являются гиперповерхности

$$\langle c, x\rangle = \gamma,$$

ортогональные вектору c — пунктирные прямые на рис. 5.1.

Обычно считают, что решение ЛП, если существует, лежит в некоторой вершине допустимого многогранника. Это не совсем точно. Неограниченный многогранник может вообще не иметь вершин (клин). Поэтому утверждение о необходимости расположения решения в одной из вершин допустимого многогранника X справедливо в предположении, что X не содержит прямых (например, в случае ограниченности X). Решение, лежащее в вершине, не обязательно единственно. Когда одна из «дальних» граней допустимого многогранника ортогональна вектору c, — все точки этой грани (в том числе вершины) будут решениями.

Задачу ЛП традиционно относили к дискретным (несмотря на непрерывность аргумента x) по той причине, что ее решение заведомо достигается в одной из вершин *допустимого многогранника* $X = \{x : Ax \leqslant b,\ x \geqslant 0\}$. Потом было осознано, что *длину входа* надо оговаривать. Поэтому ЛП как задача комбинаторного анализа предполагает либо целочисленность коэффициентов, либо рациональность при заданной точности, — что равносильно, ибо рациональные коэффициенты в $\sum_j a_{ij}x_j \leqslant b_i$ переходят в целые

после умножения неравенств на подходящее целое число N. Аргумент x рационален в любом случае.

> *Что касается дискретных мотивов — подчеркнем еще раз. Решение* (5.1) *лежит в одной из вершин допустимого многогранника, — и поэтому даже при непрерывном* x *проблема дискретна. Но в контексте комбинаторного анализа задача ЛП всегда звучит* ***с обязательной оговоркой о целочисленности либо рациональности всех коэффициентов****. В противном случае теряется связь с длиной входа. В случае иррациональных коэффициентов длина входа становится бесконечной.*

В ЛП принципиальна определенная нечувствительность решения к данным задачи. Достаточно очевидно, что при малых изменениях вектора c *оптимальное решение будет достигаться в той же самой вершине. Пределы нечувствительности решения к изменению критерия могут быть достаточно велики. Например, в ситуации, изображенной на рис.* 5.1, *решение одно и то же при любом положительном векторе* c.

Двойственность. *Двойственной* к

$$\langle c, x\rangle \to \max, \quad Ax \leqslant b, \quad x \geqslant 0,$$

считается задача

$$\langle b, y\rangle \to \min, \quad A^T y \geqslant c, \quad y \geqslant 0, \tag{5.4}$$

где A^T обозначает транспонированную матрицу[1]. Взаимосвязь задач относительно подробно рассмотрена в [6, т. 7]. Остановимся на стержневых фактах.

5.1.1. *Пусть* x *и* y *— допустимые векторы прямой* (5.1) *и двойственной задачи. Тогда*

$$\langle c, x\rangle \leqslant \langle b, y\rangle.$$

Из 5.1.1 вытекает следующий критерий оптимальности.

5.1.2. *Пусть* x^* *и* y^* *— допустимые векторы прямой и двойственной задач, причем* $\langle c, x^*\rangle = \langle b, y^*\rangle$ *Тогда векторы* x^*, y^* *являются решениями соответствующих задач.*

[1] Вместо $A^T y \geqslant c$ иногда пишут $yA \geqslant c$.

Утверждение 5.1.1 дает также ключ к пониманию основной теоремы двойственности: *если прямая и двойственная задача имеют допустимые векторы, то обе они имеют решения.*

5.1.3. Теорема. *Для того чтобы допустимые векторы прямой и двойственной задач были решениями этих задач, необходимо и достаточно выполнение следующих условий дополняющей нежесткости:*

$$\begin{cases} y_i^* = 0, & \text{если} \quad \sum\limits_j a_{ij} x_j^* < b_i, \\ x_j^* = 0, & \text{если} \quad \sum\limits_i a_{ij} y_i^* > c_j. \end{cases}$$

Понятно, что в решении исходной задачи часть неравенств обращается в равенства — эти ограничения называют *насыщающими* или *активными*, в противоположность *ненасыщающим* или *пассивным*. Решение задачи (5.1) было бы предельно простым, если бы было известно, какие ограничения активные. В этом случае было бы достаточно неравенства, соответствующие активным ограничениям, заменить на равенства — и решить получившуюся систему линейных уравнений. Но решение двойственной задачи как раз однозначно указывает, какие ограничения активные. Поэтому, если известно двойственное решение, исходную задачу можно считать «решенной».

5.2. Предыстория комбинаторного варианта

Поскольку решение задачи ЛП достигается в вершине *допустимого многогранника*, перебор — в случае существования решения — гарантирует успех за конечное число шагов. Однако в рядовых задачах число вершин оказывается астрономическим.

При описании допустимого многогранника неравенствами

$$Ax \leqslant b, \quad x \geqslant 0,$$

где A матрица $m \times n$, — имеется $m+n$ скалярных ограничений. Вершина определяется пересечением m гиперплоскостей, т. е. из $m+n$ уравнений надо выбрать m линейно независимых. Поэтому $C_{m+n}^n = \dfrac{(m+n)!}{m!n!}$ в типичных ситуациях приблизительно правильно (в процентном отношении) оценивает число вершин, которое

быстро растет с увеличением размерности. Например, при $n = 40$, $m = 20$ число вершин имеет порядок 10^{11}.

Для практического решения задачи ЛП до сих пор используется *симплекс-метод*, где на каждом шаге осуществляется целенаправленный переход в соседнюю вершину, в которой значение $\langle c, x\rangle$ лучше предыдущего[2]. Алгоритм, как правило, хорошо работает[3], но есть примеры, когда по необходимости перебираются почти все вершины допустимого многогранника. Поэтому фактически алгоритм переборный, и долгое время многочисленные попытки найти способ решения задачи ЛП, который бы заведомо избавлял от экспоненциального роста объема вычислений, не давали результата. Начинало казаться, что ЛП $\notin$ P.

В то же время, в силу своих особенностей, ЛП выглядела кандидатом на роль «*не* NP-*полной задачи* $\notin$ P» по следующей причине. *Дополнение* ЛП,

$$\langle c, x\rangle \leqslant K \quad \textit{для всех} \quad x \in P, \tag{5.5}$$

также ЛП-*задача*[4], что устанавливается следующим образом.

◀ Варианты распознавания и оптимизации задачи ЛП в ракурсе полиномиальной сводимости совпадают (см. теорему 1.2.1). Поэтому, решая двойственную задачу (5.4) и сравнивая значение оптимума $\langle b, y^*\rangle$ со значением K, получаем возможность судить, в силу утверждения 5.1.1, о справедливости (5.5). ▶

Таким образом, если бы вдруг ЛП оказалась NP-полной задачей, то теорема 4.1.1 гарантировала бы NP = co-NP, что до сих пор выглядит сомнительным.

Заметим, наконец, что проблема с полиномиальностью ЛП, высветившаяся из-за того, что симплекс-метод в некоторых ситуациях вынужден перебирать все вершины, имеет совсем другой источник. Как уже отмечалось, вариант распознавания ЛП — есть

[2] Механизм вычислений разбирается в любом учебнике по линейному программированию.

[3] В рамках естественных вероятностных моделей *симплекс-метод* полиномиален в среднем [22].

[4] Смысл утверждения уточняет нижеследующее доказательство.

задача ЛН (ЛИНЕЙНОЕ НЕРАВЕНСТВО) о разрешимости системы линейных неравенств (5.3), которую после переобозначения можно записать в виде

$$Bx \leqslant a.$$

Поэтому первый же шаг симплекс-метода — поиск любого допустимого решения — не проще задачи целиком [5)]. И будь в наших руках полиномиальный алгоритм для решения задачи ЛН — эффективное решение ЛП получалось бы в рамках стандартной дихотомической процедуры (теорема 1.2.1).

5.3. Эффекты ограниченной точности

При задании параметров и переменных рациональными числами линейные задачи меняют свое лицо, ибо возникают полиномиальные зависимости *битовых размеров* выхода от входа, т. е. решения от условия. А поскольку битовый размер ограничивает сами числа, появляются принципиальные для вычислений возможности априорных оценок решения задач ЛН и ЛП.

Чтобы уточнить сказанное, необходимо вернуться к понятию *длины описания*, т. е. *размера*, которому в разделе 1.2 было уделено три строчки, каковых в принципе достаточно, но удобнее все же исходить из более конкретных представлений.

При идиосинкразии к неопределенности за длину целого числа p можно принять число битов, $L(p) = \lceil \log_2(1+|p|) \rceil$, необходимое для записи этого числа в двоичной системе. Длина рационального числа $\alpha = p/q$, где p и q — взаимно просты, в рамках той же идеологии оказывается равной

$$L(\alpha) = L(/) + L(p) + L(q), \tag{5.6}$$

где $L(/)$ — число битов, необходимых для записи «деления», — зависит от принятых соглашений о кодировании знаков в используемом алфавите.

[5)] В рамках полиномиальной эквивалентности.

Под тем же углом зрения длина *вектора* $x = \{x_1, \ldots, x_m\}$:

$$L(x) = m \cdot L(,) + L(x_1) + \ldots + L(x_m), \tag{5.7}$$

длина $(m \times n)$-*матрицы* $A = [a_{ij}]$:

$$L(A) = mn \cdot L(\cdot, \cdot) + \sum_{i,j} L(a_{ij}). \tag{5.8}$$

Величины $L(,)$, $L(\cdot, \cdot)$ характеризуют битовые затраты на вспомогательную информацию (запятая, расположение элементов матрицы). Но педантичность здесь в большинстве случаев излишня. Интерес, главным образом, представляет асимптотическое поведение *размеров описания*, поэтому в миллион раз больше или меньше — роли не играет. Соответственно, в определении размеров возможен значительный люфт; $L(,)$, $L(\cdot, \cdot)$ в (5.7), (5.8) — без какого бы то ни было ущерба можно опустить. Тем не менее определенные ситуации удобнее не оставлять во взвешенном состоянии, выбирая что-либо конкретное типа (5.6)–(5.8).

Уравнение $px = q$ с целыми взаимно простыми p и q имеет решение $x = p/q$, длина описания которого определяется в соответствии с (5.6), (5.7). В случае рациональных p и q в $px = q$ длина «выхода» $x = p/q$ также легко считается. Не возникает особых затруднений и в случае линейного уравнения $Ax = b$. Рецепт извлекается из (5.6)–(5.8). Во-первых, как показывают элементарные соображения, размер детерминанта [6]

$$L(\det A) \leqslant L(A). \tag{5.9}$$

Во-вторых, *найдется решение уравнения* $Ax = b$, *размер которого не превосходит размера расширенной матрицы* [7] $[A, b]$, *в том числе для прямоугольных матриц* A.

---

[6] Если элементы матрицы $a_{ij} = p_{ij}/q_{ij}$, то знаменатель определителя не превосходит произведения всех q_{ij}, которое, в свою очередь, $\leqslant 2^{L(A)-1}$, что в итоге дает (5.9). При этом значение самого детерминанта $|\det A| \leqslant 2^{L(A)}$.

[7] По той же причине две прямые не могут пересекаться слишком далеко, если их коэффициенты наклона задаются ограниченным количеством цифр.

◀ В случае невырожденной матрицы A решение $Ax = b$ по *правилу Крамера* [6, т. 3] определяется отношением миноров $[A, b]$, размеры которых оцениваются неравенствами типа (5.9). В вырожденном случае решение $Ax = b$ существует лишь в том случае, когда ранг A совпадает с рангом $[A, b]$, — и тогда в $Ax = b$ можно выбрать невырожденную подматрицу[8] A' так, что одно из решений $Ax = b$ будет решением $A'x = b'$, определяемым по *правилу Крамера*. В том и другом случае размер решения оценивается сверху размером $L([A, b])$. ▶

Если размер вектора x не превосходит величины L, т. е. все x_i записываются с помощью не более L двоичных знаков, то, очевидно, $\|x\| \leqslant 2^L$, где $\|x\| = \max |x_i|$, а точность задания x не более 2^{-L}, т. е. знаменатели координат меньше 2^L.

Этот простой факт определяет главный механизм, обеспечивающий априорные оценки в задачах, связанных с линейной алгеброй[9], где размеры вычисляемых величин, как правило, зависят полиномиально от размеров входа. В этот же круг задач попадают неравенства и оптимизация.

5.3.1. Лемма. *Если система неравенств $Ax \leqslant b$ разрешима, то она имеет решение, размер которого не превосходит $L([A, b])$. Соответственно, норма этого решения ограничена сверху величиной $2^{L([A,b])}$, точность — не более $2^{-L([A,b])}$.*

◀ Для доказательства достаточно заметить, что в случае разрешимости $Ax \leqslant b$ — одно из решений можно найти как решение $A'x = b'$, где $[A', b']$ — подматрица $[A, b]$, и сослаться на аргументы, приведенные выше. ▶

5.3.2. Лемма. *Если задача* (5.1) *имеет конечное решение, то величина оптимума не превосходит $2^{L([A,b,c])}$.*

Что касается вопросов точности порядка 2^{-L}, то подоплека здесь имеется в виду следующая. Если у задачи есть решение, то у нее есть решение x_L с L знаками после запятой, т. е. в одном из узлов ε-сети, $\varepsilon = 2^{-L}$. При наличии оценки $x_L \in V$, где область V настолько мала, что не может содержать более одного узла ε-сети, задача сводится к проверке этого узла. Узел не удовлетворяет — решения нет. Именно такой фокус работает в алгоритме

[8] Размер подматрицы, разумеется, не превосходит размера матрицы.

[9] И даже в нелинейных задачах.

эллипсоидов, которые, уменьшаясь, становятся достаточно малыми, чтобы дать однозначный ответ.

5.4. Алгоритм эллипсоидов

Философский дискомфорт в связи с ЛИНЕЙНЫМ ПРОГРАММИРОВАНИЕМ устранил *Хачиян* [10], построивший полиномиальный алгоритм с помощью модификации *метода эллипсоидов* — см. далее. На вычислительную практику это никак не повлияло — симплекс-метод остался главным рабочим инструментом, — но идеологически был обеспечен крупный прорыв. И дело было даже не в решении отдельной, хотя и важной задачи, а в неком принципиальном сдвиге NP-проблематики в целом, ибо *в некотором роде* многие NP-полные задачи являются задачами ЛП. О степени лжи этого «в некотором роде» можно судить по результатам главы 7. Пока же остановимся на *методе эллипсоидов*, игнорируя технические детали, которые с общеобразовательной точки зрения малосущественны.

Конечно, кто-то мог бы добавить, что и сам метод с просветительской точки зрения малоинтересен. Особенно если учесть его низкую практическую эффективность [11]. Тем не менее *метод Хачияна* демонстрирует пример того, как можно преодолевать комбинаторные пропасти, не вникая в детали. И это заслуживает внимания.

Как уже отмечалось, при ориентации на полиномиальную разрешимость вместо ЛП можно говорить о задаче распознавания (5.3) — ЛН, которую перепишем как $Bx \leqslant a$.

Итак, ***первое.*** Если система неравенств $Bx \leqslant a$ совместна, то у нее есть решение в некотором шаре (эллипсоиде) ell_0 с центром в нуле z_0, радиус которого можно априори оценить для ограниченных коэффициентов B и a (лемма 5.3.1). ***Второе.*** Строится последовательность эллипсоидов $\{\mathrm{ell}_k\}$ (рис. 5.2) уменьшающегося в геометрической прогрессии объема, которые (все) содержат потенциально возможное решение $Bx \leqslant a$. Процесс итерационный.

[10] *Хачиян Л. Г.* Полиномиальный алгоритм в линейном программировании // Докл. АН СССР. 1979. Т. **244**. С. 1093–1096.

[11] *Алгоритм Хачияна* — хороший пример на тему «всегда ли полиномиальный алгоритм лучше переборного».

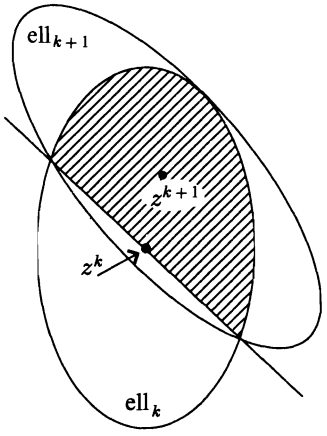


Рис. 5.2

Если ell_k уже построен, последовательно проверяются неравенства системы $Bx \leqslant a$ для $x = z^k$, где z^k — центр эллипсоида ell_k. Все удовлетворяются — задача решена. Пусть какое-то нарушено,

$$\langle b, z^k \rangle > a_i, \quad m.\,e. \quad \sum_j b_j z_j^k > a_i.$$

Это означает, что если решение есть, то оно обязано содержаться в полупространстве

$$H_k = \{x : \langle b, x \rangle \leqslant \langle b, z^k \rangle\}.$$

Поэтому в качестве ell_{k+1} берется эллипсоид минимального объема, содержащий пересечение ell_k и H_k — заштрихованную область на рис. 5.2. Соотношение объемов ell_{k+1} и ell_k оказывается меньшим $e^{-1/(2m+2)} < 1$ независимо от вектора b. **Третье.** Если объем ell_k, уменьшаясь с каждым шагом, преодолевает некий барьер $\varepsilon > 0$, вычисляемый по длине входа, а неравенство $Bz^k \leqslant a$ не удовлетворяется, задача опять-таки решена — отрицательно. Дело в том, что решение при ограниченности коэффициентов достаточно найти лишь с точностью 2^{-L} (лемма 5.3.1), оценкой которой и служит $\varepsilon > 0$.

Для понимания предмета ничего более не требуется. Переход от ell_k к ell_{k+1} можно сопроводить формулами [12]

$$D_{j+1} = \frac{m^2}{m^2-1}\left[D_j - \frac{2}{m+1}\frac{\langle D_j b, D_j b\rangle}{\langle D_j b, b\rangle}\right],$$

$$z^{j+1} = z^j - \frac{1}{m+1}\frac{D_j b}{\sqrt{\langle D_j b, b\rangle}},$$

и даже их выводом, но это не более чем упражнение по линейной алгебре, и ничего не добавляет к вышесказанному по существу.

Кое-что, правда, обрастает вынужденными оговорками. В формуле для z^{j+1} фигурирует извлечение корня, что в рациональной арифметике не всегда возможно. Однако в данном случае вычисления достаточно вести с точностью до 2^{-L}, которую можно обеспечить рациональными вычислениями с полиномиальным количеством операций. С рутинными подробностями можно ознакомиться по литературе [17, 20], но это подробности, о которых были обязаны думать первопроходцы, — теперь, когда есть гарантии, они малоинтересны.

5.5. Природа алгоритма

Игнорирование методом эллипсоидов комбинаторного духа ЛП вполне очевидно. Но «издалека» иногда думают, что результат достигается нелинейной обработкой множества вершин допустимого многогранника. Из сказанного выше ясно, что о допустимом многограннике речь вообще не идет, — и потому экспоненциальное количество вершин не более чем обстоятельство за горизонтом задачи. Вариант распознавания ЛП сводится к проверке разрешимости линейного неравенства (ЛН). И какой смысл говорить о многограннике решений ЛН, если это делу не помогает [13]. Проблему удается решить совсем иначе. Но тогда возникает вопрос, а являются ли задачи ЛП и ЛН комбинаторными?

Есть ли вообще комбинаторные задачи? Сами по себе комбинаторные, а не по трактовке. Если какую-либо NP-полную задачу

[12] Эллипсоид с центром в z задается неравенством

$$\mathrm{ell} = \{x : \langle D^{-1}(x-z), (x-z)\rangle \leqslant 1\},$$

где D — положительно определенная матрица.

[13] С тем же успехом о вещественной разрешимости квадратного уравнения можно было бы говорить как о возможности выбора подходящей пары чисел из безграничного множества.

удастся решить обходным полиномиальным маневром, идеология обозрения всевозможных вариантов сильно пострадает.

5.6. Методы внутренней точки

В «пробоину Хачияна» следующим проник *Кармаркар* [14] с *алгоритмом внутренней точки*, каковой, по замыслу, предназначался для приближения полиномиальных инструментов ЛП к потребностям практики. Обещанное превосходство над симплекс-методом снова достигнуто не было (на задачах «общего положения» не слишком больших размерностей), но далее хлынул поток работ — в количестве $\sim 10^3$, — и задача была дожата. Теперь считается, что *алгоритмы внутренней точки* — все модификации обычно называют *алгоритмами Кармаркара* [15] — наиболее предпочтительны при больших размерностях.

Алгоритмы внутренней точки в отличие от симплекс-метода, прыгающего по вершинам допустимого многогранника X, связаны с движением изображающей точки в относительной внутренности X. Все они базируются на итерационных процедурах типа

$$x_{k+1} = x_k + \Delta x_k, \tag{5.10}$$

где направление и норма корректировки Δx_k выбираются так, чтобы оставаться в X, но двигаться при этом достаточно быстро. Вариации решения этих задач, собственно, и определяют многообразие модификаций алгоритма. Сходимость (5.10) со скоростью геометрической прогрессии обеспечивает попадание x_k в достаточно малую окрестность оптимальной вершины X за полиномиальное

[14] *Karmarkar N.* A new polynomial-time algorithm for linear programming // Combinatorica. 1984. № 4. P. 373–395.

[15] Алгоритмы внутренней точки возникли, как это всегда бывает, «до первооткрывателя» (*Дикин*, *Евтушенко*, *Зоркальцев*). Заслуга *Кармаркара* состоит в том, что он вскрыл их полиномиальную природу. Такова история, собственно, и с *алгоритмом Хачияна*, внесшим в метод эллипсоидов не такие уж существенные изменения. А вот вся задача целиком была переосмыслена и поставлена с головы на ноги, что обеспечило фундаментальный прорыв. Поэтому первооткрыватели — это революционеры, «стоявшие на плечах гигантов», а первопроходцы — это всегда Адам с Евой.

число шагов. А этого «по причинам ограниченной точности» (раздел 5.3) хватает для точного решения задачи.

Серьезная проблема в связи с (5.10) возникает вроде бы с выбором начального приближения, которое обязано принадлежать X, а это равносильно решению задачи ЛН, полиномиально эквивалентной ЛП. Но в данном случае порочный круг легко разрывается. Если мы умеем полиномиально решать ЛП с помощью (5.10) при «начальном приближении $\in X$», то систему линейных неравенств $Ax \leqslant b$ можно легко решить как задачу ЛП:

$$\sum_j z_j \to \min, \quad Ax + z \geqslant b, \tag{5.11}$$

в которой при любом $x \leqslant 0$ начальное приближение легко загнать в допустимый многогранник выбором достаточно большого $z > 0$. Решение неравенства $Ax \leqslant b$ существует и будет найдено в томм случае, когда решением (5.11) окажется некоторое $z \leqslant 0$.

Практически можно действовать более рационально, не расходуя в избытке фиктивные переменные z_j, — достаточно всего одной. Вместо (5.11) удобнее решать задачу

$$\varepsilon \to \min, \quad Ax + \varepsilon r_0 \leqslant b, \tag{5.12}$$

где $r_0 = b - Ax_0$ при любом x_0. Начальное приближение $\{x_0, \varepsilon = 1\}$ лежит в допустимом многограннике (5.12). Решение неравенства $Ax \leqslant b$ существует и будет найдено в томм случае, если решением (5.12) окажется $\varepsilon = 0$.

5.7. Феномен целочисленных вершин

Если вершины допустимого многогранника M имеют целочисленные координаты, то решение ЛП является одновременно решением ЦЛП [16]. Надежды на целочисленность вершин в общем случае, конечно, никакой. Но вычленение комбинаторных задач, как и создание Вселенной, — частный случай.

[16] Ибо целочисленные точки внутри M хуже из-за линейности критерия.

Незаурядный пример (насчет вершин многогранника) — *транспортная задача* (см. раздел 1.5):

$$\sum_{i,j} c_{ij}x_{ij} \to \min, \quad \sum_{j} x_{ij} \leqslant a_i, \quad \sum_{i} x_{ij} \geqslant b_j.$$

Переменные $\{x_{ij} \geqslant 0\}$ здесь могут быть выстроены в цепочку, заново перенумерованы, — и задача приобретет стандартный вид ЛП. Правда, с самого начала проще рассматривать эквивалентную задачу [17)]

$$\sum_{i,j} c_{ij}x_{ij} \to \min, \quad \sum_{j} x_{ij} = a_i, \quad \sum_{i} x_{ij} = b_j. \tag{5.13}$$

Заменить неравенства равенствами можно, вводя фиктивных потребителей и поставщиков, числа которых допустимо считать равными, объединяя всех в один список и полагая $b_j = 0$ для чистых производителей и $a_j = 0$ для чистых потребителей.

Сложная в общей ситуации (5.1) проблема существования допустимого решения — в данном случае тривиальна. *Допустимое решение существует в томм случае, когда*

$$\sum_{j} b_j = \sum_{i} a_i. \tag{5.14}$$

◀ Необходимость (5.14) очевидна. Достаточность вытекает из существования *допустимого решения* $x_{ij} = \dfrac{a_i b_j}{n}$, которое, как легко проверяется, удовлетворяет ограничениям (5.13) при условии (5.14). ▶

Интуитивно совсем неясно, почему в случае целочисленных величин a_j, b_j оптимум (5.13) не может быть связан с перевозкой дробных количеств x_{ij} продукта. Тем не менее целочисленность a_j, b_j влечет за собой целочисленность перевозок. Поэтому, если даже (5.13) рассматривать как ЦЛП, то ее допустимо решать как непрерывную задачу ЛП — решение «такое, как нужно» получится автоматически, что следует из широко известных в линейном программировании результатов.

[17)] Задача $\langle a, u\rangle + \langle b, v\rangle \to \max$, $u_i + v_j \leqslant c_{ij}$ двойственна к (5.13). (?)

5.7.1. Теорема [18]. *Многогранник*

$$\{x : Ax \leqslant b, x \geqslant 0\}$$

при любом целочисленном векторе b *имеет целочисленные вершины в том случае, когда все миноры матрицы* A *равны либо нулю, либо* ± 1.

Легко сообразить, что теорема 5.7.1 охватывает также ситуацию с многогранником $\{x : Ax = b, x \geqslant 0\}$, и хотя она включает в себя не ахти сколько задач, но в комбинаторике иногда применяется. В частности, уникальность транспортной задачи как раз следует из п. 5.7.1. То же самое можно сказать по поводу *задачи о назначении*, но это просто вариант транспортной задачи при $a_i = 1$, $b_j = 1$.

[18] См. статью *Гофмана* и *Краскала* в сб. [14].

Глава 6

Арифметика и криптография

А Вселенская боль — твоих мук только часть,
Ты не знаешь пред кем на колени упасть,
Кто молитву твою, несмотря на цейтнот,
В небеса, как свинец, для балласта возьмет.

Арифметика в пересечении с алгоритмами искрит в основном там, где соприкасаются простые и составные числа. Но проблематика больше относится к теории чисел, нежели к комбинаторике, — что важно подчеркнуть. Иначе ПРОСТОЕ ЧИСЛО оказывается ложкой дегтя для теории алгоритмов, вклинивая в естественную комбинаторную идеологию меняющие картину исключения, — и аудитория перестает понимать, что стоит во главе угла. Например, *удостоверением* для ПЧ оказывается доказательство полиномиальной длины, что нехарактерно в рассматриваемой области, но вынуждает перекраивать общие определения и схемы, чтобы арифметике было уютно. Гостеприимства, конечно, не жалко, но беда в том, что мысль уводится из фарватера.

6.1. О причинах исключительности

Особый статус теоретико-числовых задач возникает на почве большого количества внутренних закономерностей, которые мешают свободному размножению вариантов и придают арифметической комбинаторике не вполне естественную окраску. Хотя, — если вдуматься, — с чего это вдруг совсем простая задача так туго завязана. Чтобы n было составным, надо иметь возможность разложить n шариков по нескольким ящикам так, чтобы в каждый ящик попало одинаковое число шариков. Совсем тривиальная на вид задача. Вполне комбинаторная. Но все же другая по типу, — в отличие, скажем, от ЗК, где речь идет о том, удовлетворяет ли данная конструкция некоему свойству. Здесь же стоит вопрос о существовании конструкции определенного типа в заданном классе. Различие

частично терминологически-понятийного характера, но в основе своей это задача все же другого уровня, что отчетливо видно при переходе от обычных задач на графах к задачам рамсеевского плана (раздел 4.11).

Что касается нестандартного *удостоверения* в ПЧ, то это все же из-за традиционных акцентов, потому что в дополнении МОД тоже нет *удостоверения* как «того, что ищется», а есть алгоритм проверки полиномиальной длины, который при желании можно называть доказательством. Таковы же алгоритмы для задачи ПЧ, а разница заключена в их более солидной теоремной обеспеченности, что и создает иллюзию доказательства, а не проверки. На самом же деле чисто доказательная часть в обычном понимании слова — всегда конечна, остальное — алгоритмическая проверка теоремных предположений и деклараций.

6.2. Тесты на простоту

Реальная проверка чисел на простоту опирается на идеи, заслуживающие в рамках курса по теории сложности алгоритмов определенного внимания, ибо всякая аномалия дает ощущение границ — и тем интересна.

Разложение натурального N на множители «тупым» перебором требует деления N на все простые числа, меньшие корня из N, количество которых, как известно, асимптотически пропорционально $\frac{\sqrt{N}}{\ln \sqrt{N}}$. Поэтому для 100-значного N, а в криптосистемах используются значительно бóльшие числа, необходимо проверить более 10^{40} делителей, что абсолютно нереально. Источником эффективных тестов служит

6.2.1. Малая теорема Ферма. *Для простого N и любого x, взаимно простого с N,*

$$x^{N-1} \equiv 1 \pmod{N}. \tag{6.1}$$

Проверку (6.1) для некоторого x называют *тестом Ферма*. Тождество (6.1) не является необходимым и достаточным условием простоты N. Поэтому *тест Ферма* работает с гарантией только

в одном направлении: если (6.1) нарушается, то N — составное, т. е. СЧ имеет ответ «да». В противном случае N может быть простым с некоторой вероятностью, каковая возрастает при повторении теста с другим основанием x. Но существуют составные *числа Кармайкла*, которые проходят *тест Ферма* для всех x, не являющихся их делителями.

Другими словами, N — *число Кармайкла*, если оно составное и $x^{N-1} = 1$ для любого $x \in \mathbb{Z}_N^{\times}$, где $\mathbb{Z}_N^{\times}$ мультипликативная группа вычетов [6, т. 8]. Сам Кармайкл установил (1912), что *нечетное N является числом Кармайкла в том случае, когда*

$$N = p_1 p_2 \dots p_r, \quad r \geqslant 3,$$

где p_j различные простые числа, и $N - 1$ делится на $p_j - 1$ при всех j. Гораздо позже было доказано, что *чисел Кармайкла бесконечно много*, — поэтому тест Ферма, как тест на простоту, имеет фундаментальный изъян. Изъян не такой уж маленький. Например, чисел Кармайкла, меньших $N = 10^{17}$, — более полумиллиона.

Тест Ферма можно улучшить, заметив, что для простого числа $N \neq 2$ — квадратный корень из (6.1) дает

$$x^{(N-1)/2} = \pm 1 \pmod{N}, \tag{6.2}$$

что приводит к *тесту Леманна*: если для какого-либо $x < N$ (6.2) не выполняется, то N — составное. Если выполняется, то N, — возможно, простое, причем вероятность ошибки не превосходит 0,5. Если $(N - 1)/2$ опять нечетно, то операцию можно повторить, и так далее. На этом пути возникает

6.2.2. Тест Рабина—Миллера, *состоящий в проверке выполнения одного из двух условий*:

$$x^m = 1 \pmod{N} \quad \textit{либо} \quad \exists\, r < k : \; x^{m2^r} = -1 \pmod{N}, \tag{6.3}$$

где N нечетно и $N - 1 = m \cdot 2^k$, где m нечетно.

Как и в случае *теста Ферма*, критерий (6.3) работает с гарантией только в одном направлении. Число N, не проходящее тест, — составное. В противном случае N может быть простым с некоторой вероятностью. Но что важно, для теста 6.2.2 нет аналогов чисел Кармайкла. Более того, доказано, что вероятность ошибки (при ответе «N — простое») здесь не превышает 0,25. Таким образом, при

успешном повторении теста t раз вероятность ошибки равна 4^{-t}. Фактически получается полиномиальный алгоритм, «гарантирующий» простоту N со сколь угодно малой вероятностью ошибки.

Затем *Миллер* усилил результат.

6.2.3. Теорема Миллера. *Если верна расширенная гипотеза Римана*[1] *и N проходит тест* 6.2.2 *при любом*

$$x\colon\ 1 < x < 2\log^2 N,$$

то N — простое.

Это уже полиномиальный тест на простоту с нулевой вероятностью ошибки, но червоточинка осталась в другом обличье — в виде предположения о справедливости *расширенной гипотезы Римана*, что, как говорится, вилами по воде писано, хотя и правдоподобно. Задача, как видно, так или иначе ускользала, однако создавалось впечатление, что решение где-то близко. Драматургический накал был довольно велик[2], но уверенность в успехе постепенно исчезала. На этом фоне полной неожиданностью оказалось открытие полиномиального алгоритма AKS.

6.3. Полиномиальный тест AKS

Полиномиальный *алгоритм AKS*[3] проверки числа на простоту увенчал трудоемкий поиск, и явился крупным успехом в преодолении философского дискомфорта. Как и *алгоритм Хачияна* в задаче ЛП, — AKS не повлиял на вычислительную практику, зато снял напряжение по поводу «ПЧ $\in$ P?». Центральная идея довольно проста и опирается на следующий факт.

---

[1] Обычная *гипотеза Римана* [6, т. 9] состоит в предположении, что все нетривиальные нули *дзета-функции*, аналитически продолжающей ряд $\sum_{n=0}^{\infty} \frac{1}{n^s}$, расположены на вертикальной прямой $\sigma = 1/2$. «Расширенная» — предполагает то же самое (насчет нулей) у любого *ряда Дирихле* $\sum_{n=1}^{\infty} \frac{\chi(s)}{n^s}$ с произвольной функцией $\chi(s)$, являющейся *характером*.

[2] Мы многое оставляем за кадром.

[3] Аббревиатура AKS — по первым буквам имен авторов алгоритма [21].

6.3.1. *Натуральное n, при условии НОД $(a, n) = 1$, является простым в томм случае, когда*

$$(x-a)^n \equiv (x^n - a) \pmod{n}. \tag{6.4}$$

◀ В разложении

$$(x-a)^n = \sum_{k=0}^{n} C_n^k x^k (-a)^{n-k}$$

могут не делиться на простое n только крайние слагаемые, причем по *малой теореме Ферма*

$$a^n \equiv a \pmod{n},$$

что и дает (6.4). Если же n составное, $n = p^s q$ и q не делится на p, то C_n^p не делится на n — и (6.4) нарушается. ▶

Таким образом, проверка n на простоту может быть заменена проверкой тождества (6.4). При бесхитростном подходе это едва ли может дать выигрыш, но определенные уловки позволяют свести «экспоненциальные усилия» к «полиномиальным». В частности, необходимый перебор значительно сокращается после деления (6.4) на некоторый полином вида $x^r - 1$ и рассмотрения остатков, — что позволяет заменить (6.4) менее тяжеловесным [4)]

$$(x-a)^n \equiv (x^n - a) \pmod{n,\ x^r - 1}. \tag{6.5}$$

Главный результат [21], в несколько модифицированном виде, состоит в установлении следующего факта.

6.3.2. *Пусть натуральное n и простое r таковы, что*

(i) *порядок n в группе $\mathbb{Z}_r$ больше $(\log_2 n)^2$;*

(ii) *n не делится на простые числа, меньшие r;*

(iii) *тождество (6.5) выполняется для всех $a \in [1, \sqrt{r}\log_2 n]$.*

Тогда n — степень простого числа.

6.3.3. *На базе* 6.3.2 *алгоритм AKS работает следующим образом.*

1) *делимость n на числа от 2 до $\lfloor(\log_2 n)^5\rfloor$ проверяется в лоб;*

2) *далее ищется $r \leqslant \lfloor(\log_2 n)^5\rfloor$, для которого выполняется* **(i)** *в* 6.3.2 [5)];

[4)] Равенство (6.5) означает существование такого полинома $q(x) \in \mathbb{Z}[x]$, что все коэффициенты полинома $(x-a)^n - (x^n - a) - q(x)(x^r - 1)$ кратны n.

[5)] Такое r гарантированно существует — техническая лемма.

3) *проверяется* **(iii)**;
4) *проверяется, не извлекается ли из* n *целый корень.*

Алгоритм полиномиален, но вычислительной ценности не представляет, ибо временна́я трудоемкость довольно высока — порядка $\log^{7,5} n$, а первоначальные оценки использовали двузначные показатели логарифма. Пока, правда, рецепт продолжает обрастать усовершенствованиями, но опередить вероятностные алгоритмы в чисто прагматическом отношении — едва ли удастся.

Интересно отметить, что обоснование результатов 6.3.2, 6.3.3 не совсем тривиально, но не так уж сложно. Это несколько странно на фоне того обстоятельства, что задача долгое время была в фокусе внимания большого числа желающих добиться успеха. Аналогичная ситуация имела место в связи с *алгоритмом Хачияна*. Выводы напрашиваются психологического толка. Исследовательский порыв затягивается не в ту воронку, а длительные безрезультатные усилия создают ореол неразрешимости у предмета чаяний, что подрывает веру.

6.4. Алгоритмы факторизации

Рассмотренные выше алгоритмы решения задач распознавания ПЧ, СЧ обходят стороной поиск разложения N на множители. Полного перебора здесь, конечно, можно избежать, но полиномиальные рецепты до сих пор не известны. Ситуация несколько странная, потому что в стандартных комбинаторных задачах из класса P (типа МОД) вариант, на котором достигается ответ «да» или «нет», разыскивается полиномиально. Здесь вопрос повис нерешенным, но мнение большинства склоняется в пользу «полиномиальной неразрешимости», хотя эффективность имеющихся алгоритмов не так уж плоха — субэкспоненциальна.

В основном такие алгоритмы опираются на идею представления составного N разностью двух квадратов:

$$N = a \cdot b = \left(\frac{a+b}{2}\right)^2 - \left(\frac{a-b}{2}\right)^2 = x^2 - y^2. \qquad (6.6)$$

В случае простого N разложение $N = N \cdot 1$ единственно, $a = N$, $b = 1$. Для составных N поиск решений (6.6) может опираться на естественную последовательность действий. Ищется наименьший квадрат x^2, превосходящий $N \Leftrightarrow x = \lceil \sqrt{N} \rceil$. Далее из найденного x^2 вычитается N. Если из $y = x^2 - N$ извлекается целый корень z, то

$$N = x^2 - z^2 = (x + z)(x - z),$$

и в случае $x - z \neq 1$ — задача решена. В противном случае берется следующий квадрат после x^2 и т. д.

Это, конечно, голая схема, на которую могут накладываться различные уловки и трюки. Но последние даже в рекордных вариантах не поднимаются до полиномиальной эффективности. Прорыв ожидается в области квантовых вычислений — глава 10.

6.5. Система RSA

Система шифрования RSA[6)] на сегодняшний день является крупным потребителем теорем и алгоритмов из области ПЧ, СЧ и *факторизации составных чисел*. Идея, лежащая в основе RSA, проста до гениальности. Цифровой текст x шифруется с помощью легко вычисляемой функции

$$f(x) = x^e \pmod{N}, \tag{6.7}$$

где числа e и N, составляющие *открытый ключ*, общедоступны. При определенных требованиях к $\{e, N\}$, о которых будет сказано далее, функция (6.7) обратима, но f^{-1} вычисляется уже не просто. Точнее говоря, значения f^{-1} легко определяются, если известен секрет, без которого расшифровка практически невозможна. В сказанное с трудом верится, поскольку для расшифровки сообщения $y = f(x)$ надо всего лишь решить уравнение

$$x^e = y \pmod{N}, \tag{6.8}$$

но тут коса на камень как раз и находит.

[6)] По первым буквам имен авторов: *Rivest R. L.*, *Shamir A.*, *Adleman L. M.* A method for obtaining digital signatures and public-key cryptosystems // Communications ACM. 1978. **21**, № 2. 120–126.

В криптографическом механизме здесь существенную роль играет *функция Эйлера* $\varphi(N)$, равная количеству натуральных $n < N$, взаимно простых с N. Значение $\varphi(N)$ легко вычисляется, если известно разложение N на простые множители, ибо $\varphi(p^k) = p^{k-1}(p-1)$ для простых p, и $\varphi(pq) = \varphi(p)\varphi(q)$ для взаимно простых p и q. Поэтому

$$\varphi(N) = N \prod_{p|N} \left(1 - \frac{1}{p}\right),$$

где $p|N$ обозначает простые делители числа N.

По *теореме Эйлера*[7]

$$x^{\varphi(N)} \equiv 1 \pmod{N} \tag{6.9}$$

для любого натурального x, взаимно простого с N. ◀ Для доказательства (6.9) достаточно заметить, что порядок элемента x делит порядок мультипликативной группы вычетов $\mathbb{Z}_N^\times$ [6, т. 8]. Поэтому $x^{\varphi(N)} = 1$ в $\mathbb{Z}_N^\times$. ▶

• Безошибочный тест на простоту N дает определение периода функции $f(t) = x^t \pmod{N}$, где x взаимно просто с N. Если период $r = N-1$, где $f(t+r) \equiv f(t)$, то N — гарантированно простое число, если $r < N-1$, то N — гарантированно составное. (?)

В RSA обычно полагается $N = pq$, где p и q — различные простые числа (как правило, примерно одинаковые по величине), а показатель e выбирается взаимно простым с $\varphi(N)$. *В этом случае решение* (6.8) *единственно и определяется формулой*

$$x \equiv y^d \pmod{N}, \tag{6.10}$$

где натуральное $d < \varphi(N)$ *существует и однозначно извлекается из условия*

$$de = 1 \pmod{\varphi(N)}. \tag{6.11}$$

◀ Решение (6.11) существует в силу НОД $(e, \varphi(N)) = 1$, единственность d вытекает из групповых свойств вычетов [6, т. 8].

В случае $N = pq$ имеем НОД $(y, N) \in \{1, p, q\}$. Если НОД $(y, N) = 1$, то в силу (6.9) и (6.11)

$$y^d \equiv x^{de} \equiv x \pmod{N},$$

что приводит к (6.10). Если НОД $(y, N) = p$, то $\varphi(N)$ делится на $\varphi(q)$, а из (6.8) следует НОД $(x, q) = 1$. Далее, аналогично предыдущему, получаем

$$x \equiv y^d \pmod{q} \quad \Rightarrow \quad (6.10). \quad ▶$$

[7] Прямым следствием является *малая теорема Ферма*.

Вот, собственно, и вся подноготная. Впечатление, конечно, странное. Секрет d выглядит секретом полишинеля. Ибо что мешает разложить N на простые множители, после чего вычислить $\varphi(N)$ и решить (6.11), разоблачая d? Иными словами, открытый ключ $\{e, N\}$ содержит полную информацию о секрете, и это как-то не похоже на тайнопись. Но фокус в том и заключается. Первый же шаг, вычисляющий d, сталкивается с необходимостью разложения N на простые множители, что при современных технологиях для больших N оказывается непосильной задачей.

Авторы RSA для демонстрации силы метода зашифровали некую фразу[8] с помощью открытого ключа со 129-значным N и $e = 9007$, да еще с подсказкой, что p и q в $N = pq$ 64- и 65-значны, — и объявили премию в 100$ за расшифровку. Премию пришлось все-таки выплатить. Работа по расшифровке заняла более полугода[9], в нее было вовлечено несколько сотен человек и полторы тысячи компьютеров при сетевом взаимодействии. Проделанный объем вычислений превосходил 10^{15} операций. Для 200-значного N необходимый объем вычислений имеет порядок 10^{23}, и задача выходит за рамки возможностей современных технологий.

Реализация метода сопряжена, безусловно, с преодолением определенных трудностей, начиная с точного перемножения больших чисел, для чего стандартное программное обеспечение не годится. Построение многоразрядных простых p и q — тоже проблема. Но все это, так или иначе, решается и подробно описывается в литературе. Практические рычаги — в руках компании RSA Data Security[10]. Криптографические модули RSA используются в MS Windows и сотнях других программных продуктов.

---

[8] Переведенную нумерацией букв в цифровой код.

[9] А на приготовление ушло 17 лет.

[10] Создана в 1982 г., президент — первый (по алфавиту) автор метода. О размахе дела свидетельствует чистый годовой доход более миллиарда долларов США. Подробная информация на сайте www.rsa.com.

6.6. Вариант распознавания

Толкование декодирования RSA-сообщений в терминах распознавания порождает не вполне стандартные NP-задачи.

ШИФР

По RSA-шифру y и открытому ключу $\{e, N\}$ определить, удовлетворяет ли сообщение x, т. е. решение уравнения

$$y = x^e \pmod N, \tag{6.12}$$

некоторому полиномиально проверяемому условию[11] $A(x)$?

Задача ШИФР $\in$ NP в силу полиномиальной проверяемости *удостоверения* x. Возникает также впечатление «ШИФР $\in$ P», поскольку существует полиномиальный алгоритм (6.10) вычисления

$$x = y^d \pmod N,$$

а полиномиальность $A(x)$ оговорена. С другой стороны, если предположить, что уравнение (6.12) действительно труднорешаемо, то *секрет* d не вычисляется в рамках P — и возникает парадокс, разумеется, в предположении NP $\neq$ P.

> *Вопрос в данном случае не в том, труднорешаема ли задача* (6.12) *на самом деле. Пусть труднорешаема. В этом достаточно правдоподобном случае возникает задача распознавания, которая лежит в P, но полиномиально не решается.*

Парадокс, конечно, с червоточинкой, потому что любой критик потребует *секрет* d воспринимать как часть *удостоверения*. С этим, правда, можно вяло спорить, поскольку d годится одновременно для целого семейства задач, характеризуемого открытым ключом $\{e, N\}$. Но тогда выходит — каждую подзадачу решает свой алгоритм, что опять-таки вне регламента[12]. И хотя один и тот же

[11] Подразумевается условие $A(x)$, которое вынуждает искать x.

[12] До того как было установлено ПЧ $\in$ P, неясно было, принадлежит ли ПЧ классу NP. Проблема была решена (см. раздел 1.8) указанием своего полиномиального доказательства

алгоритм (6.10) решает сразу большую группу задач, уйти полностью из-под критики не удается.

Тем не менее в сухом остатке кое-что остается, насчет уязвимости понятий P- и NP-классов. На абстрактном уровне пошуметь можно было бы и раньше, но тут правдоподобная ситуация возникает на более-менее конкретном примере, демонстрируя «как бы это могло быть». Определения 1.2.2 и 1.2.3 весьма размыты, и понимать их можно по-разному, как собственно любые другие определения и теоремы [13]. Кое-что из неприятностей уже отмечалось в разделе 3.1 — деление P-задач на те, где полиномиальность доказуема, и те, где полиномиальность принципиально невозможно установить (теорема 3.1.1). В данном случае добавляется возможная «полиномиальная неконструктивность» существования P-алгоритма. Не о ШИФРЕ речь, ибо задача находится в «гипотетически подвешенном состоянии» из-за неясного статуса проблемы факторизации чисел, плюс другие сомнительные моменты. Но теперь все же нагляднее, какого сорта мина заложена под определениями 1.2.2 и 1.2.3.

6.7. Дискретный логарифм

Криптографические системы с открытым ключом используют *функции с секретом*: $f(x)$ считается быстро, а время вычисления $f^{-1}(x)$ зашкаливает, если не известен *секрет*, с помощью которого значения $f^{-1}(x)$ также быстро вычисляются. Существование таких функций (при естественном уточнении терминологии) не доказано. Но практически — они есть, как показывает система RSA, где используется $f(x) = x^e \pmod N$. Сложность раскрытия секрета определяется необходимостью решения той или иной труднорешаемой задачи. В RSA — это разложение составного числа на простые множители.

(алгоритма) для каждого простого числа. Но это вовсе не означало ПЧ $\in$ P. Единый алгоритм AKS был получен позже.

[13] Недаром говорят, что только доказательство позволяет уяснить, о чем теорема. Но и этого, вообще говоря, мало. Чтобы хорошо понять какой-либо факт, надо посмотреть, как он применяется и как работает в разных ситуациях. Однако до конца ясно ничто не бывает.

Еще одна популярная гипотетически труднорешаемая задача — вычисление *дискретного логарифма*[14]. Речь идет об определении целого x в равенстве

$$a^x = b \pmod N, \tag{6.13}$$

где N — простое число большее двух, a — порождающий элемент[15] мультипликативной группы вычетов $\mathbb{Z}_N^\times$ и целое $b < N$. При этом пишут $x = \log_a b$, держа подробности в уме.

Дискретный алгоритм естественно возникает в *задаче формирования секретного ключа абонентами A и B, взаимодействующими по открытому каналу связи*. В соответствии с *алгоритмом Диффи—Хелмана* A и B независимо друг от друга выбирают числа X_A и X_B, и, держа их при себе, с помощью общедоступных a и N вычисляют каждый свое:

$$Y_A = a^{X_A} \pmod N, \quad Y_B = a^{X_B} \pmod N,$$

после чего в открытую обмениваются числами Y_A, Y_B и вычисляют, опять-таки каждый свое число, $Y_B^{X_A}$, $Y_A^{X_B}$, каковые оказываются равны по модулю N,

$$Y_B^{X_A} = Y_A^{X_B} = a^{X_A X_B} \pmod N,$$

в результате у каждого появляется *секретный ключ*

$$d = a^{X_A X_B} \pmod N,$$

доступный «посторонним» лишь при умении вычислять дискретный логарифм, т. е. определять X_A, X_B по значениям Y_A, Y_B.

По поводу различных задач криптографии необходимо заметить, что каждый раз предлагаемые методики имеют различные «слепые пятна», каковые могут успешно использоваться третьей стороной. В данном случае, например, отдельного рассмотрения

---

[14] Если $N - 1$ в (6.13) разлагается в известное произведение малых простых чисел, то имеется быстрый алгоритм вычисления дискретного логарифма.

[15] Говорят также: *образующий* элемент $\mathbb{Z}_N^\times$. Такие числа a называются еще *первообразными корнями*, и их количество равно $\varphi(N-1)$, где φ — *функция Эйлера*.

требует проблема аутентификации. Иначе пользователь не может быть уверен, что обменивается информацией со своим визави, и угроза имитации сильно действует на нервы, не говоря о возможных рисках. Другой пример — обход лицензионного ключа в пиратских программах, скажем, за счет увеличения пробных 30 дней до 30 лет.

6.8. Системы с нулевым разглашением

Система с нулевым разглашением основана на игре двух лиц, в которой *Оракул* $\mathcal{O}$ знает решение некой данной задачи, и в процессе диалога пытается доказать, возможно блефуя, «что знает», — другому игроку, *Вычислителю* $\mathcal{V}$. Последний в проверках ограничен «полиномиальными средствами». Иллюстрация на ИЗОМОРФИЗМЕ ГРАФОВ рассматривалась в разделе 4.9, что имеет смысл здесь напомнить.

Пусть *Оракул* знает изоморфизм (перестановку вершин) φ графов G_0 и G_1, и по протоколу посылает *Вычислителю* $\mathcal{V}$ граф $H = \psi(G_0)$, либо $H = \psi(G_1)$, где ψ — другая перестановка, не равная φ. $\mathcal{V}$ запрашивает изоморфизм либо $H \sim G_0$, либо $H \sim G_1$ — равновероятно. В первом случае $\mathcal{O}$ посылает ψ, во втором — $\varphi \cdot \psi^{-1}$. Игра повторяется N раз.

Если графы действительно изоморфны, $G_0 \sim G_1$, все проверки дадут положительный результат. Если φ — блеф, с вероятностью 2^{-N} хотя бы один раз *Оракул* вынужден будет соврать — в том случае, когда $\mathcal{V}$ попросит $H \sim G_1$, тогда как $H = \psi(G_0)$. В итоге *Вычислитель* вскроет обман [16].

Уникальность ситуации в том, что $\mathcal{O}$ убеждает $\mathcal{V}$ в знании φ, не предъявляя самого φ. И если изоморфизм взять в качестве пароля, ключа, — появляются невиданные криптографические возможности. Пароль не посылается, а обладание им подтверждается *пустым* для подслушивающей стороны диалогом.

Будь NP-полнота ИЗОМОРФИЗМА ГРАФОВ доказана, в качестве пароля можно было бы брать решение любой другой NP-задачи, используя полиномиальную сводимость задач. Поскольку

[16] Вероятность ошибки 2^{-N} может быть сделана сколь угодно малой выбором N независимо от размерности задачи.

ситуация под вопросом, *протокол с нулевым разглашением* стандартно организуют для достоверно NP-полной задачи РАСКРАСКА ГРАФА В 3 ЦВЕТА. Делается это совсем просто.

Оракул раскрашивает вершины графа, затем случайным образом переставляет три цвета между собой, и «закрывает» вершины. *Вычислитель* случайно выбирает ребро и просит открыть две его вершины. Цвета разные — ответ принимается. Игра повторяется $N \cdot n^2$ раз [17].

Если правильная раскраска существует, $\mathcal{O}$-тест проходит. Если — не существует, *Оракулу* приходится каждый раз какие-то две вершины закрашивать одним цветом, что обнаруживается *Вычислителем* за n^2 попыток с вероятностью не менее $(1 - n^{-2})^{n^2} \geqslant e^{-1}$. При $N \cdot n^2$ попытках (партиях) вероятность прохождения теста не более e^{-N}.

Вероятность ошибки, таким образом, опять-таки может быть сделана сколь угодно малой, но теперь необходимое число партий зависит от размерности задачи — растет пропорционально n^2.

Кстати, *Оракул* здесь не обязан быть семи пядей во лбу. На его месте справится любой, кто знает пароль — «изоморфизм» или «раскраску». Первая задача, правда, выглядит предпочтительнее, поскольку ее «протокол» естественно ложится в принятые нормы общения через Интернет. Что касается второй задачи, то там привязка к реальности сопряжена с преодолением технических трудностей по воплощению схемы в интернетовские стереотипы [18].

6.9. Другие задачи

Не все, конечно, сосредоточено на криптографии. Теория чисел является источником большого количества своеобразных задач, проливающих дополнительный свет на P/NP-проблематику. Свет, скорее, манящий, нежели озаряющий, — но он подталкивает мысль

[17] Случайная перестановка цветов в каждой партии нужна для поддержания *нулевого разглашения*.

[18] Недаром стандартное изложение описанной выше схемы сопровождается обычно интерпретацией, в которой окрашенные вершины графа помещаются в закрытые ящики и посылаются *Вычислителю*, а тот может два ящика открыть. Электронная модель такой процедуры элементарна, но вопрос в том, как не потерять при этом «нулевое разглашение».

в нестандартных направлениях. Говорить подробно об арифметических задачах здесь неуместно [19], однако кое-что заслуживает упоминания, хотя бы для примера.

Арифметика богата не только сложными задачами, но и простыми, — что позволяет класс P рассматривать с близкого расстояния. Классический пример — *алгоритм Евклида* поиска наибольшего общего делителя натуральных a и b — НОД(a, b).

Механизм в предположении $a > b$ работает так. Сначала a делится на b: $a = bq_1 + r_2$, — после чего b делится на r_2: $b = r_2q_2 + r_3$. Далее остаток r_2 делится на r_3, — и так до остановки итерационного процесса, которая неизбежна, ибо остаток на каждом шаге строго уменьшается. В целом процесс имеет вид:

$$\begin{cases} a = bq_1 + r_2, \\ b = r_2q_2 + r_3, \\ r_2 = r_3q_3 + r_4, \\ \vdots \\ r_{n-2} = r_{n-1}q_{n-1} + r_n, \\ r_{n-1} = r_nq_n, \end{cases} \tag{6.14}$$

что в результате дает $r_{n+1} = 0$. Просматривая теперь (6.14) сверху вниз, обнаруживаем: общие делители a и b совпадают с общими делителями b и r_2, которые, в свою очередь, совпадают с общими делителями r_2 и r_3 и т. д. Таким образом,

$$\text{НОД}(a, b) = \text{НОД}(b, r_2) = \text{НОД}(r_2, r_3) = \ldots = \text{НОД}(r_{n-1}, r_n) = r_n,$$

т. е. r_n оказывается искомым НОД чисел a и b.

Держа описанную схему в поле зрения, легко понять, что

$$\text{НОД}\,(ah,\ bh) = \text{НОД}\,(a, b) \cdot h,$$

а также

$$\text{НОД}\left(\frac{a}{s}, \frac{b}{s},\right) = \frac{\text{НОД}\,(a, b)}{s},$$

где s — любой общий делитель a и b. В частности,

$$\text{НОД}\left(\frac{a}{\text{НОД}\,(a, b)},\ \frac{b}{\text{НОД}\,(a, b)},\right) = 1.$$

[19] Такие задачи будут рассмотрены в 13-м томе «Теория чисел».

6.9.1. Теорема[20]. *Всегда можно указать такие числа* u, v, *что*

$$\boxed{au + bv = НОД(a, b).}$$

Иначе говоря, НОД (a, b) *линейно выражается через* a *и* b *с помощью коэффициентов* $u, v \in \mathbb{Z}$.

Легко убедиться, что объем вычислений (6.14) имеет порядок $O(\log a + \log b)$. Таким образом, *алгоритм Евклида* полиномиален по длине записи обрабатываемых чисел, и задача распознавания «НОД $(a, b) > K$?» принадлежит классу P. Такого сорта арифметических P-задач — произрастающих на почве вычислений — довольно много. Но угодить в NP-бездну здесь довольно легко при безопасном на вид дрейфе. Вычисление $z = x^n \pmod N$ обходится менее чем в $\lceil \log n \rceil$ итераций, а на *дискретный логарифм* уходят экспоненциальные ресурсы. Вот еще несколько достаточно невинных с виду задач, выходящих за пределы класса P, разумеется, в предположении $\mathrm{P} \neq \mathrm{NP}$.

• *Существует ли решение* $x \leqslant a$ *у сравнения* $x^2 = b \pmod N$? Задача NP-полна, но решается за *псевдополиномиальное* время, и становится P-задачей при известном разложении N на простые множители и снятии ограничения $x \leqslant a$. В случае простого N и справедливости *расширенной гипотезы Римана* задача становится полиномиальной.

• *Существует ли натуральное* c, *делящее больше членов последовательности* $\{a_1, \ldots, a_n\}$, *чем членов последовательности* $\{b_1, \ldots, b_m\}$?

• *Имеет ли уравнение* $ax^2 + by = c$, *где* $a, b, c \in \mathbb{N}$, *решение в целых положительных* x, y? Задача NP-полна. Общий вопрос о существовании целых корней у полинома с коэффициентами из $\mathbb{Z}$ вообще неразрешим [6, т. 6]. Для выяснения разрешимости линейного уравнения $\sum\limits_k a_k x_k = c$ достаточно полиномиального времени.

• *Существует ли целое* x, *при котором* N *делится на* $ax + 1$ *без остатка*? Казалось бы, чего проще, но задача специфически труднорешаема [7].

[20] Обоснование легко извлекается из схемы (6.14).

6.10. Технические дополнения

Теоретико-числовые трюки представляют интерес не только сами по себе, но и как инструмент, меняющий облик комбинаторных задач. Трудно поверить, например, что РЮКЗАК — задача максимизации линейной формы при *одном* (!) линейном ограничении — эквивалентна общей задаче ЦЛП. В основе здесь лежит следующий тривиальный, но тем не менее неожиданный факт.

6.10.1. *Любой системе*

$$\sum_{j=1}^{n} a_{ij}x_j = b_i, \quad i = 1, \ldots, m, \tag{6.15}$$

с коэффициентами a_{ij}, b_i *из* $\mathbb{N}$ *можно сопоставить одно уравнение*

$$\sum_{j=1}^{n} \alpha_j x_j = \beta, \tag{6.16}$$

такое что множества неотрицательных целочисленных решений (6.15) *и* (6.16) *совпадают. При этом коэффициенты* (6.16) *могут быть получены как линейные комбинации*

$$\alpha_j = \sum_{i=1}^{m} a_{ij}\tau_i, \quad \beta = b_1\tau_1 + \ldots + b_m\tau_m, \tag{6.17}$$

целых $\tau_1, \ldots, \tau_m$.

◀ В случае (6.17) множество неотрицательных решений (6.15) содержится в множестве неотрицательных решений (6.16). Это очевидно. Обратное устанавливается немного сложнее, с использованием известного факта: «Линейное уравнение вида (6.16) разрешимо в целых числах в том случае, когда НОД $(\alpha_1, \ldots, \alpha_n)$ делит β». ▶

Декларированная выше «эквивалентность», разумеется, не гарантирует ограниченность коэффициентов (6.16) при ограниченных коэффициентах (6.15). Иначе задача ЦЛП не была бы сильно NP-полной, и сводилась бы к псевдополиномиальному РЮКЗАКУ.

Кольцо вычетов $\mathbb{Z}_n$ — весьма популярный и эффективный теоретико-числовой инструмент — представляет собой множество

$$\mathbb{Z}_n = \{0, 1, \ldots, n-1\}, \tag{6.18}$$

на котором заданы две операции: сложение и умножение по модулю n,

$$a + b \pmod{n}, \quad a \cdot b \pmod{n},$$

что вписывается в определение *кольца* — см. [6, т. 8].

Множество (6.18) с операцией сложения $a + b \pmod{n}$ образует *аддитивную группу* $\mathbb{Z}_n^+$ кольца $\mathbb{Z}_n$. Что касается совокупности (6.18) по умножению $a \cdot b \pmod{n}$, то она группой, вообще говоря, не является. Но элементы (6.18), имеющие обратные по умножению, уже образуют группу, которая называется *мультипликативной группой* $\mathbb{Z}_n^\times$ кольца $\mathbb{Z}_n$. В группу $\mathbb{Z}_n^\times$ входят те и только те элементы кольца $a \in \mathbb{Z}_n$, которые взаимно просты с n, т. е. НОД $(a, n) = 1$.

Если n простое [21], то все элементы (6.18), за исключением нуля, входят в $\mathbb{Z}_n^\times$. В общем случае это не так. Например,

$$\mathbb{Z}_4^\times = \{1, 3\}, \quad \mathbb{Z}_6^\times = \{1, 5\}, \quad \mathbb{Z}_8^\times = \{1, 3, 5, 7\}, \quad \mathbb{Z}_9^\times = \{1, 2, 4, 5, 7, 8\},$$

где $\mathbb{Z}_4^\times = \langle 3 \rangle$, $\mathbb{Z}_6^\times = \langle 5 \rangle$, $\mathbb{Z}_9^\times = \langle 2 \rangle = \langle 5 \rangle$ — *циклические группы*, $\mathbb{Z}_8^\times$ — не циклическая.

Повышенное внимание к группам $\mathbb{Z}_{p^k}^\times$ объясняется тем, что $\mathbb{Z}_n^\times$ в случае разложения $n = p_1^{k_1} \dots p_t^{k_t}$ на простые множители — есть прямое произведение групп $\mathbb{Z}_{\tau_i}^\times \left(\tau_i = p_i^{k_i} \right)$, каковые оказываются «структурными блоками». Все группы $\mathbb{Z}_{\tau_i}^\times$ циклические за исключением $\mathbb{Z}_{2^k}^\times$, $k \geqslant 3$. Порядок группы $\mathbb{Z}_n^\times$ равен значению $\varphi(n)$ *функции Эйлера* φ.

Естественный интерес представляют *порождающие элементы* (первообразные корни) группы $\mathbb{Z}_n^\times$, каковыми являются те и только те элементы $x \in \mathbb{Z}_n^\times$, где n — степень простого нечетного числа, которые удовлетворяют условию $x^{(n-1)/p} \neq 1 \pmod{n}$ для любого простого делителя p числа $n - 1$.

6.10.2. Китайская теорема об остатках. *Каковы бы ни были взаимно простые* $n_1, \dots, n_k \in \mathbb{N}$ *и целые* $x_1, \dots, x_k \in \mathbb{N}$, — *существует такое* x, *что*

$$\begin{aligned} x &= x_1 \pmod{n_1}, \\ x &= x_2 \pmod{n_2}, \\ &\dots\dots\dots\dots \\ x &= x_k \pmod{n_k}, \end{aligned} \tag{6.19}$$

т. е. существует x, *дающее при делении на* $n_1, \dots, n_k$ *остатки* $x_1, \dots, x_n$.

[21] В случае простого n кольцо $\mathbb{Z}_n$ является *полем*, — по поводу алгебраических понятий см. [6, т. 8].

При этом один из таких иксов может быть получен как $x = x^*$,

$$x^* = \sum_{i=1}^{k} \left(\frac{n}{n_i}\right)^{-1} x_i,$$

где $n = n_1 n_2 \ldots, n_k$, *а* $(n/n_i)^{-1}$ *— обратный в* $\mathbb{Z}_{n_i}^{\times}$ *к элементу* n/n_i. *Остальные* x, *удовлетворяющие* (6.19), *сравнимы с* x^* *по модулю* n.

◀ Элемент n/n_i делится на все n_j при $j \neq i$. Поэтому

$$\left(\frac{n}{n_i}\right)^{-1} x_i = \begin{cases} x_j \pmod{n_j}, & j = i, \\ 0 \pmod{n_j}, & j \neq i, \end{cases}$$

что означает $x^* = x_j \pmod{n_j}$ при любом j. Отсюда для x из (6.19) имеем $x - x^* = 0 \pmod{n_j}$ при любом j, что дает $x - x^* = 0 \pmod{n}$. ▶

Глава 7

Геометрический подход

Формальная позиция всегда очень сильна и корректна в глазах публики; но она стерильна, особенно сейчас. И, прибавлю, просто скучна.

Н. Лузин

Диалектика толкает к истине дорогой лжи. Один раз соврешь так, другой — эдак, и кое-что начинает вырисовываться.

Геометрический взгляд на любую математическую проблему подключает шестое чувство и расширяет горизонты. В данном случае это, пожалуй, единственный ракурс, в котором перспективы $\mathbf{P} \stackrel{?}{=} \mathbf{NP}$ не упираются в глухую стену. По крайней мере, интуитивно.

7.1. Общая картина

Вообще говоря, не вполне ясно, чем легко решаемые задачи отличаются от трудно решаемых. Каковы признаки, и есть ли таковые? В принципе, конечно, есть. В конце концов, даже в случае P=NP нынешние предположительно труднорешаемые задачи чем-то да отличаются от — простых.

Определенный свет на истоки здесь проливает геометрическая интерпретация комбинаторных задач. Речь идет о следующем подходе [3,4]. Практически всякая задача комбинаторной оптимизации может быть представлена как

$$c(x) \to \max, \quad x \in X,$$

где X — дискретное множество точек комбинаторной природы, а $c(x)$ — целевая функция. В линейном случае $c(x) = \langle c, x\rangle$ множество X без ущерба для задачи можно заменить выпуклой оболочкой со-X, что приводит *вроде бы* к задаче ЛП:

$$\langle c, x\rangle \to \max, \quad x \in \text{со-}X, \tag{7.1}$$

каковую для краткости обозначают иногда как $[X, c]$.

> *Принципиальное отличие* (7.1) *от стандартной задачи ЛП заключено в описании многогранника* со-X *не определяющими неравенствами, как обычно, а выпуклой оболочкой вершин. Фундаментальная роль этого обстоятельства будет неоднократно всплывать далее.*

Не будь разницы в описании задачи ЛП, ларчик бы открывался совсем просто. Масса NP-полных задач легко записывается в виде (7.1), см. далее, после чего полиномиальный *алгоритм Хачияна* или *Кармаркара* решает задачу ЛП, и проблема исчерпана. На этом пути, однако, возникает «непреодолимая» загвоздка. Приведение (7.1) к стандартному виду

$$\langle c, x\rangle \to \max, \quad Ax \leqslant b,$$

требует привлечения вычислительных ресурсов, которые могут иметь «экспоненциальную природу», после чего экономия на решении самой задачи ЛП ничего не дает.

Тем не менее тот факт, что разнообразные NP-задачи напрямую перекликаются с ЛП, — выглядит многообещающе и кое-что реально вскрывает. Можно даже сказать, что вся NP-проблематика фокусируется на взаимодействии *фасетного* и *вершинного* описания многогранников [1].

Задачи на графах приводятся к виду (7.1) примитивным образом. Пусть c_{ij} обозначает вес дуги $(i, j) \in E$ в полном графе $G(V, E)$, где E — совокупность ребер, а $V = \{1, \ldots, n\}$ — множество вершин. Занумеровав ребра в некотором порядке, от 1 до m, и полагая координаты x_k вектора $x \in \mathbb{R}^m$ равными 1, если k-е ребро входит в рассматриваемое подмножество ребер, и $x_k = 0$ в противном

[1] Имеется в виду описание многогранника с помощью неравенств (*фасетное*) и с помощью выпуклой оболочки вершин (*вершинное*).

случае, — мы получаем возможность говорить о различных *подмножествах ребер* X как о совокупностях «характеристических векторов». Например, если $X \subset \mathbb{R}^m$ обозначает множество гамильтоновых контуров[2)] графа G, то (7.1) является задачей КОММИВОЯЖЕРА.

Аналогично переводятся в форму (7.1) КЛИКА, МИНИМАЛЬНОЕ ОСТОВНОЕ ДЕРЕВО и другие задачи на графах[3)]. При этом ясно, что речь не идет о каких-то хитростях. Преобразования тривиальны и сводятся к обыкновенному переходу на другой язык. Иногда требуется минимальная изобретательность. Скажем, УПОРЯДОЧЕНИЕ ЧИСЛОВОГО МАССИВА $m = \{m_1, \ldots, m_n\}$ приводится к виду (7.1), если в качестве X взять множество векторов со всевозможными перестановками координат $\{m_1, \ldots, m_n\}$ и положить, например,

$$c = \{n, n-1, \ldots, 1\}.$$

Обратим еще раз внимание, что замена оптимизации на конечном множестве X оптимизацией на сплошном многограннике co-X возможна благодаря линейности критерия $\langle c, x\rangle \to \max$. При этом оптимум достигается в одной из вершин co-X, множество $\operatorname{ext}(\text{co-}X)$ которых обычно совпадает с X, но вообще говоря, $\operatorname{ext}(\text{co-}X) \subset X$. Для простоты будем считать, тем не менее,

$$\operatorname{ext}(\text{co-}X) = X,$$

что выполняется в подавляющем большинстве задач.

Выгодный ракурс рассмотрения задачи (7.1) дает *конусная интерпретация*, сопоставляющая точкам $x \in X$ конусы

$$K(x) = \{w : \langle w, x\rangle \geqslant \langle w, y\rangle,\ \forall\, y \in X\}.$$

В примере, изображенном на рис. 7.1, вершина $K(x)$ перенесена из начала координат в точку x. Содержательно конус $K(x)$ представляет собой множество градиентов $w \in \mathbb{R}^m$, при которых решение задачи ЛП (для $c = w$) достигается в вершине x. Поэтому *задача* (7.1) *равносильна поиску конуса* $K(x)$, *в котором лежит градиент с линейного функционала* $\langle c, x\rangle$. Еще раз. Конусы образуют

[2)] Точнее говоря, соответствует множеству гамильтоновых контуров.

[3)] В варианте распознавания $\langle c, x\rangle \to \max$ заменяется вопросом о разрешимости неравенства $\langle c, x\rangle \geqslant b$.

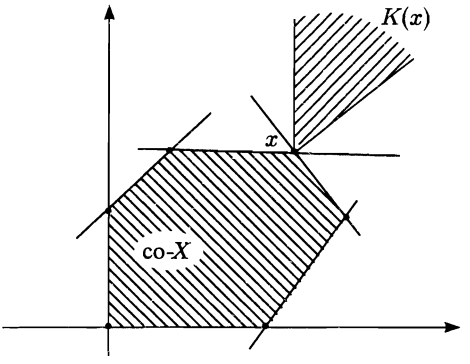


Рис. 7.1

разбиение пространства (рис. 7.2),

$$\bigcup K(x) = \mathbb{R}^m, \tag{7.2}$$

и задача (7.1) состоит в определении конуса, удовлетворяющего условию

$$c \in K(x). \tag{7.3}$$

Дальнейший сюжет развивается вокруг конусного разбиения (7.2), каковое может быть — и бывает — весьма непривычным с интуитивной точки зрения. Сюрпризная сторона явления здесь связана с существованием «аномальных» многогранников, открытых *Каратеодори* и переоткрытых *Гейлом* [5, 14], у которых все

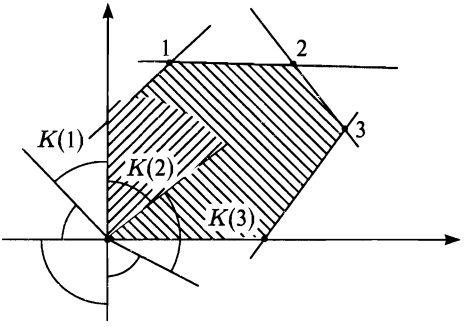


Рис. 7.2

вершины смежные[4]. Конусный букет (7.2), построенный по такому многограннику, обладает тем свойством, что любые два конуса граничат друг с другом по $(m-1)$-мерной грани. В результате, если алгоритм бракует часть конусов, деля пространство $\mathbb{R}^m$ на две части плоскостями, то на каждом шаге может быть отброшен лишь один конус — и полного перебора избежать принципиально невозможно.

На этом феномене высокой *плотности графа*[5] со-X базируется идея отделения простых задач от сложных. Не все, конечно, так просто (см. далее), но вчерне возникающая картина выглядит убедительно, особенно если не увлекаться оговорками. NP-полные задачи, как выясняется, имеют высокую *плотность графа*, тогда как полиномиальные — низкую. Более того, все комбинаторные алгоритмы — за редкими исключениями[6] — оказываются *алгоритмами прямого типа* (раздел 7.5), действующими с помощью линейных сравнений и способными гарантированно отбрасывать не более одного конуса (не более одной вершины многогранника). При высоких *плотностях* такие алгоритмы в принципе не позволяют избежать экспоненциального перебора. В то же время именно среди таких алгоритмов велся поиск полиномиального решения NP-задач. Искали, как говорится, «под фонарем». Поэтому аргумент в пользу $\mathrm{NP} \neq \mathrm{P}$, состоящий в устойчивой безуспешности доказательства $\mathrm{NP} = \mathrm{P}$, скорее всего, не аргумент — ибо не там искали. Сказанное, разумеется, нуждается в пояснениях, что требует некоторого углубления в дебри многомерной геометрии, где интуиция не столько помогает, сколько мешает.

7.2. Выпуклые многогранники

Иллюзия очевидности при изучении выпуклого анализа, возникающая из опыта размышлений над $\mathbb{R}^2$ или $\mathbb{R}^3$, в бóльших размерностях быстро рассыпается. Кое-

[4] Иначе говоря, любые две вершины соединены ребром, что возможно в размерностях $\geqslant 4$, — см. раздел 7.3.

[5] Упоминание графа со-X подразумевает граф, узлами и дугами которого являются вершины и ребра многогранника со-X.

[6] Собственно, говорить можно о двух исключениях. О полиномиальных методах решения задач ЛИНЕЙНОЕ ПРОГРАММИРОВАНИЕ и ПРОСТОЕ ЧИСЛО.

что остается, безусловно, в неизменном виде, но появляются совершенно новые явления, как-будто из другого мира, — и тогда избавление от старого опыта превращается в трудноодолимый барьер.

Напомним простейшие определения и факты [19].

Множество Ω называется *выпуклым*, если вместе с любыми двумя точками $x, y \in \Omega$ содержит отрезок,

$$\lambda x + (1 - \lambda)y \in \Omega, \quad \lambda \in [0, 1],$$

их соединяющий.

Линейную комбинацию $\sum_{j=1}^{N} \lambda_j \mathbf{x}_j$ точек $\{\mathbf{x}_1, \ldots, \mathbf{x}_N\}$, при условии $\sum_{j=1}^{N} \lambda_j = 1$, где все $\lambda_j \geqslant 0$, называют *выпуклой*. Объединение всех выпуклых комбинаций *конечных* наборов точек из $\Omega \subset \mathbb{R}^m$ называется *выпуклой оболочкой* [7] множества Ω и обозначается $\mathrm{co}\text{-}\Omega$.

7.2.1. *Выпуклая оболочка всегда совпадает с объединением всевозможных выпуклых комбинаций конечных подмножеств из* $\mathbb{R}^m$, *содержащих не более* $m + 1$ *точек* [8].

Множество $\mathcal{A} \subset \mathbb{R}^m$ называется *аффинным*, если вместе с любыми двумя различными точками оно содержит проходящую через них прямую. В случае $0 \subset \mathcal{A}$ аффинное множество $\mathcal{A}$ является линейным пространством. Всякое аффинное множество — есть пересечение *гиперплоскостей*, описываемых уравнениями вида $\langle a, x\rangle = \beta$. Совокупность решений системы линейных уравнений $Ax = b$ — есть аффинное множество.

Отображение S из R^n в $\mathbb{R}^m$, удовлетворяющее условию

$$S(\alpha x + \beta y) = \alpha S x + \beta S y, \quad (\alpha + \beta = 1),$$

называется *аффинным*. Аффинное отображение — это всегда преобразование вида $S(x) = Ax + b$, где A — линейное отображение.

[7] Выпуклая оболочка Ω равносильно определяется как минимальное по включению выпуклое множество, содержащее Ω.

[8] Это *теорема Каратеодори*.

Отделимость. Выпуклые множества $X, Y \subset \mathbb{R}^m$ называются *отделимыми* (*строго отделимыми*), если существует *разделяющая гиперплоскость* $H \subset \mathbb{R}^m = \{x : \langle a, x\rangle = \beta\}$, относительно которой множества располагаются в разных замкнутых (открытых) полупространствах H^+, H^-:

$$x \in X \quad \Rightarrow \quad \langle a, x\rangle \leqslant \beta \qquad (< \beta),$$
$$x \in Y \quad \Rightarrow \quad \langle a, x\rangle \geqslant \beta \qquad (> \beta).$$

7.2.2. Теорема отделимости. *Непересекающиеся замкнутые выпуклые множества X, Y всегда отделимы, и — строго отделимы, если хотя бы одно их этих множеств ограничено.*

Точка, не принадлежащая замкнутому выпуклому множеству X, строго отделяется от X. **(?)**

Опорной гиперплоскостью к X называют гиперплоскость H, которая имеет хотя бы одну общую точку с X, и либо $X \subset H^+$, либо $X \subset H^-$.

• *У выпуклого множества всегда существует опорная гиперплоскость, проходящая через любую наперед заданную граничную точку.* Такое свойство, кстати, равносильно определению выпуклости.

• *Любое замкнутое выпуклое множество совпадает с пересечением всех содержащих его замкнутых полупространств.*

Конусы. Замкнутое выпуклое множество $K \subset \mathbb{R}^m$, содержащее вместе с любой точкой $x \in K$ луч λx, проходящий через эту точку, называется *конусом*.

Многогранники. Выпуклую оболочку конечного множества точек называют *многогранником*.

Пересечение F многогранника P с любой своей опорной гиперплоскостью называют *гранью многогранника*; k-*гранью*, если $\dim F = k$; 0-грани — *вершины*, 1-грани — *ребра*. Грань максимальной размерности называют *фасетой* — у m-мерного многогранника фасеты имеют размерность $m - 1$. Совокупность вершин и ребер называют ***графом многогранника***.

• *Каждая грань многогранника, в свою очередь, является многогранником.*

• *Всякий многогранник совпадает с выпуклой оболочкой своих вершин.*

Пересечение конечного числа замкнутых полупространств — *полиэдр*. Ограниченный полиэдр — многогранник. Множество решений системы линейных неравенств $Ax \leqslant b$ — полиэдр.

Принципиальную роль в теории многогранников играет понятие *крайнего множества*.

7.2.3. *Выпуклое подмножество E выпуклого множества X называется* ***крайним****, если внутренняя точка любого отрезка $[a, b]$ из X принадлежит E вместе с $[a, b]$. Крайнее подмножество X, состоящее из одной точки, называется* ***крайней точкой****.*

Крайние подмножества многогранника — суть грани многогранника, крайние точки — вершины.

Идеологическая прозрачность данных определений и фактов довольно обманчива. Теория многогранников изобилует нерешенными проблемами. Внешне все выглядит весьма просто. Имеется N точек $\{\mathbf{x}_1, \ldots, \mathbf{x}_N\} \subset \mathbb{R}^m$. Выпуклые комбинации их подмножеств исчерпывают все грани и сам многогранник. Это все, что потенциально интересует. Но здесь часто нелегко ответить на простейшие с виду вопросы. Будет ли некоторая точка из $\{\mathbf{x}_1, \ldots, \mathbf{x}_N\}$ вершиной многогранника? Тем более трудно сказать, будут ли некоторые вершины $\mathbf{x}_i$, $\mathbf{x}_j$ смежными (соединены ребром)? Например, вопрос о смежности вершин в многограннике КОММИВОЯЖЕРА оказывается сам по себе NP-полной задачей [9]. Трудности, таким образом, имеют алгоритмическую природу. Академический ракурс тривиален:

7.2.4. *Точка $\mathbf{x}_i$ является вершиной многогранника*

$$\operatorname{co}\{\mathbf{x}_1, \ldots, \mathbf{x}_N\} \tag{7.4}$$

в том случае, когда $\{\mathbf{x}_i\}$ и $\{\mathbf{x}_1, \ldots, \mathbf{x}_{i-1}, \mathbf{x}_{i+1}, \ldots, \mathbf{x}_N\}$ строго отделимы.

[9] *Papadimitriou C. H.* The adjacency relation on the traveling salesman polytope is NP-complete. Math. Prog. 1978, 312–324.

7.2.5. *Две вершины многогранника* (7.4) *являются смежными в томм случае, когда они строго отделяются от остальных вершин.*

Что касается предостережения в начале раздела о возможности неожиданных явлений, то в данном контексте это связано главным образом со следующим фактом.

7.2.6. *В* $\mathbb{R}^m$ *при* $m \geqslant 4$ *существуют выпуклые многогранники с произвольным количеством смежных вершин.*

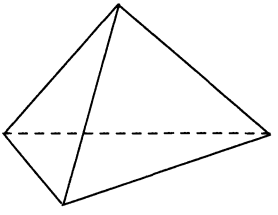


Рис. 7.3

Дабы получить удовольствие, полезно начать размышление издалека. Многогранник, у которого все вершины смежные, сколько вершин может иметь максимально? На плоскости $\mathbb{R}^2$ указанным свойством обладает треугольник — максимальное число вершин $m = 3$. В $\mathbb{R}^3$ — тетраэдр $(m = 4)$. В пространстве четырех измерений и выше такой многогранник *может иметь вершин сколько угодно*! Вопрос рассматривается в следующем разделе. Особое внимание стоит обратить на тот факт, что речь идет не об аномалии, а о характерном для $m > 3$ явлении.

7.3. Циклические многогранники

Особенно приятно, когда неожиданное имеет простой облик. Рассмотрим так называемый *циклический многогранник* в $\mathbb{R}^4$ с любым числом вершин, лежащих на кривой

$$x(\tau) = \{\tau,\ \tau^2,\ \tau^3,\ \tau^4\}, \quad \tau \geqslant 0.$$

Удивительное свойство кривой $x(\tau)$ заключается в следующем. Через любые две точки $x(t)$, $x(s)$ можно так провести гиперплоскость H_{ts} (в данном случае трехмерную), что все остальные точки $x(\tau)$ будут находиться по одну сторону H_{ts}.

Действительно,

$$\varphi(\tau) = (\tau - t)^2(\tau - s)^2 = \alpha_0 + \alpha_1\tau + \alpha_2\tau^2 + \alpha_3\tau^3 + \alpha_4\tau^4 \geqslant 0.$$

Поэтому вся кривая $x(\tau)$ лежит по одну сторону гиперплоскости H_{ts}, описываемой уравнением

$$\langle h, x(\tau)\rangle + \alpha_0 = 0, \quad h = \{\alpha_1, \alpha_2, \alpha_3, \alpha_4\},$$

и лишь две точки $x(t)$, $x(s)$ попадают на саму гиперплоскость H_{ts}, поскольку

$$\varphi(t) = \varphi(s) = 0.$$

Далее остается вспомнить, что многогранник является выпуклой оболочкой своих вершин, в данном случае точек

$$x(\tau_1), \ \ldots, \ x(\tau_N). \tag{7.5}$$

Все плоскости $H_{\tau_i\tau_j}$, $\tau_i \neq \tau_j$, *опорные* — поэтому отрезки, соединяющие любые две точки из (7.5), будут ребрами циклического многогранника [10].

Аналогичный пример в $\mathbb{R}^m$, $m > 4$, получается из данного. В качестве вершин можно взять любые N точек, у которых первые четыре координаты совпадают с координатами точек (7.5).

У циклического многогранника C_{mN}, где m — размерность многогранника, N — число вершин, не только все вершины попарно смежны, но и любые $k = [m/2]$ — образуют k-грань. Скажем, в $\mathbb{R}^{15}$ выпуклая оболочка любых $k \leqslant 7$ точек на кривой

$$x(\tau) = \{\tau, \ \tau^2, \ \ldots, \ \tau^{15}\}$$

образует k-грань циклического многогранника, независимо от количества остальных вершин.

Число фасет у C_{mN} растет экспоненциально:

$$\nu_{mN} = \begin{cases} \dfrac{N(N-j-1)!}{j!(N-2j)!}, & \text{если} \quad m = 2j, \\ \dfrac{2(N-j-1)!}{j!(N-2j-1)!}, & \text{если} \quad m = 2j+1. \end{cases}$$

Экстремальные свойства циклических многогранников C_{mN} характеризуются *максимальным числом граней любой размерности среди всех многогранников с N вершинами в $\mathbb{R}^m$*. Как говорится, хуже не бывает.

[10] Ясно, что отрезок, соединяющий $x(t)$ и $x(s)$, весь лежит в H, и ни одна его точка не может быть получена как выпуклая комбинация остальных вершин (7.5), поскольку те лежат по одну сторону H.

Многогранники с высокой плотностью графа, разумеется, не обязаны быть циклическими [11], — и возникает вопрос, насколько широко распространены такие многогранники. Довольно неожиданно, что при больших размерностях «аномалия» становится правилом. Вот один из вариантов точного утверждения [3]. Выберем наугад (равновероятно) k вершин m-мерного куба $[0, 1]^m$, и пусть X обозначает их выпуклую оболочку, а p_{km} — вероятность того, что все вершины многогранника X попарно смежны.

7.3.1. *При $m \geqslant 4$ и $k \leqslant 2^m$ справедлива оценка*

$$p_{km} > 1 - \frac{k^4}{4}\left(\frac{5}{8}\right)^m. \tag{7.6}$$

◀ Любые две вершины X смежны, если для любых двух пар (x, y), (u, v) из выбранных вершин куба нарушается хотя бы одно из условий

$$x \wedge y \leqslant u \leqslant x \vee y \quad \text{или} \quad x \wedge y \leqslant v \leqslant x \vee y, \tag{7.7}$$

где $\wedge$, $\vee$ обозначают покоординатное логическое, соответственно, умножение и сложение $(0, 1)$-векторов.

Вероятность выполнения обоих условий (7.7) равна $q_m = (r_m)^m$, где r_m — вероятность выполнения этих же условий по одной координате. По *формуле полной вероятности* [6, т. 4]:

$$r_m = \frac{2^{m-1}+1}{2^m} + \left(\frac{2^{m-1}-1}{2^m}\right)\left(\frac{2^{m-1}-2}{2^m}\right)\left(\frac{2^{m-1}-3}{2^m}\right) < \frac{5}{8}.$$

Остается заметить, что число различных пар (x, y), (u, v) среди k вершин равно

$$\left[\frac{k(k-1)}{2}\right]^2 < \frac{k^4}{4}.$$

В итоге получается оценка (7.6). ▶

В соответствии с (7.6) получается, что взятые наугад 10^5 вершин 100-мерного куба образуют многогранник, у которого с вероятностью более 0,99 все *сто тысяч* вершин попарно смежны.

7.4. Линейные разделяющие деревья

Комбинаторные алгоритмы обычно работают на основе линейных сравнений. Решая задачу $[X, c]$, т. е. (7.1), алгоритм на каждом шаге

[11] См. статью *Д. Гейла* в сборнике [14].

проверяет неравенство $\langle f_i, c\rangle > 0$ и переходит к следующему тесту $\langle f_{i+1}, c\rangle > 0$, где выбор функционала f_{i+1} происходит дихотомически — в зависимости от ответа «да» или «нет» на предыдущий вопрос о справедливости неравенства $\langle f_i, c\rangle > 0$.

На рис. 7.4 приведена схема работы алгоритма, сортирующего массив из трех чисел. Проверка неравенств типа $u < v$, т. е. $v - u > 0$, представляет собой как раз линейное тестирование с помощью функционала $\langle f_i, \cdot\rangle$, где $f_i = \{-1, 1, 0\}$. Ветвлению при ответах «нет» соответствуют пунктирные ребра. Схематично то же самое происходит и в общем случае, но рисунок типа 7.4 отражает лишь каркас работы алгоритма — его *линейное разделяющее дерево* (ЛРД). Детали, иногда громоздкие, остаются за кулисами. *Жадный алгоритм*, например, решающий задачу МОД, помимо линейных сравнений проверяет постоянно, не образуются ли циклы при добавлении ребер и не охвачены ли уже все узлы (правило остановки), — что находится за рамками *бинарного разделяющего каркаса*. Но именно *глубина оптимального ЛРД* [12] свидетельствует о трудоемкости алгоритма — по крайней мере, об оценке трудоемкости снизу.

ЛРД разбивает $\mathbb{R}^m$ плоскостями $\langle f_i, c\rangle = 0$, и любая ветвь β, заканчивающаяся в висячей вершине, отвечает последовательности полупространств $\langle f_i, c\rangle \lessgtr 0$, вырезающих в $\mathbb{R}^m$ конус K_β, в который попадает вектор $c \in K_\beta$. Причем $K_\beta \subset K(x)$, где $K(x)$ — конус разбиения (7.2). В итоге $c \in K_\beta \subset K(x)$, — и алгоритм, основанный на ЛРД, обеспечивает выбор решения $x \in X$ для задачи $[X, c]$.

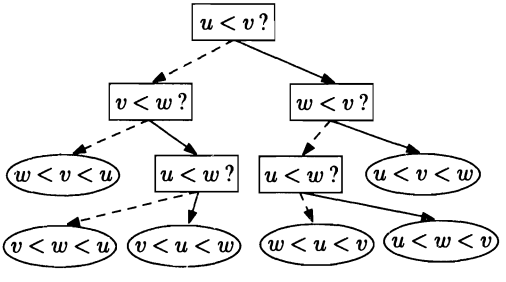


Рис. 7.4

[12] *Глубина дерева* — максимальная длина ветви до висячей вершины. Оптимальным ЛРД считается дерево, имеющее минимальную глубину.

7.4.1. Теорема Мошкова[13]. *Оптимальное ЛРД для любой задачи* $[X, c]$ *имеет полиномиальную глубину*

$$D(X) = O\big(m^3 \log |X|3\big). \tag{7.8}$$

Оценка (7.8), и вообще теорема 7.4.1, воспринимается поначалу как NP = P, ибо число необходимых сравнений оказывается полиномиальным, а это как раз и составляет основной вклад в трудоемкость во всех известных алгоритмах. Беда заключается в том, что ЛРД — это не алгоритм, а виртуальный каркас. ЛРД существует, но кто будет подсказывать функционалы f_i? Если *оракул*, — то его негде взять. Конечно, можно обойтись без потусторонних сил, утверждая существование подходящих правил вычисления $f_{i+1} = \Theta_X(f_i, r_i)$, где r_i — ответ «да» или «нет» на предыдущем шаге. И тогда, казалось бы, можно гарантировать существование полиномиального алгоритма в принципе, не располагая конкретизацией $\Theta_X(\cdot)$, — чего было бы достаточно для эстетического удовлетворения. Но где гарантия, что функция $\Theta_X(\cdot)$ полиномиально вычислима? Хуже того, где взять полиномиально вычислимое правило остановки алгоритма? Дерево деревом, но алгоритм вынужден работать в другой среде, опираясь на содержательные данные исходной постановки задачи.

Короче говоря, наличие линейного разделяющего дерева ничего не дает для построения алгоритма. Практически ничего. Но теоретически — существование полиномиального ЛРД говорит о том, что на пути создания полиномиального алгоритма нет препятствий принципиального характера, каковые могли бы перечеркнуть надежды. И если уж говорить об ожиданиях, то на данном уровне непонимания — возникают аргументы, скорее, в пользу NP = P, хотя градус алхимического стиля мышления при этом — довольно высок.

Теорема 7.4.1 представляет собой достаточно глубокий геометрический факт. Дело вовсе не в том, что n линейных сравнений в задаче $[X, c]$ разбивают телесный угол в $\mathbb{R}^m$ на 2^n узких пучков, и поэтому, дескать, полиномиальная глубина ЛРД естественна.

[13] *Мошков М. Ю.* Об условных тестах // ДАН СССР. 1982. **265**, № 3. 550–552.

Феномен заключается в другом. При полиномиальной глубине линейного разделяющего дерева действительно обеспечивается достаточно мелкая нарезка $\mathbb{R}^m$ экспоненциальным числом плоскостей, но конец любой ветви дает конус $K_\beta \subset K(x)$ с полиномиальным числом граней [14], причем конусы K_β оказываются так уложены, что в точности исчерпывают каждый $K(x)$. Сама по себе возможность подобного разрезания пространства, может быть, не так удивительна. Неожиданной оказывается возможность «добраться» до любого нужного конуса (типа K_β) с помощью бинарной процедуры полиномиальной длины. Такова геометрическая структура задачи, и она (структура) не только не создает препятствий для NP = P, но и подталкивает мысль в соответствующем направлении.

7.5. Алгоритмы прямого типа

Все известные алгоритмы решения комбинаторных задач базируются на линейных сравнениях [15], но в более приземленном исполнении, чем это подразумевалось за кадром ЛРД. Суть дела удобнее всего объяснить, опираясь опять-таки на конусное разбиение (7.2) с ориентацией на условие (7.3).

Движение по ветви ЛРД подразумевает исключение каждый раз одного из полупространств $\langle f_i, c\rangle > 0$ или $\langle f_i, c\rangle < 0$. *Алгоритм прямого типа* также проводит линейное сравнение

$$\langle f_i, c\rangle > 0 \quad (?),$$

но исключает не все полупространство

$$H^- = \{c : \langle f_i, c\rangle < 0\},$$

а лишь конусы $K(x)$ разбиения (7.2) [16], лежащие в H^-. И если конусное разбиение (7.2) таково, что все конусы попарно граничат друг с другом, — отсечь более одного конуса за один раз не удается. Это, разумеется, крайняя ситуация. Но алгоритм остается переборным и в том случае, когда бóльшая часть конусов $K(x)$ попарно

[14] Равным длине ветви. В то время как число граней $K(x)$ может расти экспоненциально, — например, в случае циклического многогранника.

[15] Упоминая «все» алгоритмы, мы подразумеваем два исключения: полиномиальные методы решения задач ЛИНЕЙНОЕ ПРОГРАММИРОВАНИЕ и ПРОСТОЕ ЧИСЛО.

[16] Это не совсем точное определение *алгоритма прямого типа* [3], но оно выделяет главное.

граничат друг с другом — иначе говоря, когда плотность графа многогранника со-X экспоненциальна.

Исключение конусов $K(x)$ происходит, конечно, иносказательно. Фактически отсекаются вершины многогранника со-X — и от комбинаторного алгоритма, манипулирующего *комбинациями*, т. е. вершинами, было бы трудно ожидать чего-либо другого.

Из сказанного ясно, что *алгоритмы прямого типа принципиально не могут решать за полиномиальное время задачи, имеющие высокую плотность графа многогранника* со-X. Последнее (высокая плотность графа) установлено для многих NP-полных задач. С другой стороны, установлено [4], что *все встречавшиеся в литературе алгоритмы решения комбинаторных задач — являются алгоритмами прямого типа*[17].

Разумеется, «все» за исключением полиномиальных алгоритмов для задач ЛИНЕЙНОЕ ПРОГРАММИРОВАНИЕ и ПРОСТОЕ ЧИСЛО, — что оговаривалось выше. Но исключительность этих двух случаев более-менее очевидна сама по себе. И дело не только в том, что обе задачи долгое время были под вопросом в смысле принадлежности классу P. Многогранник задачи ЛП, например, может иметь граф с экспоненциальной плотностью (раздел 7.6), что в принципе исключает возможность полиномиального решения ЛП *алгоритмом прямого типа*. Полиномиальный алгоритм мог найтись лишь среди нелинейных, либо линейных, но не чисто бинарных.

К ЛП можно было бы добавить еще одно исключение [3]. Максимизация многочлена

$$z(t) = c_1 t + c_2 t^2 + c_3 t^3 + c_4 t^4 \tag{7.9}$$

на множестве точек $\{t_1 < t_2 < \ldots < t_N\}$ равносильна максимизации $\langle c, x\rangle$ на множестве $X = \{x_1, \ldots, x_N\} \subset \mathbb{R}^4$, где

$$x_i = \{t_i, t_i^2, t_i^3, t_i^4\}, \quad i = 1, \ldots, N,$$

[17] Среди рассмотренных: алгоритмы сортировки, «жадный алгоритм», венгерский метод для задачи о назначениях, различные реализации *динамического программирования* (для задачи о кратчайшем пути и КОММИВОЯЖЕРА), а также *метода ветвей и границ*.

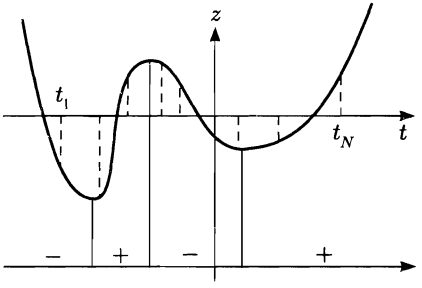


Рис. 7.5

а значит, равносильна максимизации $\langle c, x\rangle$ на *циклическом многограннике* со-X, у которого все вершины смежные. Несмотря на это, максимум (7.9) легко (полиномиально) находится. График полинома $z(t)$ изображен на рис. 7.5 (для случая $c_4 > 0$). Ось t элементарно разбивается на четыре промежутка: на двух «минусовых» $z(t)$ убывает, на «плюсовых» — возрастает. На «минусовых» надо взять самые левые значения из $\{t_1, \ldots, t_N\}$, на «плюсовых» — самые правые, после чего из четырех значений $z(\cdot)$ выбрать наибольшее. Рецепт полиномиален и очень прост, но это, разумеется, не алгоритм прямого типа.

Пример, тем не менее, бьет мимо цели. Не будь значения $\{t_1, \ldots, t_N\}$ изначально упорядочены, их пришлось бы сортировать — и трудоемкость решения стала бы экспоненциальной [18]. Поэтому секрет успеха здесь состоит в предварительной организации данных. Другое дело, что данные для рассматриваемой задачи оказывается возможным организовать не «под задачу», а универсально, для любого набора $\{t_1, \ldots, t_N\}$.

Гораздо хуже другое. Задача максимизации (7.9) по своей природе не является комбинаторной. Здесь нет «комбинаторных атомов» (см. раздел 1.9) и нет стенографической экономности задания списка. *Длина входа* — битового описания набора $\{t_1, \ldots, t_N\}$ — пропорциональна $N \log_2 N$, и разговор о полиномиальности алгоритма выглядит даже странным. Потому что речь идет о полиномиальной зависимости не от длины входа, а от $\log_2 N$. Иначе говоря,

[18] Трудоемкость сортировки $\sim N \log_2 N$, тогда как в данном случае речь идет о полиномиальности по $\log_2 N$.

входные данные избыточны, не все нужны. Из миллиона точек $\{t_1, \ldots, t_N\}$ для решения задачи требуются всего — четыре. Их бы надо было искать, сортируя $\{t_1, \ldots, t_N\}$, но тут как раз спасает упорядоченность [19].

Что касается фактора организации данных, то в обычной теории информации он не играет роли, ибо трудоемкость приведения информации в определенный порядок там не учитывается. В теории алгоритмов ситуация иная — предпочтительны «упорядоченные биты». Если бы, скажем, в задаче ЛП ограничения выписывались, начиная с фасет, определяющих оптимальную вершину, задача решалась бы в один шаг. Или логические функции представлялись бы в какой-либо удобной форме вместо КНФ. Например, в виде ДНФ, где решение ВЫПОЛНИМОСТИ видно невооруженным взглядом. Но тогда ясно, что преобразование КНФ $\Rightarrow$ ДНФ само по себе обязано быть труднорешаемой задачей.

7.6. Релаксационные многогранники

Если в ограничениях комбинаторных задач целочисленным переменным разрешить принимать непрерывные значения [20], возникают описания так называемых *релаксационных многогранников*. Вот один из простейших примеров.

Пусть координаты вектора $x \in \mathbb{R}^m$, $m = n^3$, имеют трехиндексную нумерацию. Рассмотрим многогранник M_n, определяемый системой ограничений [21]

$$\sum_{i,j=1}^{n} x_{ijk} = \sum_{i,k=1}^{n} x_{ijk} = \sum_{j,k=1}^{n} x_{ijk} = 1, \quad x_{ijk} \geqslant 0. \tag{7.10}$$

При условии целочисленности вершины многогранника M_n образуют множество X_n, соответствующее задаче ТРЕХМЕРНОЕ СОЧЕТАНИЕ, — и граф со-X_n имеет экспоненциальную плотность [4]. В то же время M_n имеет и нецелочисленные вершины [9]. При

[19] Пример (7.9) сохраняет, тем не менее, идеологическое значение, высвечивая некоторые стороны рассматриваемой проблематики. Определенный интерес представляет композиция (7.9) с какой нибудь чисто комбинаторной схемой, в которой список $\{t_1, \ldots, t_N\}$ задается комбинаторно.

[20] Освободить их (relax) от необходимости быть дискретными.

[21] M_n принадлежит классу трехиндексных транспортных многогранников.

этом ребра многогранника со-X_n являются ребрами M_n, т. е. граф со-X_n — подграф графа M_n. Поэтому *граф M_n также имеет экспоненциальную по n плотность*, — и это весьма примечательный факт, ибо *задача ЛП*

$$\langle c, x\rangle \to \max, \quad x \in M_n$$

при фасетном описании (7.10) *имеет допустимый многогранник с экспоненциальной плотностью вершин, но решается — полиномиально (как любая задача ЛП).*

> *Принципиальность данного результата подчеркивает извечная туманность взаимосвязей между фасетным и вершинным описанием многогранников. Разумеется, многогранники с экспоненциальной плотностью вершин существуют (раздел 7.3), но возможно ли их полиномиальное фасетное описание? Оказывается, возможно — как показывает пример* (7.10).

Многогранник Бондаренко—Падберга. Существенно более интересна другая геометрическая конструкция. В $\mathbb{R}^m$, где $m = 4n^2$, обозначим координаты точек x_{ij}, y_{ij}, z_{ij}, t_{ij}, и определим многогранник $\mathcal{M}_n$ системой ограничений [22]:

$$\begin{cases} x_{ij} + y_{ij} + z_{ij} + t_{ij} = 1, \\ x_{ij} + y_{ij} = x_{kj} + y_{kj}, \\ x_{ij} + z_{ij} = x_{ik} + z_{ik}, \\ x_{ij} = x_{ji}, \quad t_{ij} = t_{ji}, \quad y_{ij} = z_{ji}, \\ z_{ii} = y_{ii} = 0, \\ x_{ij} \geqslant 0, \quad y_{ij} \geqslant 0, \quad z_{ij} \geqslant 0, \quad t_{ij} \geqslant 0, \end{cases} \tag{7.11}$$

где i, j, k независимо пробегают значения $1, \ldots, n$.

[22] Впервые многогранник $\mathcal{M}_n$ рассматривался в статье: *Бондаренко В. А.* Об одном комбинаторном многограннике // Моделирование и анализ вычислительных систем. Сб. научн. тр. Яросл. гос. ун-та. Ярославль, Изд-во Яросл. гос. ун-та, 1987. с. 133–134. См. также: *Padberg M. V.* The boolean quadratic polytope: some characteristics, facets and relatives // Math. Programming. 1989. **45**. 139–172; *Бондаренко В. А.* Об одном классе многогранников и их использовании в комбинаторной оптимизации // ДАН СССР. 1993. **328**, № 3. 303–304.

Многогранник $\mathcal{M}_n$ обладает определенными свойствами, выделяющими его в разряд весьма перспективных объектов исследования [8]. Как и в случае (7.10), *общее число ограничений в* (7.11) *полиномиально по* n, *а целочисленные вершины многогранника* $\mathcal{M}_n$ *попарно смежны и образуют множество* $\mathcal{X}_n$ *из* 2^n *точек с координатами*:

$$x_{ij} = x_{ii}x_{jj}, \qquad y_{ij} = (1 - x_{ii})x_{jj},$$
$$z_{ij} = x_{ii}(1 - x_{jj}), \quad t_{ij} = (1 - x_{ii})(1 - x_{jj}),$$

где $\{x_{11}, \ldots, x_{nn}\} \in \mathbb{R}^n$ — *произвольный вектор из нулей и единиц.*

Поэтому, аналогично ситуации порождаемой M_n, здесь также возникает пример полиномиально решаемой задачи ЛП

$$\langle c, x\rangle \to \max, \quad x \in \mathcal{M}_n,$$

при экспоненциальной плотности графа допустимого многогранника [23].

Выгоды рассмотрения $\mathcal{M}_n$ вместо M_n появляются в связи с рассмотрением нецелочисленных вершин, каковые в случае $\mathcal{M}_n$ эффективно описываются. Эффективность перечисления снимает принципиальное различие между целочисленными и нецелочисленными вершинами, и дает возможность ставить комбинаторные задачи *на всех* вершинах $\mathcal{M}_n$, либо на их подмножествах, порождая азарт поиска на этом поле NP-полной задачи — что, в случае успеха, обеспечило бы NP = P. Некоторые попытки в этом направлении рассмотрены в [4].

7.7. Аффинная сводимость

Исследование комбинаторных характеристик многогранников обычно трудоемко, и на фоне полиномиальной сводимости задач возникает соблазн сэкономить усилия, не производя «вычисления» каждый раз заново. Подобную возможность дает понятие *аффинной сводимости* [24], поскольку NP-*полные задачи* часто сводятся одна

[23] Разумеется, это фигура речи. Подразумевается семейство задач ЛП, соответственно, семейство допустимых многогранников.

[24] Представляющее собой частный случай полиномиальной сводимости.

к другой *аффинным преобразованием* вида

$$\mathcal{A}(x) = Ax + a, \tag{7.12}$$

где A — прямоугольная $m \times n$ матрица[25].

Транспортная задача (*задача о назначениях*), каковой соответствует многогранник

$$P = \text{со-}X = \left\{ [x_{ij}] : \sum_i x_{ij} = \sum_j x_{ij} = 1,\ x_{ij} \geqslant 0 \right\} \subset \mathbb{R}^{n^2},$$

аффинным отображением

$$y_i = nx_{i1} + (n-1)x_{i2} + \ldots + 1 \cdot x_{i1},$$

как легко убедиться, переводится в многогранник $P' = \text{со-}Y$ *задачи сортировки*, где Y — множество векторов-перестановок $(i_1, \ldots, i_n)$ чисел $1, \ldots, n$.

Менее тривиальный пример аффинной сводимости дает теорема 4.3.1, вернее, ее доказательство, см. также коллекцию примеров в [4].

Аффинные преобразования сохраняют некоторые свойства многогранников, и в этом заключена идея извлечения выгод.

- *Для любого аффинного преобразования* $\mathcal{A} : \mathbb{R}^m \to \mathbb{R}^n$ *и любого множества* $X \subset \mathbb{R}^m$

$$\mathcal{A}(\text{со-}X) = \text{со-}\mathcal{A}(X).$$

Поэтому аффинный образ многогранника — тоже многогранник.

- *В случае аффинного преобразования каждая вершина многогранника* $P' = \mathcal{A}(P)$ *является образом некоторой грани многогранника* P.

- *Если числа вершин многогранников* P *и* $P' = \mathcal{A}(P)$ *совпадают*[26], *то граф многогранника* P' *является подграфом графа многогранника* P. *А если* P *и* P' *аффинно эквивалентны, то их графы изоморфны.*

[25] Подразумевается $\mathcal{A} : \mathbb{R}^m \to \mathbb{R}^n$, $m \leqslant n$. В случае $\operatorname{rank} A = m$ можно говорить об «обратном отображении» $\mathcal{A}^{-1}$, определенном на m-мерной гиперплоскости $\mathcal{A}(\mathbb{R}^m) \subset \mathbb{R}^n$.

[26] То есть аффинное отображение $\mathcal{A}$ обратимо (инъективно).

Таким образом, в случае инъективного аффинного преобразования у многогранников P и P' совпадают: *плотности графов*, *диаметры*, *степени вершин*, *число фасет*. В неинъективном случае ситуация менее благоприятна, но некоторые сравнения (больше/меньше) возможны.

7.8. Комментарии и дополнения

• Множество $2j + k$ точек в $\mathbb{R}^{k+1}$ назовем *равномерно распределенным*, если каждое открытое полупространство в $\mathbb{R}^{k+1}$ содержит не менее j точек этого множества.

7.8.1. *Для любых целых положительных j и k существует равномерно распределенное множество из $2j + k$ точек в $\mathbb{R}^{k+1}$.*

Факт 7.8.1 напрямую связан с существованием циклических многогранников[27] и дает новое освещение того же явления.

• Рассмотрим многогранник $\mathcal{M}_{n3} \subset \mathcal{M}_n$, удовлетворяющий, разумеется, ограничениям (7.11) и кроме того — дополнительным ограничениям для каждой тройки i, j, k, при условии $i < j < k$:

$$\begin{cases} x_{i,j} + t_{i,j} + x_{i,k} + t_{i,k} + y_{j,k} + z_{j,k} \leqslant 2, \\ x_{i,j} + t_{i,j} + y_{i,k} + z_{i,k} + x_{j,k} + t_{j,k} \leqslant 2, \\ y_{i,j} + z_{i,j} + x_{i,k} + t_{i,k} + x_{j,k} + t_{j,k} \leqslant 2, \\ y_{i,j} + z_{i,j} + y_{i,k} + z_{i,k} + y_{j,k} + z_{j,k} \leqslant 2. \end{cases} \tag{7.13}$$

Многогранник $\mathcal{M}_{n3}$ имеет те же целочисленные вершины, что и $\mathcal{M}_n$, число ограничений остается полиномиальным по n.

7.8.2. *Для многогранников $\mathcal{M}_{n3}$ задача распознавания целочисленности полиномиально разрешима*[28].

Под *задачей распознавания целочисленности* здесь подразумевается выяснение справедливости равенства

$$\max_{x \in M} \langle c, x \rangle = \max_{z \in Z} \langle c, z \rangle,$$

где Z — множество всех целых точек из выпуклого многогранника M.

---

[27] См. статью *Гейла* в сборнике [14].

[28] *Бондаренко В. А.*, *Урывaев Б. В.* Об одной задаче целочисленной оптимизации // АиТ. 2007. № 6. В этой же статье приведено доказательство контрастирующего с п. 7.8.2 утверждения 7.8.3.

• В пространстве $\mathbb{R}^s$ при $s = 6mn$ обозначим координаты точек

$$x_{ij}^{11},\ x_{ij}^{12},\ x_{ij}^{13},\ x_{ij}^{21},\ x_{ij}^{22},\ x_{ij}^{23}, \quad i \in \{1, 2, \ldots, m\}, \quad j \in \{1, 2, \ldots, n\},$$

и определим многогранник $\widetilde{\mathcal{M}}_{mn}$ системой ограничений:

$$\begin{cases} x_{ij}^{11} + x_{ij}^{12} + x_{ij}^{13} + x_{ij}^{21} + x_{ij}^{22} + x_{ij}^{23} = 1, \\ x_{ij}^{11} + x_{ij}^{12} + x_{ij}^{13} = x_{kj}^{11} + x_{kj}^{12} + x_{kj}^{13}, \\ x_{ij}^{11} + x_{ij}^{21} = x_{il}^{11} + x_{il}^{21}, \\ x_{ij}^{12} + x_{ij}^{22} = x_{il}^{12} + x_{il}^{22}, \\ x_{ij}^{11} \geqslant 0, \quad x_{ij}^{12} \geqslant 0, \quad x_{ij}^{13} \geqslant 0, \quad x_{ij}^{21} \geqslant 0, \quad x_{ij}^{22} \geqslant 0, \quad x_{ij}^{23} \geqslant 0, \end{cases} \tag{7.14}$$

где i, k независимо пробегают значения из $\{1, 2, \ldots, m\}$, а j, l — значения из $\{1, 2, \ldots, n\}$. Заметим, что общее число ограничений в (7.14) полиномиально по m, n.

7.8.3. *Для многогранников $\widetilde{\mathcal{M}}_{mn}$ задача распознавания целочисленности является NP-полной.*

Глава 8

Вероятностные алгоритмы

Там где нет уловимой причины,
Барабан лотереи не в счет,
Миром косвенно правят глубины,
Нисходя из небесных высот.

Хаотичный стиль поведения иногда решает проблему там, где жесткая дисциплина — хотелось бы написать, увлекает в пропасть. Но выгоды не столь разительные, чтобы говорить о сенсациях. Да, подбрасывание монетки довольно часто эффективнее детерминированного упрямства. И действуя наугад, временами быстрее идешь к цели. Но еще ни один вероятностный алгоритм не привел к скачку задачи из NP в P. Так что с принципиальной точки зрения, если замахиваться на водораздел между полиномиальными и «переборными» территориями, вероятностные уловки ничего не дают. Хотя какое-то время думалось по-другому — из-за простых чисел. А теперь так думается по инерции, но оснований уже нет — из-за AKS. Тем не менее выигрыш иногда возникает, и практика свидетельствует о широком использовании рандомизации в комбинаторных вычислениях. Причем возможность обеспечить вероятность *ошибки алгоритма*, меньшую чем — *ошибки счета*, фактически уравнивает случайные вычисления с детерминированными даже в идеологическом отношении. Кроме того, есть направления, в которых вероятностные алгоритмы открывают возможности революционного характера. В первую очередь это касается *интерактивных доказательств* с разветвлением на PCP-*проблематику* и *системы с нулевым разглашением*. Последние имеют отдаленное отношение к теории сложности алгоритмов, но где и как еще аукнутся — будущее покажет.

8.1. Напоминание о смешанных стратегиях

За обыденностью вероятностных инструментов есть опасность не рассмотреть экстраординарные возможности. Использование случайности иногда позволяет решить задачи, которые детерминированным образом вообще не решаются. Тезис достаточно красноречиво подтверждается феноменом *смешанных стратегий* [6, т. 4]. Напомним.

Пусть на рынке ценных бумаг имеется два типа акций, S_1 и S_2, прибыльность которых зависит от каких-то трудно прогнозируемых событий (принятие закона о таможенных пошлинах, война). Акция S_1 даст либо 8 %, либо 2 % прибыли; S_2, соответственно, — *минус* 4 % или *плюс* 14 %. Компактно ситуацию удобно записать в виде матрицы

$$\begin{bmatrix} 8 & -4 \\ 2 & 14 \end{bmatrix}. \tag{8.1}$$

Покупатель выбирает столбец, т. е. какие акции покупать; строка определяется неизвестными заранее обстоятельствами.

Покупать акции S_2 рискованно, а гарантированная прибыльность S_1 — всего 2 %. Тем не менее из ситуации легко «выжимается» 5 %. Действительно, если акции S_1, S_2 купить в количестве x_1, x_2 штук, то прибыль будет:

$$\textit{либо}\quad d_1 = \frac{1}{x_1 + x_2}(8x_1 - 4x_2)\,\%, \quad \textit{либо}\quad d_2 = \frac{1}{x_1 + x_2}(2x_1 + 14x_2)\,\%.$$

Если ориентироваться на гарантированную прибыль $\min\{d_1, d_2\}$, то ее максимум обеспечивается равенством $d_1 = d_2$, откуда следует $x_1 : x_2 = 3 : 1$. В этом случае

$$\min\{d_1, d_2\} = 5\,\%,$$

независимо от обстоятельств. *Покупка потенциально убыточных акций поднимает гарантированную прибыль с двух до пяти процентов.*

Теперь представим, что речь идет о покупке всего одной акции. Какой отдать предпочтение? Вся информация о задаче заключена в матрице (8.1), но для решения требуется привлечение посторонних средств. Необходимо произвести случайный эксперимент (бросить несимметричную монету) и купить акцию S_1 с вероятностью 3/4 или S_2 с вероятностью 1/4. Такая же ситуация возникает в *«русской рулетке»* [6, т. 7], где пистолет целесообразно выбирать случайным образом.

Обратим внимание, что понятие смешанной стратегии интуиции неведомо. Зачатки оптимизации Создатель вложил, а стереотип рационального теоретико-игрового поведения — нет. То ли было решено, что на динозавров и так можно охотиться, то ли было не до того. Но так или иначе, на генетическом уровне есть пробелы.

8.2. Интерактивные доказательства

Вероятностные алгоритмы опираются обычно на целенаправленное ветвление счета. Как правило, это дает эффект (раздел 8.4), но не радикальный.

На абстрактном уровне конкретная рецептура рассыпается, и остается чистый холст в виде *вероятностной машины Тьюринга* (ВМТ), которая работает, как и *недетерминированная*, — с той лишь разницей, что ветвление в (3.1) определяется с помощью подбрасывания монеты [1]. Понятно, что результаты «подбрасывания» можно записать на специальной ленте, и свести ВМТ к детерминированной машине Тьюринга с двумя лентами. Получается как бы два возможных режима работы, *on-line* и *off-line*.

Дополнение ВМТ *Оракулом* $\mathcal{O}$ порождает иногда принципиально новые возможности, возникающие, в частности, в *интерактивных системах* (раздел 4.9), где вместо ВМТ предпочитают говорить о *Вычислителе* $\mathcal{V}$. При этом «случайность» часто тянет одеяло на себя, настаивая на собственной решающей роли. Более сбалансированную оценку легче дать по конкретным примерам из раздела 6.8. В обоих рассмотренных там задачах *эффект нулевого разглашения* возникал благодаря *протоколу* взаимодействия *Оракула* с *Вычислителем*, но *протокол* существенно опирался на «случайность». И поэтому отвечать на вопрос, что главнее, здесь бессмысленно, не разобравшись предварительно в круговороте «яйцо — курица». Хотя «случайность», безусловно, один из незаменимых ингредиентов интерактивных систем.

Особого упоминания заслуживает роль *Оракула*. В задачах раздела 6.8 она была «никакая». Роль $\mathcal{O}$ там вполне мог выполнять толковый парень с тремя копейками в кармане, для подбрасывания.

[1] Поэтому чаще говорят о двухвариантном ветвлении типа (3.1).

Так что никаких чудодейственных талантов у $\mathcal{O}$ не предполагалось. Но вот пример иного толка.

Задача о неизоморфизме графов, $G_0 \not\sim G_1$, дополнительная к ИЗОМОРФИЗМУ ГРАФОВ (раздел 6.8). Проблема, вообще говоря, сложнее, и протокол, ведущий к решению, — другой.

По протоколу начинает *Вычислитель* $\mathcal{V}$, случайно выбирая $i \in \{0, 1\}$ и перестановку π вершин G_i, после чего посылает *Оракулу* граф $H = \pi(G_i)$ и требует определить индекс i. Тот отвечает: «j». Ответ принимается в случае $j = i$. Таких партий разыгрывается N штук.

Если графы не изоморфны, и *Оракул* владеет ситуацией, умея правильно отвечать на любой вопрос (а это предполагается), все N ответов будут приняты. В противном случае (изоморфности графов) с вероятностью 2^{-N} *Оракул* ошибется хотя бы один раз, и соответствующая проверка зажжет красную лампочку. При этом вероятность ошибки 2^{-N} может быть сделана сколь угодно малой выбором N независимо от размерности задачи.

Здесь уже в качестве $\mathcal{O}$ требуется чудотворец, способный в одно касание решать каждый раз задачу об изоморфизме G_i и $H = \pi(G_i)$, не зная случайной перестановки π.

Если допустить NP-полноту ИЗОМОРФИЗМА ГРАФОВ, а также NP $\neq$ P и co-NP $\neq$ NP, то данная интерактивная система обеспечивает фундаментальный прорыв, решая с помощью задачи из NP более сложную задачу из co-NP. Решение сколь угодно надежное, при достаточно большом N, но парадокса

$$\text{co-NP} \neq \text{NP} \quad \Rightarrow \quad \text{co-NP} = \text{NP},$$

все-таки не получается из-за вероятностной подоплеки [2].

Интересно, что примеры [3], рассмотренные здесь и в разделе 6.8, — это более-менее все, чем богата литература на тему «нулевого разглашения». Любопытным сие кажется потому, что лишь горстка переборных задач естественно вписываются в «нулевое разглашение», несмотря на то что все NP-задачи находятся под

[2] До окончательного вывода не хватает того самого зазора 2^{-N}.

[3] *Goldreich O., Micali S., Wigderson A.* Proofs that Yield Nothing but their Validity, or All Languages in NP have Zero-Knowledge Proof Systems // Journal of the ACM. 1991. **38**(1). 691–729.

этим колпаком, но для остальных — легче задействовать полиномиальную сводимость к РАСКРАСКЕ ГРАФА В 3 ЦВЕТА, нежели искать прямые пути. Нечто близкое по духу можно предполагать у ЛП насчет возможности избавления от комбинаторного характера постановки задачи (см. главу 5). И тогда в русле сказанного представляется целесообразным поиск подходящей NP-полной задачи аналогичного толка[4)], в том смысле, что не все NP-полные — одинаково хороши.

Если криптографические фокусы отложить в сторону, акценты смещаются. Интерактивные доказательства для ИЗОМОРФИЗМА ГРАФОВ и РАСКРАСКИ В 3 ЦВЕТА вообще теряют смысл, потому что *Оракул* знал решения, их можно было предъявить и полиномиально проверить. Все. А интерактивность с точки зрения трудоемкости — только лишние хлопоты. Другое дело задача о «*неизоморфизме*». Тут есть «выигрыш», но в кавычках — из-за сверхъестественных возможностей *Оракула*.

Последние все же годятся для теоретического использования, что реализуется как раз в конструкции IP-*класса*, который в разделе 4.9 определялся как *совокупность задач, полиномиально решаемых с помощью интерактивных доказательств*. Определение, надо признать, несколько с гуманитарным уклоном, но здесь не хотелось бы множить формальности, типа «сколько раз можно раз обращаться к *Оракулу*». По сути все правильно. Протокол любой, *Оракул* вычислительно всемогущ, диалог обязан занимать время, полиномиальное от длины входа. В результате

$$\boxed{\mathrm{IP} = \mathrm{PSPACE},} \qquad (8.2)$$

и хотя в IP сидит нереализуемый вычислительный гений, равенство (8.2) дает на PSPACE взгляд с другой стороны. А всякое иное суждение, даже с потусторонней позиции, нередко открывает новые обстоятельства.

Факт (8.2) преподносится иногда как выдающийся. Но, положа руку на сердце, надо признать результат не таким уж блистательным. Доказательство простое,

[4)] В надежде на NP = P.

перспективы больше иллюзорные. И хотя презумпция неэлементарности сильнее вдохновляет, она же и опустошает. Поэтому трезвый взгляд тоже необходим. Дело в том, что величие проблемы $\mathbf{NP} \stackrel{?}{=} \mathbf{P}$ создает вокруг атмосферу праздника, и все вблизи обретает дополнительную яркость. Флюиды ожидания открытий питают ажиотаж, вдохновляют до изнеможения, однако выталкивают процесс в насыщение и уравнивают большое с малым. А тогда из гор получается равнина.

8.3. PCP-проблематика

Ограничения на протокол интерактивного взаимодействия порождают новые классы внутри IP, среди которых наиболее известен *класс* PCP[5], определяемый как совокупность задач, решаемых следующей интерактивной системой.

- Система представляет собой тандем *Вычислителя* $\mathcal{V}$ (BMT) с *Оракулом* $\mathcal{O}$, который для любой индивидуальной задачи распознавания с описанием x и ответом «да» — располагает ее решением, доказательством $\mathcal{O}(x)$. Точнее говоря, строчкой $\mathcal{O}(x)$ доказательства[6].

- *Вычислитель* осуществляет проверку некоторых битов строки[7] $\mathcal{O}(x)$, и *принимает* доказательство, если задача x действительно имеет ответ «да».

- Ложное доказательство (если решение задачи «нет») *Вычислитель* принимает с вероятностью не более[8] $1/2$.

- Класс сложности обозначается $\mathrm{PCP}[r(\cdot), q(\cdot)]$, где r и q целочисленные функции, если $\mathcal{V}$ «подбрасывает монету» не более $r(|x|)$ раз и запрашивает из строки $\mathcal{O}(x)$ не более $q(|x|)$ бит.

Предполагается, само собой, повторение процедуры, пока *Вычислитель* не уменьшит вероятность ошибки до желаемого уровня. Вместо конкретных r и q рассматриваются обычно классы функций: *poly* — полиномиально ограниченные функции, *log* — функции

[5] PCP — аббревиатура от ***Probabilistically Checkable Proof System***.

[6] Наличие строки доказательства конкретизирует процедуру, которая в случае IP весьма размыта.

[7] Это второе отличие от класса IP. *Вычислитель* PCP ограничен в своих запросах фрагментами строки доказательства.

[8] Одна вторая принята. Ее можно заменить любой константой $p \in (0, 1)$, ибо повторением процедуры вероятность ошибки понижается сколь угодно.

$f(n) = O(\log n)$, **1** — функции класса $O(1)$. Основной (по значению) сложностный класс PCP[log , **1**] часто обозначается просто как PCP. В этом случае PCP-*теорема* утверждает равенство

$$\boxed{\text{NP} = \text{PCP}.} \tag{8.3}$$

В свое время это произвело фурор. Снежный ком публикаций нарастает до сих пор, не давая разобраться, где наука, а где пиар. Результат в самом деле замечательный, но ожидания остаются неудовлетворенными. Новое, по существу, определение NP-класса проливает дополнительный свет, однако без особых последствий [9]. Восторги же связаны в основном с привходящими обстоятельствами. Доказательство на ста страницах [10], и главное — возможность практического использования PCP-диалогов, которые, дескать, позволяют быстро проверять длинные доказательства. Последнее, разумеется, выдумано журналистами, но очень популярно.

Реальность экономных проверок и так хорошо известна. Опрос трех процентов избирателей очень точно оценивает мнение электората, не говоря о выборочных проверках бракованной продукции или просмотрах математических статей «по диагонали». PCP-ракурс ничего тут не дает. Что же касается самой PCP-системы, то она существенно опирается на волшебного *Оракула*, и поэтому не выпускает равенство (8.3) в практическую плоскость.

В то же время пренебрежительно толковать PCP-*теорему* тоже нельзя. В конце концов, PCP-ниточка — это философские Дарданеллы, неясно что с чем связывающие, но придающие определенное настроение поиска предначертанного пути. Возникают полезные, может быть, ассоциации. Равенство (8.3) допустимо трактовать, например, как полиномиальность глубины экспоненциального по объему (хотя и вытянутого в строчку) решения NP-задачи. В чем-то это перекликается с *теоремой Мошкова* (п. 7.4.1), в которой утверждается полиномиальная глубина *линейного разделяющего дерева*. Там и там для NP-задач всегда есть путь из полиномиального числа шагов, но неясно, можно ли его найти без *Оракула*. Тем

[9] Кое-что с помощью (8.3) все же удалось сделать. Установить, например, неаппроксимируемость некоторых задач оптимизации, невозможность их приближенного решения полиномиальными алгоритмами.

[10] Существенно ужатое теперь *I. Dinur* (2005).

не менее сами полиномиальные пути в NP-классе всегда есть — в этом сила отмеченных теорем, и аргумент в пользу NP = P.

8.4. Рутинная колея

Случайность, как средство достижения определенных выгод, имеет свои резоны. Богатство идей реализации несколько размывает феномен, но суть вертится возле простого трюка, часто называемого *усилением вероятности*. Если вычисление — основанное на использовании случайных параметров и обстоятельств — дает в задаче распознавания верный ответ с вероятностью $p > 0$, то его повторение N раз при неизменности ответа — приводит к ошибке с вероятностью $(1-p)^N$, каковая при больших N может быть сделана сколь угодно малой [11].

Ситуация, конечно, может варьироваться. Например, *алгоритмы типа Лас-Вегас* с некоторой вероятностью могут не давать ответа, но если дают — то ответ обязан быть правильным. *Алгоритмы типа Монте-Карло* могут ошибаться в ответах. При этом делается подразделение на алгоритмы с *односторонней* и *двусторонней* ошибкой — ошибка возможна либо только при ответе «нет», либо в обоих случаях [12]. Вариации исполнения еще шире. В алгоритмах задействуются различные идеи: случайный порядок действий, вероятностная фиксация подзадач, стохастическое ветвление и т. п. При этом в большинстве случаев суть механизмов вполне бесхитростна.

Одной из наиболее притягательных, судя по числу публикаций, является задача 3-ВЫП (3-SAT). Полный перебор здесь требует времени $O(2^n)$. Подключение здравого смысла дает экономию. Вот некоторые вехи на пути совершенствования перебора, в том числе рандомизированного:

$O(1{,}619^n)$ — Monien, Speckenmeyer (1985);
$O(1{,}505^n)$ — Kullmann (1992);
$O(1{,}447^n)$ — Paturi, Pudlak, Saks, Zane (1998);
$O(1{,}333^n)$ — Schoning (1999);

[11] Последовательность ответов, конечно, не обязана быть стабильной, но тогда для подсчета вероятности желаемого «да» или «нет» вместо $(1-p)^N$ надо брать формулу типа $p^k(1-p)^{N-k}$.

[12] Вникать в классификацию особого смысла нет, потому что «ориентироваться на местности» она особо не помогает — проще так сообразить.

$O(1{,}329^n)$ — Baumer, Schuler (2002);
$O(1{,}328^n)$ — Rolf (2003);
$O(1{,}324^n)$ — Iwama, Tamaki (2003).

Роль уменьшения константы вполне прозрачна. Например, $1{,}41^n = 2^{n/2}$, поэтому алгоритм с константой $1{,}41 = \sqrt{2}$ повышает разрядность технически решаемых задач вдвое, скажем, с 2^{50} до 2^{100}, что немало, но горькая пилюля экспоненциальности все равно остается.

• Первые шаги рационализации перебора даются совсем легко. Например, *метод резолюций* при переходе от 2-ВЫП к 3-ВЫП перестает работать, но та же логика позволяет кое-что сэкономить. Концентрируем внимание на какой-нибудь дизъюнкции. Скажем, $D = x_1 + x_2 + x_3$. Полагаем $x_1 = 1$. Если при этом задача целиком не решается, организуем перебор (мысленно) для $x_1 = 0$, $x_2 = 1$. Если и в этом случае задача целиком не решается, то гарантированно: $x_1 = 0$, $x_2 = 0$, $x_3 = 1$. Поэтому

$$t_n \leqslant t_{n-1} + t_{n-2} + t_{n-3} \quad \Rightarrow \quad t_n \leqslant (1{,}84)^n,$$

где t_n — время решения задачи с n переменными, а 1,84 — корень уравнения $t^3 - t^2 - t - 1 = 0$.

• Действуя наугад, выиграть можно больше. В каждой дизъюнкции задачи 3-ВЫП удалим случайно по одному литералу. Останется 2-ВЫП, решаем ее полиномиально. Если решение находится, то оно годится и для 3-ВЫП. Вероятность оставить хотя бы один истинный литерал (в решении 3-ВЫП) для отдельной дизъюнкции равна 2/3, для всех дизъюнкций — $(2/3)^m$, где m — общее количество дизъюнкций. Повторяя трюк N раз, находим решение с вероятностью

$$1 - \left[1 - \left(\frac{2}{3}\right)^m\right]^N = O\left(N\left(\frac{2}{3}\right)^m\right).$$

Поэтому необходимый перебор реально сокращается до $O((3/2)^m)$. Дополнительную экономию дают разные усовершенствования. Например, если удаление литерала у дизъюнкции оставляет два литерала, которые входят еще в дюжину дизъюнкций, то эти два литерала оставляем и в дизъюнкциях из этой дюжины, исключая в каждой — «третий лишний».

Таким образом, опора на «потенциал случайности» облегчает решение практических задач, сокращая перебор с 2^n до величин типа $2^{n/4}$. Но этого все же недостаточно, чтобы выбраться из экспоненциальной грязины. Кроме того, выигрыш нельзя относить полностью на долю «случайности», ибо вероятностные изворотты применяются обычно в комбинации с другими ухищрениями типа *динамического программирования* или элементарного логического

анализа, — что является предметом совсем другого курса. Поэтому в данном контексте вероятностные вычисления дислоцированы на периферии, чтобы только обозначить горизонты.

8.5. Простые числа

Вероятностные алгоритмы для ПРОСТОГО ЧИСЛА привлекают к себе значительное внимание по нескольким причинам. Во-первых, в силу практической эффективности. Во-вторых, причем это многослойный пункт, из-за богатства и красоты теоретических идей, стоящих за кадром. Но природа этих идей арифметическая, поэтому их более уместно обсуждать в томе «Теория чисел».

Что касается самих алгоритмов, то они группируются в основном вокруг *теста Рабина—Миллера* (п. 6.2.2). А вероятность возникает в связи с тем, что прохождение теста составным N зависит от выбора в (6.3) целого $x \in (0, N)$. Число N, не проходящее тест, — составное. Если N проходит тест, то оно простое, но при условии удачного выбора $x \in (0, N)$. При неудачном выборе икса возможна ошибка: составное N проходит тест. Поэтому эффективность алгоритма при случайном выборе $x \in (0, N)$ зависит от того, какой процент «удачных» x имеется на $(0, N)$. Процент, определяемый арифметическими причинами, есть вероятность ошибки, каковая быстро стремится к нулю при повторении теста.

Глава 9

Прагматика и эвристика

Решение практической задачи должно проходить под творческим надзором индивида, способного «в случае чего» переосмыслить постановку и направить усилия в другое русло.

Преодолевать отвращение к конкретике нелегко, но это потом окупается.

Утилитарный срез алгоритмической тематики необозрим, но это предмет другого курса. Здесь кое-что упоминается для обозначения границ и сходящего на нет пейзажа.

9.1. Сетевое программирование как обобщение динамического

Целочисленная функция $\varphi(x) = \varphi(x_1, \ldots, x_n)$ часто может быть представлена в виде композиции функций меньшего числа переменных [1]. Например,

$$f(x_1, \ldots, x_5) = f_0\big[f_1(x_1, x_3),\ f_2(x_2, x_4, x_5)\big]. \qquad (9.1)$$

Структура композиции наглядно изображается в виде графа, типа изображенного на рис. 9.1, который можно дополнять еще одним нижним уровнем иерархии с перечислением аргументов и указанием их вхождения в те или иные функции. Если в композиции аргументы или промежуточные $f_{i\ldots k}$ не входят в подфункции более

[1] Нас интересуют не результаты *Колмогорова—Арнольда* о существовании подходящего представления, а фактическая возможность.

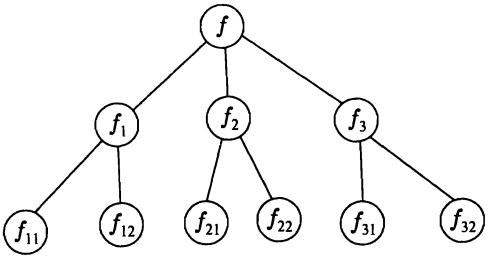


Рис. 9.1

высокого уровня, — граф представления является деревом. Каждая функция f имеет определенную свободу в выборе своего *сетевого представления*.

Если в задаче целочисленного программирования

$$f(x) \to \max, \quad g(x) \leqslant b, \quad x = \{x_1, \ldots, x_n\}, \tag{9.2}$$

f и g имеют одинаковые древовидные *сетевые представления*, то это позволяет задействовать для решения (9.2) механизм типа *динамического программирования*.

Пример

Пусть

$$f(x) = f_0[f_1(x_1, x_2),\ f_2(x_3, x_4)],$$
$$g(x) = g_0[g_1(x_1, x_2),\ g_2(x_3, x_4)].$$

Задача (9.2) в данном случае сводится к

$$f\big(f_1(b_1), f_2(b_2)\big) \to \max, \quad g(b_1, b_2) \leqslant b,$$

где $f_i(b_i)$ — решения задач

$$f_i(\cdot) \to \max, \quad g_i(\cdot) = b_i, \quad i = 1, 2.$$

Таким образом, четырехмерная оптимизация декомпозируется здесь на три задачи размерности два. Механизм практического извлечения выгоды аналогичен стереотипам *динамического программирования*, но данная методология [2)] применима к более широкому

[2)] Идея принадлежит *Буркову*, см.: *Бурков В. Н. и др.* Метод сетевого программирования // Проблемы управления. 2005. № 3. 23–29.

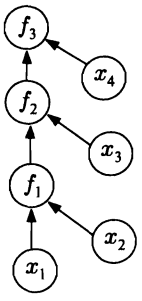


Рис. 9.2

классу ситуаций [3]. Более того, даже там, где динамическое программирование работает, нередко предпочтительны иные схемы иерархического представления функций. Скажем, в аддитивной ситуации,

$$f(x) = \sum_k f_k(x_k),$$

$$g(x) = \sum_k g_k(x_k),$$

возможны представления с любой желаемой древовидной структурой, чем можно пользоваться для сокращения перебора. В частности, в РЮКЗАКЕ сетевое ветвление можно организовать более эффективно, чем обычное вытягивание траекторий под динамическое программирование.

9.2. Ареал жадного алгоритма

Жадный алгоритм решает оптимизационную задачу с помощью последовательной селекции неких элементов, после чего на каждом шаге выбирает «лучший кусок». Иными словами, задача решается близоруко-тактически, с минимальным горизонтом виденья. Результат может быть весьма далек от оптимального, но по соотношению «цена/качество» — бывает приемлем. Вот как действует рецепт в конкретных ситуациях.

• В КОММИВОЯЖЕРЕ на полном реберно-взвешенном графе: «*иди в ближайший из непройденных городов*». Итоговый результат может быть сколь угодно далек от оптимального, но это — в специально приготовленном графе. В ситуациях «общего положения» гамильтонов цикл получается длиннее самого короткого — в два–три раза, а то и в полтора. Во сколько в среднем — зависит от вероятностных предположений о природе графа, каковыми, при желании, можно заниматься

[3] Сетевая оптимизация переходит в динамическое программирование, когда дерево иерархического представления имеет тип, изображенный на рис. 9.2, — что позволяет говорить об оптимизации на траекториях в задачах типа

$$\Theta(x^{n+1}) \to \max, \quad x^{k+1} = \psi(x^k, u^k, k),$$
$$x^0 = 0, \quad u^k \in U, \quad k = 0, 1, \ldots, n.$$

всю жизнь, предполагая то так, то эдак — не без ущерба для репутации в связи со злоупотреблением.

• Вероятностный аналог предыдущего алгоритма основывается на случайном выборе ребра, инцидентного текущей вершине, с вероятностью, обратно пропорциональной весу ребра. Рецепт позволяет избежать ловушечных вариантов, и при наличии времени — ждать пока не повезет, многократно пересчитывая и дополняя поиск *локальными спусками*.

• *В заданном семействе* $\Gamma = \{G_1, \ldots, G_m\}$ *найти минимальное по числу подмножеств* $G_i \subset S$ *покрытие конечного множества* S. Это NP-полная задача ПОКРЫТИЕ[4]. Жадный алгоритм на k-м шаге выбирает то $G_t \in \Gamma$, которое покрывает максимальное число элементов непокрытой части $S_k \subset S$. Если μ — число подмножеств Γ в минимальном покрытии, то

$$|S_k| \leqslant |S|\left(1 - \frac{1}{\mu}\right)^k,$$

что обеспечивает покрытие S, когда наступает событие $|S_k| < 1$, обязательно происходящее в пределах $k \leqslant \mu(1 + \ln|S|)$. (**?**) Таким образом, алгоритм дает решение хуже оптимального не более чем в $(1 + \ln|S|)$ раз.

Вероятностное освобождение алгоритма от жесткого регламента (выбор следующего $G_t \in \Gamma$ с вероятностью, пропорциональной покрываемому числу элементов непокрытой части $S_k \subset S$) может дать выигрыш в среднем[5], но гарантированная оценка пропадет.

9.3. Приближенные алгоритмы

Наивные решения — вполне целесообразный образ действий. Это порождает различные эвристики, которые бывают глупы до святости, но «в среднем» хорошо работают, а вероятностный аппарат при этом служит идеологической подпоркой. Жадный алгоритм — одна из иллюстраций[6]. Есть и другие идеи, которые фактически классификации не поддаются.

В ЗК обходим остовное дерево в соответствии с принципом, изображенным на рис. 9.3. Получаем замкнутый маршрут, проходящий через все вершины. Листья

[4] Как всегда, упоминание NP-полноты подразумевает вариант распознавания: покрытие мощности не более K. Если все $|G_t| = 3$ и $K = |S|/3$, — задача превращается в ТОЧНОЕ ПОКРЫТИЕ 3-МНОЖЕСТВАМИ [7].

[5] А может и не дать. Заключение возможно по вероятностным характеристикам семейства Γ, если таковые имеются.

[6] Разумеется, не в случае матроидов, где алгоритм дает точные решения.

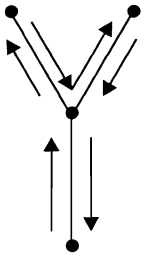


Рис. 9.3

посещаются один раз, внутренние вершины — многократно. Исходные требования нарушены, но не все. Обход вершин соблюдается, но посещение не обязательно по одному разу, да еще каждое ребро контура проходится туда и обратно. Зато длина контура максимум в два раза хуже оптимального решения исходной задачи[7]. Кому-то такой маршрут вполне может подойти. Глупо настаивать на одноразовом посещении городов, если мороки больше и путь длиннее. Даже фиктивные переналадки станка (см. комментарий к ЗК в разделе 1.4) можно использовать, если общее время переналадок сокращается.

Эвристические алгоритмы, оцениваемые в категориях правдоподобия и здравого смысла, получают иногда вполне корректное обоснование. Лучше от этого они не становятся, но выясняются конкретные условия, в которых алгоритмы хорошо работают, например, дают асимптотически точные решения.

Вот простая иллюстрация. Пусть в РЮКЗАКЕ имеется n предметов, v_i — стоимость i-го предмета, w_i — его вес. Надо выбрать группу предметов с максимальной суммарной стоимостью $\sum_i v_i x_i$ при ограниченном суммарном весе $\sum_i w_i x_i \leqslant W$, где x_i могут принимать значение 1 или 0.

Естественный, но не оптимальный, алгоритм решения состоит в том, чтобы упорядочить предметы по удельной стоимости $c_i = v_i/w_i$, а потом поочередно складывать их в трюм корабля, пока соблюдается ограничение по весу. Такой алгоритм будет при $n \to \infty$ асимптотически оптимален, если все $w_i/W \to 0$.

Качество алгоритма часто зависит от типичности решаемой задачи. Ориентация на трудоемкость в худшем случае туманит взор. Симплекс-метод, например, на плохих задачах сводится к перебору, но в среднем полиномиален [22]. Понятно, от расшифровки «в среднем» многое зависит. В подходящих условиях даже пропадает необходимость решения задачи, как в нижеследующем примере.

[7] А может быть, и лучше (по длине) оптимального решения.

Пусть ребро у n-вершинного графа G_n появляется с вероятностью

$$p_n \geqslant \sqrt{2\frac{\ln n}{n}}.$$

Тогда при достаточно больших n почти все графы G_n имеют хотя бы один гамильтонов контур.

9.4. Метод ветвей и границ

Метод ветвей и границ завоевал славу, которая вынуждает его упоминание даже при отсутствии прямой необходимости. К сожалению, общность превращает «нечто» в «ничто». То же самое происходит с *«методом»* при изложении на абстрактном уровне. Слишком общая схема. Реализации друг на друга не похожи. Тем не менее каркас един и достаточно плодотворен, чтобы подступиться к любой задаче, а иногда и решить.

Суть дела примерно такова. Множество S вариантов разбивается приблизительно пополам[8], на S_1, S_2, и в каждой части оценивается значение минимизируемой функции $f(x)$. Наилучшая оценка объявляется *рекордом*. Далее каждое S_j снова делится пополам, возникает новый уровень (ярус) дробления, на котором производятся оценки $f(x)$ на возникающих в результате деления подмножествах S_{j1}, S_{j2}, после чего обновляется значение *рекорда*, затем по тем или иным правилам отбрасываются неперспективные подмножества (*правило отсечения ветвей*) и выбираются $S_{i\ldots k}$ для дальнейшего дробления (*правило ветвления*)[9]. Последующие итерации, как правило, сопровождаются *локальным спуском*[10] из текущей точки, в которой произошло обновление рекорда.

Рекорды, дробление, отсечение ветвей — все это замыкается на специфику задачи. *Локальный спуск*, если используется, заключается в ограниченном поиске лучших вариантов целенаправленным

[8] Или на большее число подмножеств.

[9] Правила ветвления делят на два типа: «просмотр в ширину» — при котором просматриваются все неотброшенные $S_{(\ldots)}$ данного яруса до перехода на следующий уровень дробления, и «просмотр в глубину» — при котором наиболее предпочтительные варианты просматриваются в первую очередь. На практике эти правила обычно смешиваются.

[10] В непрерывном случае, например, градиентным.

перебором. Для этого тем или иным способом вводится близость вариантов (типа метрики), что формализует понятие окрестности. И тогда *спуск* организуется по «ступенькам» окрестностей [11]. Формирование рекорда тоже имеет значительный люфт, от задачи к задаче. Рекордом может быть гарантированная оценка, типа решения задачи ЛП для ЦЛП, или случайное значение $f(x)$, оценка среднего и т. п. В зависимости от исполнения интерпретация «*метода*» бывает разная. Иногда говорят о точном решении задачи, подразумевая доведение перебора до логической точки. Но обычно схема реализуется половинчато и останавливается где-то в середине или даже в начале пути. Достигнутый рекорд предлагается в качестве приближенного решения.

9.5. О задаче ЦЛП

Задача ЦЛП в комбинаторном ряду выделяется своей геометрической наглядностью, в связи с чем к ней удобно примерять различные идеи и алгоритмы. Визуальная очевидность, правда, чревата некоторыми заблуждениями — из-за шаблонов воображения. Новичку часто кажется, например, что решение ЛП после округления дает почти то, что нужно. Рис. 9.4 демонстрирует аномальную в этом отношении ситуацию. Заштрихованный допустимый многогранник ЛП содержит единственную целую точку A, решение же ЛП лежит в B — при любом $c \geqslant 0$. А слегка пошевелив задачу, легко прийти к ситуации, в которой ЦЛП вообще не имеет решения. Либо, наоборот, ЦЛП имеет решение, а ЛП — нет (многогранник неограничен, но целых точек конечное число).

Напомним, что ЛП из ЦЛП

$$\langle c, x\rangle \to \max, \quad Ax \leqslant b, \tag{9.3}$$

получается освобождением (релаксацией) переменных x от необходимости принимать целые значения. Потому, кстати, допустимый многогранник $Ax \leqslant b$ называется *релаксационным*.

[11] В начальной окрестности ищется минимум $f(x)$, как правило, простым перебором. Положение минимума окружается новой окрестностью, ищется новый минимум и т. д.

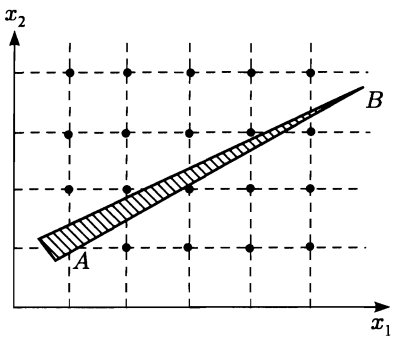


Рис. 9.4

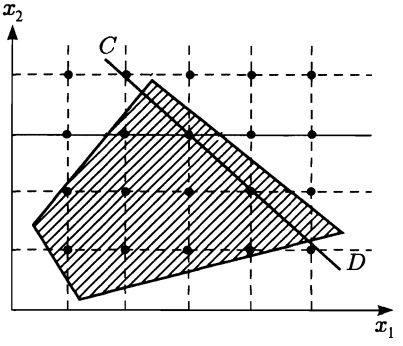


Рис. 9.5

Хотя непрерывная задача ЛП может ничего не давать для ЦЛП, отказываться совсем от релаксации переменных жалко, тем более что здесь можно выжать кое-что дополнительно. Широко используются методы *отсекающих плоскостей Гомори*. Идею поясняет рис. 9.5. Отсекающая «плоскость» CD урезает релаксационный (допустимый) многогранник, но не исключает при этом ни одну целочисленную точку. Решение соответствующей задачи ЛП даст более точную оценку для решения ЦЛП. Соответствующие формульные рецепты описываются в любом учебнике по дискретной оптимизации, см., например, [17].

Глава 10

Квантовые вычисления

Авторитет квантовой механики уже так велик, что «волны вероятности» можно вполне заменить эквивалентной формулировкой «колеблется непонятно что».

В жизни понятно не все, но живем.
В музыке непонятно все, но слушаем.

Квантовая механика, родившаяся как ересь, постепенно растеряла ореол чуда и даже наскучила, ибо ушла острота ощущений. Но теперь шокирующий потенциал микромира опять выходит на авансцену. Вычислительная техника прицеливается шагнуть вглубь, и есть смысл заранее приготовиться.

10.1. В чем идея и каковы препятствия

Двоичное число вводится в регистр обычного компьютера фиксацией каждой ячейки в положении единицы или нуля:

1	0	0	...	1	0	1

.

Запись всех n-разрядных чисел требует 2^n регистров.

Если же в каждую ячейку регистра поместить *элементарную частицу*, способную пребывать в одном из двух возможных состояний, — условно обозначаемых 0 или 1; • или ○; ↑ или ↓; «спин вверх» или «спин вниз», — то **(!)** *в одном регистре будут присутствовать все* 2^n *возможностей* (*чисел*). Потому что квантовая система как бы пребывает во всех возможных состояниях одновременно, и есть лишь вероятности обнаружить ее при измерении в том или ином конкретном состоянии.

Разумеется, это бред с точки зрения здравого смысла, но плоды измерений в конечном счете именно таковы. Поэтому, что там происходит глубоко внутри, никто не знает, хотя кому-то чудится [1], — но квантовая механика оказывается тем арифмометром, который верно предсказывает результаты.

Итак, квантовому компьютеру для записи 2^n чисел требуется всего n ячеек. Следующий трюк еще более удивителен. В детерминированном варианте, преобразуя каждое n-разрядное число x с помощью m-разрядной функции f, имеем табличное задание $f(x)$ при наличии опять-таки экспоненциального числа регистров. В квантовом случае достаточно выполнить один раз преобразование f исходного регистра, где содержатся все n-разрядные x, и функция $f(x)$ готова — для ее записи потребуется $m + n$ ячеек. В этом, собственно, заключается главный фокус, хотя другие рассказчики могут напирать на иные особенности.

Безусловно, когда «доходит до дела», не все идет гладко, как хотелось бы. Функция упакована в регистре, но для определения x, при котором, скажем, $f(x) = 1$, требуется произвести измерение, разрушающее остальную информацию [2]. Есть и другие неприятности. Тем не менее кое-что удается вычислять весьма эффективно. Например, поиск объекта, обладающего неким заданным свойством, в массиве мощности N не может иметь асимптотику вычислений лучше $O(N)$ — разумеется, при «детерминированной философии и соответствующей технологической базе». Квантовый компьютер справляется с задачей за $O(\sqrt{N})$ шагов, — что с классической точки зрения выглядит неправдоподобно [3]. Другой показательный пример — полиномиальный *алгоритм факторизации* (разложения) числа на множители.

Достижения, правда, остаются пока на бумаге, но это естественно. Путь от теории к практике обычно требует времени и усилий.

---

[1] Вплоть до хлопот о переустройстве Вселенной.

[2] Такова природа квантовых систем. Измерение выливается в неконтролируемое макровоздействие, меняющее дальнейшее поведение объекта.

[3] Потому что в худшем случае все N «предметов» надо перебрать. В среднем, может быть, «не все», но все-таки $\sim N$.

Достаточно вспомнить атомную бомбу, лазер, компьютер, мобильную связь. Тем не менее уже факторизовано число $15 = 3 \times 5$ квантово-механическим способом. С точки зрения арифметики результат смехотворный, но дело ведь в принципе. Поначалу многое не ясно. Не помешает ли «вычислениям» неустранимый квантовый шум, не будет ли перегреваться q-*компьютер*, не сглазят ли процесс американцы — будь они неладны, хватит ли точности с учетом *соотношения неопределенности* и т. п. Так или иначе, разложение $15 = 3 \times 5$ поменяло атмосферу, вдохнуло оптимизм. Исследования стали обретать реальные черты [4].

Энтузиазм и умонастроения в научной работе играют немаловажную роль. Идея создания q-компьютера принадлежит *Р. Фейнману* (1982), но она поначалу мало кого воодушевила и едва тлела вплоть до изобретения *алгоритма Шора* (1994). Тогда вдруг физико-математическая общественность всколыхнулась, поверив в перспективу. С тех пор активность преобладает, несмотря на продвижения в час по чайной ложке.

10.2. Основные понятия

Если думать о квантовых компьютерах с точки зрения алгоритмов, понимание квантовой механики необязательно [5]. Но общие приблизительные впечатления полезны, иначе схематичное описание механизмов может показаться не имеющим под собой реальной почвы. Институтских смутных ощущений в какой-то мере достаточно. Поэтому *спин электрона* здесь упоминается как известное понятие [6], хотя и добавляются беглые комментарии с целью кое-что напомнить либо замаскировать.

Итак, пусть в каждую ячейку регистра помещена частица, способная пребывать в одном из двух возможных состояний, которые условно будем обозначать $|0\rangle$ и $|1\rangle$.

---

[4] Примерно как с управляемым термоядом, сказали бы циники.

[5] Не говоря о том, что стремление добиться соответствующего понимания сильно отвлекает и ведет к разочарованию, ибо объект ведет себя противоестественно и разумению не поддается. В то же время физики не хотят разбираться в математике, математики — в физике, а определенная группа ученых преуспевает в обоих направлениях одновременно, что придает описаниям квантовых компьютеров дополнительную загадочность.

[6] Хотя по большому счету никто не знает, что это такое.

Это широко распространенные в квантовой проблематике *дираковские обозначения*. Вектор x гильбертова пространства H обозначается как $|x\rangle$, эрмитово-сопряженный ему — как $\langle x|$, скалярное произведение — $\langle x|y\rangle$. Произведение $|y\rangle\langle x|$ — уже оператор. Так же как $\mathbf{x}^*\mathbf{y}$ — скалярное произведение, $\mathbf{y}\mathbf{x}^*$ — матрица. Обозначения имеют свои плюсы, но в глазах немного рябит.

Состояние частицы «выясняется» только после измерения, а *латентное состояние* представляет собой линейную смесь

$$\lambda_0|0\rangle + \lambda_1|1\rangle, \tag{10.1}$$

где λ_0, λ_1 — *комплексные амплитуды*, квадраты модулей которых являются вероятностями обнаружения частицы при измерении в соответствующем — $|0\rangle$ или $|1\rangle$ — состоянии[7]. Разумеется, $|\lambda_0|^2 + |\lambda_1|^2 = 1$. Состояния $|0\rangle$, $|1\rangle$ образуют *базис*. Базис могут составлять и два других ортогональных состояния вида $\alpha|0\rangle + \beta|1\rangle$.

Существенно для понимания предмета понятие *измерения*. Приблизительно говоря, квантовая механика предполагает, что частица до измерения (до наблюдения) действительно находится во взвешенном состоянии (10.1), т. е. «ни в том, ни в другом». И только акт измерения вносит определенность, после чего частица ведет себя в соответствии с проявленным состоянием. До того — до измерения — динамика частицы неопределенна. Бессмысленно даже говорить о частице. Ее нет. Распространяются *комплексные волны вероятности*, дающие размытый прогноз обнаружения частицы в том или ином положении. Вероятности непрерывны, результаты измерений — дискретны.

В обычную логику сказанное не ложится, а нескончаемые попытки изобрести подходящую интерпретацию принципиально ничего не дают, если не считать обрастания теории дискредитирующими инсинуациями. Квантовая механика многим сильно не нравилась, начиная с *Эйнштейна*. Но как бы «неправильна» она ни была, результаты вычислений получаются верные. Поэтому физики постепенно — по мере вымирания противников — убедили

[7] Сказанное неуклюже и в значительной мере некорректно, но при первоначальном знакомстве с предметом лучше двигаться к аксиоматическому подходу дорогой проб и ошибок.

себя и других, что махинация приемлема. Все притерпелись, и волны вероятности превратились в узаконенный фокус, не дающий покоя философски настроенной части населения.

Единичный вектор (10.1) называют *квантовым битом*, или *q-битом* (*кубитом*). Обычный классический бит, как единица информации, есть по сути двоичный разряд, ячейка, куда помещается 1 или 0. Кубит — это ячейка, куда помещается частица в состоянии $|\psi\rangle = \lambda_0|0\rangle + \lambda_1|1\rangle$.

С увеличением числа ячеек в регистре ситуация кардинально меняется, что связано с возможной взаимозависимостью (*сцепленностью*) состояний [8]. Совокупное состояние двух *независимых* ячеек есть

$$(\lambda_0^1|0\rangle + \lambda_1^1|1\rangle) \otimes (\lambda_0^2|0\rangle + \lambda_1^2|1\rangle) =$$
$$= \lambda_0^1\lambda_0^2|0\rangle \otimes |0\rangle + \lambda_0^1\lambda_1^2|0\rangle \otimes |1\rangle + \lambda_1^1\lambda_0^2|1\rangle \otimes |0\rangle + \lambda_1^1\lambda_1^2|1\rangle \otimes |1\rangle, \qquad (10.2)$$

где $\{|0\rangle \otimes |0\rangle, |0\rangle \otimes |1\rangle, |1\rangle \otimes |0\rangle, |1\rangle \otimes |1\rangle\}$ — так называемый *тензорный базис* пространства состояний двух кубитов (ячеек), каковой удобнее записывать в форме

$$\{|00\rangle, |01\rangle, |10\rangle, |11\rangle\},$$

ибо операция $\otimes$ ничего особенного не обозначает — просто объединяет состояния в упорядоченные пары.

В сухом осадке ничего сложного. Но система из двух *q-битов* может находиться, так считается, в любом состоянии

$$\lambda_{00}|00\rangle + \lambda_{01}|01\rangle + \lambda_{10}|10\rangle + \lambda_{11}|11\rangle, \qquad (10.3)$$

где λ_{ij} — по-прежнему *комплексные амплитуды*, квадраты модулей которых являются вероятностями обнаружения теперь уже пары частиц при измерении в соответствующем $|ij\rangle$-состоянии. «Новость» заключается в том, что набор коэффициентов λ_{ij} не обязательно получается в виде произведений (10.2), — и это отвечает как раз *сцепленным состояниям*.

Сцепленным будет, например, состояние

$$\frac{1}{\sqrt{2}}(|00\rangle + |11\rangle), \qquad (10.4)$$

[8] Терминология колеблется. Говорят также о *спутанных*, *запутанных* или *переплетенных* состояниях.

либо $\frac{1}{\sqrt{2}}\big(|01\rangle+|10\rangle\big)$, не достигаемое с помощью тензорного произведения (10.2). Состояние (10.4) служит эталоном, на котором популярный тандем «Алиса — Боб» занимается *кодированием* под колпаком у «подслушивающей Евы». Фокус в том, что если измерение состояния первой частицы показывает $|0\rangle$ или $|1\rangle$, состояние второй можно не выяснять — она (частица) в том же состоянии[9], на чем, собственно, и строятся неожиданные трюки. Причина заключена в *сцепленности* состояния (10.4), тогда как в ситуации $\frac{1}{\sqrt{2}}\big(|00\rangle+|10\rangle\big)$ частицы в ячейках не сцеплены, — как с точки зрения возможных исходов измерения, так и с точки зрения (10.2):

$$\frac{1}{\sqrt{2}}\big(|00\rangle+|10\rangle\big)=\frac{1}{\sqrt{2}}\big(|0\rangle+|1\rangle\big)\otimes|0\rangle.$$

При обобщении двух-кубитовой системы (10.3) на n-кубитовую — квантовый регистр из n ячеек по аналогии описывается линейной комбинацией

$$|\psi\rangle=\sum_{x=0}^{2^n-1}\lambda_x|x\rangle, \tag{10.5}$$

где состояния $|010\ldots01\rangle$ для краткости обозначаются двоичными числами x,

$$|x\rangle=|010\ldots01\rangle,$$

а λ_x — как и прежде, *комплексные амплитуды*, квадраты модулей которых являются вероятностями обнаружения при измерении теперь уже n частиц в соответствующем $|x\rangle$-состоянии. Поэтому

$$\sum_x|\lambda_x|^2=1.$$

10.2.1. *Вычислительные квантовые процессы представляют собой унитарные преобразования (раздел 10.6) ψ-состояний (10.5).*

Во внутреннюю кухню здесь лучше не вникать. Динамика любой квантовой системы описывается *уравнением Шрёдингера*, решением которого является функция $|\psi(t)\rangle=e^{-\imath Ht}|\psi(0)\rangle$, где t — время, $\boldsymbol{H}$ — гамильтониан системы. В данном

[9] Причем не была, а стала — после измерения первой. В этом заключена как раз фундаментальная особенность измерения квантовой системы, на чем приходится сто раз спотыкаться, пока не выработается привычка.

контексте важно лишь, что $|\psi(t)\rangle$ получается умножением $|\psi(0)\rangle$ на *унитарный оператор* $U(t) = e^{-iHt}$. Иначе говоря, координаты

$$\lambda(t) = \{\lambda_0(t), \ldots, \lambda_N(t)\}$$

разложения $|\psi(t)\rangle$ по базису $\{|0\rangle, \ldots, |N\rangle\}$ получаются из координат вектора

$$\lambda(0) = \{\lambda_0(0), \ldots, \lambda_N(0)\}$$

умножением последнего на матрицу

$$U(t) = e^{-iHt}.$$

При этом, благодаря инвариантности евклидовой длины векторов, все время удовлетворяется условие нормировки

$$\sum_x |\lambda_x(t)|^2 \equiv 1,$$

причем любое *унитарное преобразование* регистра в квантовом компьютере можно организовать технически, — но это уже чистая физика.

Напомним, *унитарная матрица* представляет собой комплексный вариант ортогональной. Матрица U *унитарна*, если $U^*U = I$, где I — единичная, а U^* — не просто транспонированная, а эрмитово сопряженная матрица, т. е. $u^*_{ij} = \overline{u_{ji}}$. Характерные свойства унитарных матриц:

- Столбцы взаимно ортогональны и единичны по длине, аналогично — строки;
- евклидова длина преобразованного вектора $x = Uy$ $(y = U^*x)$ сохраняется, скалярное произведение[10] $\langle x, x\rangle = \langle y, y\rangle$;
- $U^* = U^{-1}$;
- все собственные значения по модулю равны единице, $\lambda_k = e^{i\varphi_k}$.

Особое внимание уделяется унитарным преобразованиям, полезным с точки зрения организации вычислительного процесса. Таковым является, например, однокубитовое *преобразование Уолша—Адамара*[11]

$$H = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}, \tag{10.6}$$

[10] Под скалярным произведением в комплексном пространстве понимается

$$\langle x, y\rangle = \sum_i x_i y_i^*.$$

[11] Другой эталон — *вращение фаз*. В однокубитовом исполнении достигается унитарной матрицей $R = \begin{bmatrix} e^{j\varphi_1} & 0 \\ 0 & e^{j\varphi_2} \end{bmatrix}$.

трансформирующее состояние $|0\rangle$ в суперпозицию состояний

$$\frac{1}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}|1\rangle,$$

а $|1\rangle$ — в суперпозицию

$$\frac{1}{\sqrt{2}}|0\rangle - \frac{1}{\sqrt{2}}|1\rangle.$$

Применяя H к n кубитам по отдельности [12], получаем суперпозицию всевозможных 2^n состояний:

$$\frac{1}{\sqrt{2^n}}(|00\ldots00\rangle + |00\ldots01\rangle + \ldots + |11\ldots11\rangle) = \frac{1}{\sqrt{2^n}}\sum_{x=0}^{2^n-1}|x\rangle. \tag{10.7}$$

Обратное преобразование переводит (10.7) в состояние $|00\ldots0\rangle$, обнуляя коэффициенты при остальных.

(!) Для получения функции $f(x)$ с n-разрядным входом и m-разрядным выходом вводится оператор

$$U_f\colon\ |x, y\rangle \to |x, y \oplus f(x)\rangle, \tag{10.8}$$

где $\oplus$ обозначает поразрядное сложение по модулю 2, т. е.

$$0+0=1+1=0, \quad 1+0=0+1=1.$$

Применение U_f к $|x, 0\rangle$ дает $|x, f(x)\rangle$ в регистре из $(n+m)$ ячеек. А поскольку U_f *линейный унитарный оператор* [13], — суперпозиция всевозможных состояний по x порождает суперпозицию всевозможных состояний $f(x)$:

$$U_f\left(\frac{1}{\sqrt{2^n}}\sum_{x=0}^{2^n-1}|x, 0\rangle\right) = \frac{1}{\sqrt{2^n}}\sum_{x=0}^{2^n-1}U_f(|x, 0\rangle) = \frac{1}{\sqrt{2^n}}\sum_{x=0}^{2^n-1}|x, f(x)\rangle.$$

Тем самым вся информация о функции загоняется в регистр.

---

[12] Это n-кубитовое *преобразование Уолша—Адамара*, задаваемое блочно-диагональной матрицей с блоками (10.6) на диагонали.

[13] Несмотря на возможную нелинейность функции f, ибо U_f действует не на числах, а на разрядах. Унитарность U_f вытекает из очевидной симметричности, а $U_f U_f = I$ следует из $f(x) \oplus f(x) = 0$.

Если $f(x)$ принимает значения $(0, 1)$, то для записи всей информации о функции $f(x)$ в суперпозиции $|x, f(x)\rangle$ требуется всего одна дополнительная ячейка и один электрон, каковой не справился бы с задачей, если бы находился в независимом состоянии типа (10.1). Но ячейки (частицы) оказываются сцеплены, чем обеспечиваются ненулевые амплитуды состояний регистра вида

$$|x, f(x)\rangle = |01\ldots010 f(01\ldots010)\rangle$$

и нулевые амплитуды остальных состояний.

10.3. Вычисление и феномен измерения

Специфика квантовых вычислений обладает рядом неприятных черт. Доступ к результатам (измерение) разрушает состояние системы. Поэтому обычная рецептура программирования не работает. Далее. Несмотря, скажем, на присутствие в регистре всех состояний $|x, f(x)\rangle$ измерение из суперпозиции $\sum_x \lambda_x |x, f(x)\rangle$ фиксирует только одно $|x_0, f(x_0)\rangle$, остальная информация пропадает. Какое состояние $|x_0, f(x_0)\rangle$ будет считано, — заранее сказать нельзя, — процесс случайный, определяется вероятностями $|\lambda_x|^2$.

Впечатление от перечисленного скверное. Инструмент — хоть выбрасывай. Какой толк от присутствия в регистре всей функции, если извлечь можно только одно значение? Выигрыш, тем не менее, для определенных типов задач возможен. Остальное можно выполнять на классической основе. Вероятностный характер счета — тоже не препятствие. *Усиление вероятности* (повторение процедуры счета) позволяет фактически достичь детерминированного результата [14]. Цель же квантовых вычислений (одна из целей) заключена в увеличении модулей комплексных амплитуд $|\lambda_{x_0}|$ тех состояний $|x_0, f(x_0)\rangle$, которые хотелось бы получить в результате считывания.

Такова примерно ожидаемая картина работы квантового компьютера. Вычисление — последовательность унитарных преобра-

[14] С вероятностью ошибки типа 10^{-100}, каковую нельзя исключать и в классических вычислениях.

зований ненаблюдаемого состояния регистра. Измерение (считывание) извлекает усеченную информацию и носит вероятностный характер. Феномен измерения, как свидетельство существования непроходимого барьера между «тем» миром и «этим», заслуживает осмысления и требует привычки, потому что налицо избыток странностей. Например, результатом измерения кубита может быть не только $|0\rangle$ или $|1\rangle$, но и $\frac{1}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}|1\rangle$ или $\frac{1}{\sqrt{2}}|0\rangle - \frac{1}{\sqrt{2}}|1\rangle$, что зависит от того, как организовано измерение, см. раздел 10.8.

10.4. Квантовые вентили

О квантовых вычислениях в главе пишется не для того, чтобы разобраться в деталях, а для того, чтобы легче было стартовать в случае необходимости. Поэтому в какие-то направления нет резона вникать.

Определенное внимание уделяется изучению универсальных наборов простейших унитарных операторов $U_0, \ldots, U_n$, из которых можно образовать с любой наперед заданной точностью любой наперед заданный оператор [15] $U = U_{i_1}, \ldots, U_{i_m}$.

Стандартные однокубитовые элементы:

$$I = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \quad \begin{matrix} |0\rangle \to |0\rangle \\ |1\rangle \to |1\rangle \end{matrix}; \qquad X = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \quad \begin{matrix} |0\rangle \to |1\rangle \\ |1\rangle \to |0\rangle \end{matrix};$$

$$Y = \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix} \quad \begin{matrix} |0\rangle \to -|1\rangle \\ |1\rangle \to |0\rangle \end{matrix}; \qquad Z = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} \quad \begin{matrix} |0\rangle \to |0\rangle \\ |1\rangle \to -|1\rangle \end{matrix}.$$

В качестве строительных блоков рассматриваются и более сложные операторы [16]. Например, двух-кубитовое «*контролируемое нет*», трех-кубитовый *гейт Тоффоли* и др. На всем этом в данном контексте нет смысла задерживаться, ибо связь с вопросами сложности алгоритмов весьма отдаленная. Фокус внимания там другой. Из каких элементов и как «собирать» более сложные операторы, и как реализовать сами элементы на базе, скажем, магнитного ядерного резонанса.

[15] Аналогично набору логических функций с помощью того или иного стандартного комплекта связок — например, конъюнкций, дизъюнкций и отрицаний.

[16] Называемые также *гейтами*, от англ. gate.

10.5. Алгоритм ускоренного поиска

Задача поиска решения уравнения $f(x) = 1$, где функция $f(x)$ принимает значения из $\{0, 1\}$, — охватывает массу практических ситуаций [17], и в неблагоприятных ситуациях требует $O(N)$ шагов, потому что в худшем случае все N значений x надо перебрать. *Алгоритм Гровера* позволяет осуществлять поиск за $O(\sqrt{N})$ шагов. Ускорение с точки зрения труднорешаемости не принципиальное, но оно удивительно и перспективно.

Предположим для простоты, что имеется лишь одно значение x_0, при котором $f(x_0) = 1$, остальные $f(x) = 0$. Считаем также заданным унитарный оператор (10.8), т. е.

$$U_f\colon\ |x, y\rangle \to |x, y \oplus f(x)\rangle,$$

что соответствует программному введению данных о функции $f(x)$ в компьютер. Вводим далее в регистр суперпозицию иксов (10.7), а в последнюю $(n+1)$-ю ячейку — значения $f(x)$ с помощью U_f. Применяя затем *преобразование Уолша—Адамара* к последней ячейке, получаем в $|x, f(x)\rangle = |x\rangle \otimes |f(x)\rangle$ состояния [18] $|x\rangle \otimes (|0\rangle + |1\rangle)$ — там, где $f(x) = 0$, и $|x\rangle \otimes (|0\rangle - |1\rangle)$ — там, где $f(x) = 1$.

Затем снова применяем к регистру оператор U_f. Где было $f(x) = 0$, ничего не меняется, а состояние $|x_0, f(x_0)\rangle$ меняет знак в силу

$$U_f : |x_0\rangle \otimes (|0\rangle - |1\rangle) \to |x_0\rangle \otimes (|0+1\rangle - |1+1\rangle) = -|x_0\rangle \otimes (|0\rangle - |1\rangle).$$

Фактически однократным применением оператора U_f достигнут удивительный результат: выделено решение уравнения $f(x) = 1$ — при соответствующем $x = x_0$ изменен знак, точнее говоря, у комплексной амплитуды λ_{x_0} состояния $|x_0\rangle$ перевернута фаза. Но это заметно лишь в «том» мире. Измерение состояния регистра зафиксирует состояние $|x_0\rangle$ с той же вероятностью $1/N$, с какой и любое другое. Возникает задача увеличения $|\lambda_{x_0}|^2$.

---

[17] Например, поиск *гамильтонова цикла* на множестве пронумерованных, тем или иным способом, путей.

[18] Нормирующий множитель $1/\sqrt{2}$ опускаем.

Увеличения $|\lambda_{x_0}|^2$ за счет других $|\lambda_x|^2$ можно добиться *преобразованием диффузии*

$$\sum_{x=0}^{N-1} \lambda_x |x\rangle \rightarrow \sum_{x=0}^{N-1} (2\Lambda - \lambda_x)|x\rangle, \tag{10.9}$$

где Λ — среднее λ_x. Матрица преобразования (10.9),

$$D = \begin{bmatrix} -1+\frac{2}{N} & \frac{2}{N} & \dots & \frac{2}{N} \\ \frac{2}{N} & -1+\frac{2}{N} & \dots & \frac{2}{N} \\ \dots & \dots & \dots & \dots \\ \frac{2}{N} & \frac{2}{N} & \dots & -1+\frac{2}{N} \end{bmatrix}, \tag{10.10}$$

как легко убедиться, унитарна, а значит, квантово-реализуема [19]. Значение $|\lambda_{x_0}|^2$ при переходе (10.9) растет пока $|\lambda_{x_0}| < \Lambda$, и потому после $\sim \sqrt{N}$ повторений процедуры (10.9) легко достигнуть ощутимых значений $|\lambda_{x_0}|^2$ порядка, скажем, 0,2. Вероятность 0,2 уже достаточно велика, и потому десяток-другой измерений с очень большой вероятностью хотя бы один раз даст состояние $|x_0\rangle$, чего достаточно для решения исходной задачи [20]. Временна́я сложность алгоритма $O(\sqrt{N})$ обусловлена, таким образом, необходимым числом повторений преобразования (10.9).

10.6. Квантовое преобразование Фурье

Дискретное преобразование Фурье (ДПФ)

$$X(k) = \frac{1}{N} \sum_{n=0}^{N-1} x(n) e^{-\frac{2\pi j \cdot kn}{N}} =$$

[19] Причем довольно просто на базе стандартных вентилей.

[20] Поскольку проверка $f(x) = 1$ для любого отдельного x занимает лишь единицу времени.

$$= \frac{1}{N} \sum_{n=0}^{N-1} x(n) \left[\cos \frac{2\pi kn}{N} - j \sin \frac{2\pi kn}{N} \right] \qquad (10.11)$$

преобразует «временну́ю» последовательность

$$x = \{x(0), \ldots, x(N-1)\}$$

в «частотную»

$$X = \{X(0), \ldots, X(N-1)\},$$

и служит весьма общим примером унитарного преобразования с помощью матрицы

$$\Phi = [\varphi_{kn}] = \left[e^{-\frac{2\pi j \cdot kn}{N}} \right].$$

Обратное преобразование (10.6) вычисляется в одно касание,

$$x(k) = \sum_{n=0}^{N-1} X(n) e^{\frac{2\pi j \cdot kn}{N}}.$$

Переход в квантовую тематику осуществляется заменой дискретных отсчетов квантовыми состояниями:

$$|X(k)\rangle = \frac{1}{N} \sum_{n=0}^{N-1} |x(n)\rangle e^{-\frac{2\pi j \cdot kn}{N}}.$$

Эффект от использования ДПФ — тот же, что и в классической области. Комбинирование исходных последовательностей в области Фурье-преобразований иногда оказывается проще, как, например, в ситуации свертки:

$$|z(n)\rangle = \sum_{q=0}^{N-1} |x(q)y(n-q)\rangle \quad \Rightarrow \quad |Z(k)\rangle = |X(k)Y(k)\rangle,$$

$$k = 0, \ldots, N-1.$$

Последовательность $\{|z(n)\rangle\}$ может быть получена обратным преобразованием, которое реализуемо на квантовом компьютере как унитарное, причем в дополнение к классике $\{|z(n)\rangle\}$ может быть функцией сразу всех значений аргумента.

Определение периода. Обычное преобразование Фурье,

$$\widehat{f}(\omega) = \frac{1}{2\pi}\int\limits_{-\infty}^{\infty} f(t)e^{-i\omega t}\,dt \quad \Leftrightarrow \quad f(t) = \int\limits_{-\infty}^{\infty} \widehat{f}(\omega)e^{i\omega t}d\omega,$$

основывается на разложении $f(t)$ в сумму гармонических сигналов, и переводит функции $f(t)$ с периодом τ в функции $\widehat{f}(\omega)$, отличные от нуля лишь при частотах ω, кратных $2\pi/\tau$. Нечто подобное имеет место и для ДПФ. *Дискретное преобразование Фурье функции $x(n)$ периода r — при условии что N кратно r — есть функция $X(k)$, отличная от нуля лишь при значениях аргумента k, кратных N/r*, что легко усматривается из (10.6)[21]. Если же N не кратно r — сказанное выполняется приближенно[22]. Это позволяет измерением состояния квантового регистра, содержащего последовательность $\{|X(k)\rangle\}$, определить значение r — либо сразу, если N кратно r, либо «чуть погодя», повторяя эксперимент несколько раз для уменьшения вероятности ошибки.

10.7. Алгоритм факторизации Шора

Описанная в конце раздела 10.6 возможность полиномиального выяснения периода функции — дает ключ к эффективному разложению числа N на множители. Соответствующий *алгоритм Шора* работает следующим образом. Берется число a, и рассматривается функция

$$f(x) = a^x \pmod N,$$

равная остатку от деления a^x на N. Без ограничения общности можно считать, что a и N взаимно просты, НОД $(a, N) = 1$, —

[21] Обнуление $X(k)$ происходит в результате интерференции. В силу периодичности $x(n)$ все $x(n_0 + rm) \equiv x(n_0)$, и при условии $N = rp$ и k, не кратного N/r, значения

$$X(k) = \frac{1}{N}\sum_{n_0}\sum_{m} x(n_0)\exp\left\{-\frac{2\pi j \cdot k(n_0 + rm)}{rp}\right\} = 0.$$

[22] Потому что основную часть слагаемых уничтожает интерференция, остаются небольшие «хвосты».

иначе, определяя НОД (a, N) с помощью *алгоритма Евклида*[23], можно сразу получить множитель в разложении N. Далее определяется период r функции $f(x)$.

Если $r = N - 1$, то по *малой теореме Ферма* N — гарантированно простое число. Если $r < N - 1$, то N — гарантированно составное[24]. По *теореме Эйлера* $a^{\varphi(N)} \equiv 1 \pmod N$, откуда ясно, что максимально возможный период $r = \varphi(N)$, где *функция Эйлера* $\varphi(N)$ есть порядок мультипликативной группы вычетов $\mathbb{Z}_N^\times$. Значение $\varphi(N)$ всегда четно, но порядок (период r) элемента a может быть меньше $\varphi(N)$, если элемент *не образующий*. Однако на отрезке от 1 до $N - 1$ по крайней мере половина элементов a имеет четный период. Поэтому, повторяя случайный выбор числа a, можно довольно быстро добиться ситуации $r = 2s$.

Если $r = 2s$, то в силу $a^{2s} = 1 (\mathrm{mod}\, N)$, ибо $a^0 = 1 (\mathrm{mod}\, N)$, имеем

$$(a^s)^2 - 1 = 0 \ (\mathrm{mod}\, N) \quad \Rightarrow \quad (a^s - 1)(a^s + 1) = 0 \ (\mathrm{mod}\, N),$$

что означает наличие общего множителя у N и либо $(a^s - 1)$, либо $(a^s + 1)$. Задействуя опять-таки алгоритм Евклида для вычисления НОД $[N, (a^s \pm 1)]$, решаем задачу.

Алгоритм Шора оголяет принципиальную проблему. Время счета с точностью до полиномиальной эквивалентности не меняется при смене технологии вычислений, якобы. Факт легко обоснуется в ситуациях типа «*компьютер* ↔ *машина Тьюринга*», и при благодушном настрое воспринимается как теорема, хотя соответствующего уровня, разумеется, никогда не достигнет. В данном случае, похоже, возникает исключение фундаментального характера. Квантовый прибор дает эффективное решение там, где обычная технология терпит фиаско, по-видимому.

10.8. Антипод здравого смысла

О квантовой механике выгоднее молчать, ибо что ни скажи — все мимо. Оно, конечно, везде так — и в жизни, и в религии. И есть

[23] Алгоритм Евклида действует по схеме:
136 : 51 = 2 (остаток 34);
51 : 34 = 1 (остаток 17);
34 : 17 = 2 (остаток 0). ⇒ НОД (136, 51) = 17.

[24] Здесь полезно вдуматься, почему определение периода сразу и безошибочно решает обе задачи ПЧ и СЧ, — и тем самым эффективнее *теста Ферма* и ему подобных.

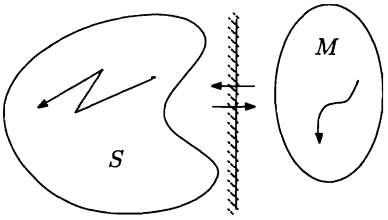


Рис. 10.1

лишь одна возможность двигаться — дорогой лжи. Сфальшивишь раз-другой — кое-что проясняется. Вот так галсами, правдами и неправдами... А иного-то и пути нет.

Тут не в эпатаже дело. Между системой S и моделью M (рис. 10.1) всегда пролегает стена, за которой система, вообще говоря, «не видна». Наблюдаемы всплески — измерения. Все понятия — время, пространство, дух, материя, информация — принадлежат моделям. Приписывание их системе рождает иллюзию понимания, но ведет в трясину противоречий, потому что M — будь то физическая теория или религиозный трактат — моделирует доступную наблюдению оболочку, хотя и претендует на большее.

Квантовая механика неуязвима в стерильном варианте, как чисто математическая модель. Неприятности появляются вслед за интерпретациями[25], конструирующими объяснения из того, что в голове, а не из того, что в S. Некоторые выдуманные понятия стали, правда, ассоциироваться с реальностью, но при изучении микромира такой фокус не проходит. Слишком велика пропасть между M и S.

Здесь почти все не укладывается в привычную картину мира. Дифрагирующий электрон проходит одновременно через две дифракционные щели. Один электрон через две щели! Нонсенс. Поэтому формулировать приходится аккуратнее. Поначалу, мол, нет никакого электрона, а есть дифрагирующий волновой процесс (колеблется непонятно что), рождающий при столкновении с экраном частицу, «географическое» положение которой определяется вероятностью пропорциональной размаху колебания.

[25] Вопросы «почему» не имеют ответов, но аудитория требует — и книги говорят больше, чем следовало бы. Комментарии превращаются потом в догмы, переворачивая теорию с ног на голову.

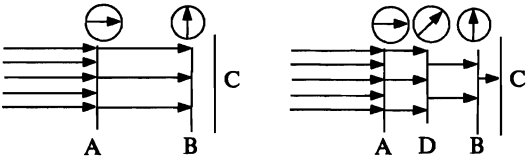


Рис. 10.2

Тут вранья еще больше. Тяга к аккуратности засасывает в воронку, каковую сколько ни провозглашай теорией — эффекта не добьешься. В миниатюре это отражает всю пропасть между интерпретациями и рецептами. Само применение математического аппарата здесь не вызывает проблем. Дифракционная картина легко и точно считается. Иначе говоря, вопрос «как» — имеет четкое решение. Однако «почему» — совершенно неясно. Но не все готовы смириться с тайной. Хочется, чтобы хоть кто-то понимал — и тогда выдумки насчет *корпускулярно-волновой двойственности* частично успокаивают, а правда раздражает.

Другой пример. На пути потока случайно поляризованных фотонов (рис. 10.2) ставится «горизонтальный» фильтр A ($\rightarrow$), не пропускающий вертикальную составляющую поляризации, и «вертикальный» фильтр B ($\uparrow$), не пропускающий горизонтальную составляющую поляризации. На экран C свет не проходит. Добавление фильтра D ($\nearrow$), не пропускающего составляющую $\searrow$, странным образом приводит к появлению света на экране C, что по классическим воззрениям невозможно — дополнительный фильтр увеличивает прозрачность системы.

В данной ситуации удобно задуматься об измерении, или о взаимодействии микро- и макромира. Понять суть дела, разумеется, невозможно[26], но само правило просто, хотя и странно. Всякое измерение кубита (двумерной системы) характеризуется *ортогональным базисом*. Скажем, фотон в состоянии $\alpha|\rightarrow\rangle + \beta|\uparrow\rangle$ при измерении в базисе $\{|\rightarrow\rangle, |\uparrow\rangle\}$ оказывается в одном из состояний $|\rightarrow\rangle$, $|\uparrow\rangle$, — и далее *остается* (*обязан оставаться*) *в этом состоянии*[27]. Фильтр A проходят лишь те фотоны, которые при взаимодействии с A оказываются в состоянии $|\rightarrow\rangle$, но они не проходят фильтр B, способный пропускать только фотоны в состоянии $|\uparrow\rangle$.

[26] Если не передвигать загадку в другое место.

[27] Квантовые биты меняются при необратимом взаимодействии. Поляризационный фильтр как бы измеряет плоскость поляризации фотона, но вместо «истинного» $\alpha|\rightarrow\rangle + \beta|\uparrow\rangle$ может дать один из двух результатов: либо $|\psi\rangle$ — фотон проходит фильтр и попадает

Что касается фильтра D, то он работает в базисе

$$\{|\nearrow\rangle,\ |\searrow\rangle\} = \left\{ \frac{1}{\sqrt{2}}(|\rightarrow\rangle + |\uparrow\rangle),\ \frac{1}{\sqrt{2}}(|\rightarrow\rangle - |\uparrow\rangle) \right\},$$

и пропускает фотоны в состоянии $|\nearrow\rangle$, которое имеет ненулевую проекцию на вектор $|\uparrow\rangle$, — поэтому фильтр B их частично пропускает. Все это настоящего физического объяснения не имеет. Чисто механическое правило, аксиома.

10.9. ЭПР-парадокс и скрытые параметры

Среди квантово-механических «нелепостей» широкую огласку получил ЭПР-*парадокс Эйнштейна—Подольского—Розена* (1935), суть которого в том, что в *сцепленных состояниях* типа (10.4) элементарные частицы «обязаны» согласовывать поведение друг с другом, как бы мгновенно обмениваясь сигналами — в нарушение *запрета сверхсветовых скоростей*, — что *Эйнштейну*, конечно, не могло понравиться. Точнее говоря, если измерение состояния первой частицы в ситуации (10.4) дает $|0\rangle$, то вторая в тот же момент обязана оказаться в том же состоянии.

Подобная «дикость» получила экспериментальные подтверждения, на чем квантовая механика заработала дополнительные очки. Поэтому копья ломаются лишь по поводу интерпретаций. Красивое решение предложил *Бом*: парадокса нет, поскольку нет различных частиц. Частицы, мол, не отдельные объекты, а лишь разные лица чего-то единого. Поэтому никакого обмена сигналами вообще нет, как нет его между отражениями одного предмета в разных зеркалах. Для пущей убедительности *Бом* «нарисовал» картинку. Аквариум с рыбкой проецируется на два взаимно перпендикулярных экрана. Наблюдатель, которому видны только экраны, уверен, что есть две рыбки. Потом он замечает, что их поведение согласованно, и делает вывод о наличии мгновенного обмена сигналами. Трюк с аквариумом играет гипнотизирующую роль, но идея уперлась в глухую стену.

в состояние, совпадающее с плоскостью поляризации фильтра $|\psi\rangle$, либо $|\psi^{\perp}\rangle$ — фотон не проходит фильтр.

Что касается самого ЭПР-эффекта, то отсутствие «классического объяснения» не мешает его использованию при *плотном кодировании* и *телепортации* (см. раздел 10.11). Несколько слов необходимо сказать об изначальном, довольно поспешном объяснении ЭПР-парадокса на основе *теории скрытых параметров*, — потому что круги по воде до сих пор расходятся. Дескать, поведение квантовых объектов однозначно характеризуется скрытыми параметрами, детерминированно определяющими результаты измерений. Отсюда, мол, вероятности и прочая эквилибристика. Отсюда и ЭПР-эффект, раз вероятностный характер измерения всего лишь от незнания (параметров).

Упрощенно говоря, в случае $\frac{1}{\sqrt{2}}(|00\rangle + |11\rangle)$ обе частицы находятся либо в состоянии $|0\rangle$, либо $|1\rangle$. Просто никто не знает — в каком. Поэтому для объяснения корреляции измерений нет никакой необходимости в передаче сигналов между частицами.

Идея навскидку привлекательная, но она мгновенно рассыпается даже при самом поверхностном анализе. Достаточно рассмотреть измерение состояния пары в другом базисе. Реальный сюжет, правда, оказался достаточно тяжеловесен, вплоть до теоремы о невозможности объяснить квантовые явления скрытыми параметрами, звучащей в кулуарах с оттенком значительной глубины. Но такой результат очевиден с самого начала.

Дело в том, что разговоры о квантовых волнах вероятности имеют чисто иносказательный смысл, и метафору не надо воспринимать прямолинейно. Колеблются не вероятности, а действительно «непонятно что». Последнее абсолютно точно отражает положение дел [28]. Вероятности появляются на стадии измерения, точнее говоря, при необратимых изменениях квантовых объектов [29], сопровождающих приведение в контакт микро- и макромира. А *уравнение*

[28] Кому-то из физиков может не понравиться несколько бесцеремонное обращение со «священной коровой». Но тут надо иметь в виду, что носиться с предметом исследования, как с писаной торбой, отпугивая посторонних, вредно и для предмета, и для обслуживающего персонала.

[29] При обратимых — квантовый объект как бы ненаблюдаем.

Шрёдингера описывает колебание некой *фикции*, имеющей длину и фазу. Длина в необратимых взаимодействиях выступает в роли вероятности. Но *фикция* в целом вероятностной интерпретации не поддается, потому что вероятностные процессы в классическом понимании обязаны подчиняться определенной аксиоматике. Там же, где «вероятности» имеют фазу и подвержены интерференции, — возникает сумасшедший дом [30], т. е. квантовая динамика.

Разумеется, идея о скрытых параметрах трещит по швам при попытке толкования квантовых процессов как детерминированных с неизвестными характеристиками. Но с другой точки зрения состояние квантовой системы точно характеризуется набором (скрытых) *комплексных амплитуд* $\{\lambda_x\}$, каковой детерминированно эволюционирует «там». Другое дело, что комплект $\{\lambda_x\}$ принципиально ненаблюдаем и *неинтерпретируем* «здесь». Иными словами, квантовый объект представляет собой как бы потустороннюю субстанцию, которая не укладывается в привычные понятия и не помещается без искажения в наблюдаемый пласт бытия.

10.10. О перетягивании каната

В главе настойчиво подчеркивается, что интерпретационные компоненты квантовой механики, как правило, лишь затемняют суть дела. Имея психологический спрос, они, конечно, неизбежны. Более того, ложные трактовки могут быть полезны, ибо инициируют исследовательскую активность и будят воображение. Однако для определенной части аудитории это, безусловно, яд.

В то же время крен в стерильно-математическую нишу еще хуже. Тенденция математизировать знание, изгоняя из него реальные явления, уже давно подтачивает науку. Что есть механика, как не теория дифференциальных уравнений? Математическая среда так думает почти в открытую. И спорить с тем, что «в каждом знании столько истины, сколько математики», — весьма трудно. Но диалектика физико-математического знания такова, что каждая половина без своего дополнения перестает быть жизнеспособной.

[30] Что вполне ясно, и может быть продемонстрировано на примерах, чем физики и развлекались.

10.11. Комментарии и дополнения

• Беда квантовой механики, или счастье, — не в пороках исследовательской мысли, а в самом объекте изучения. На краю того среза Вселенной, в котором мы обитаем, привычные понятия теряют смысл. И никакие усилия классического толкования запредельных явлений не изменят устройства мироздания. Загадку, разумеется, можно гонять по кругу, переходя от представлений *Шрёдингера* к картине *Гейзенберга* или *Дирака*, меняя волны *де Бройля* на собственные значения гамильтонианов, но аппарат все равно останется фокусом, потому что растет «из и для» загадки.

• «Заоблачность» микромира проявляется в разных аспектах. Одни из них способствуют вычислениям, другие — препятствуют. К последним относится невозможность копирования состояния — обычно говорят о *невозможности клонирования*, что подключает генетическую аранжировку. Если без пиар-отступлений, то речь идет о невозможности обеспечения равенства $U_c : |\psi, 0\rangle \to |\psi, \psi\rangle$ для любого ψ — с помощью *унитарного оператора* U_c, который в случае существования можно было бы назвать клонирующим.

• Неудобство квантовых процессов еще можно охарактеризовать как невозможность получить информацию о коэффициентах в $\alpha|0\rangle + \beta|1\rangle$ с помощью измерения. Ибо последнее дает либо $|0\rangle$, либо $|1\rangle$, но это ничего не говорит о значениях α, β (разве что $\alpha \neq 0$ или $\beta \neq 0$). Поэтому возникает впечатление, что измерение также не может помочь копированию, тем более что оно разрушает состояние измеренного объекта. На этом фоне удивительно явление *телепортации*. Оказывается, проделывая определенные манипуляции над частицей в состоянии $\alpha|0\rangle + \beta|1\rangle$, находящейся в пункте А, и передавая обычную информацию по классическому каналу связи в пункт В, можно перевести частицу в В в то же самое состояние $\alpha|0\rangle + \beta|1\rangle$. Противоречия с невозможностью клонирования не возникает, потому что состояние частицы в А оказывается разрушенным в результате проводимых «манипуляций». Имеются экспериментальные подтверждения феномена, а описание процедуры есть в любой книжке о квантовых компьютерах.

• Квантовая технология в большинстве арифметических задач заведомо не выигрышна. Пока имеется не так много задач, где найденные алгоритмы дают выгоду. Эффект в *алгоритме Гровера*, правда, не так велик — зато *поиск* в любых вычислениях составляет весомую часть, и суммарная польза может быть существенна. Так или иначе, но даже дополнение обычного компьютера двумя квантовыми блоками *Шора* и *Гровера* — дало бы существенный результат.

• Технологическая база современной вычислительной техники с принципиальной точки зрения не требует совершенствования. Все, что может быть описано, физически или даже метафизически промоделировано, — может быть посчитано в соответствии с *тезисом Тьюринга* [6, т. 6]. Но кое-что считается

не так просто и не так быстро, как хотелось бы. Поэтому идеи дополнения компьютера различными специализированными блоками заслуживают внимания[31]. В соответствующие рамки ложатся различные поползновения: аналоговые схемы, нейронные сети, генетические алгоритмы, ДНК-компьютеры и другие естественные и аномальные выбросы. Особое положение в этом ряду занимают квантовые вычислительные блоки, которые в ряде случаев могли бы рассматриваться как реализация гипотетических оракулов. Понятие *оракула* как устройства, которое в состоянии решать за «один шаг» некоторые типы задач, в том числе неразрешимые (например, об *останове машины*), имеет потусторонний аромат и не вполне стыкуется с конструктивными подходами в теории алгоритмов. Оракул, мгновенно решающий ЗК в предположении $NP \neq P$, также выглядит «выходцем с того света», но квантовые компьютеры могут кардинально изменить ситуацию. Что касается конкретно ЗК — вопрос неясен, но труднорешаемая задача факторизации числа квантовым блоком решается эффективно, и это похоже на прорыв философского значения по части оракулов.

[31] Необходимость ждать результатов 10^{100} лет заслоняет многие неприятности, но думать приходится и в других направлениях. Некоторые задачи вообще плохо решаются, типа многоэкстремальной оптимизации. И тогда приходится изобретать что-либо эвристически целесообразное. Генетические алгоритмы, например, каковые можно реализовать на обычной электронной базе, но не грех подыскать и что-либо более подходящее.

Глава 11

Сводка основных определений и результатов

11.1. Список NP-полных задач

✓ **КОММИВОЯЖЕР (ЗК)**

Дан граф с n *вершинами и длины ребер* r_{ij}. *Требуется найти замкнутый путь минимальной длины, проходящий через все вершины по одному разу.* Это классический оптимизационный вариант *задачи коммивояжера* (**ЗК**). Вариант «распознавания»: *существует ли путь, проходящий через все вершины по одному разу, длина которого,* $r_{i_1 i_2} + r_{i_2 i_3} + \ldots + r_{i_n i_1}$, *не превосходит* r? Задача остается NP-полной в частном случае проверки существования самого **ГАМИЛЬТОНОВА ЦИКЛА** — пути, проходящего через все вершины по одному разу.

✓ **ИЗОМОРФИЗМ ПОДГРАФУ**

Даны два графа G_1, G_2. *Верно ли, что* G_1 *содержит подграф, изоморфный*[1] G_2? При совпадении числа вершин и ребер у G_1 и G_2 задача переходит в **ИЗОМОРФИЗМ ГРАФОВ** — изоморфны ли G_1 и G_2? — но NP-полнота этой задачи под вопросом.

✓ **КЛИКА**

Существует ли в графе клика, т. е. полный[2] *подграф, с числом вершин не менее* k? Это вариант «задачи распознавания». В случае «оптимизации» речь идет о нахождении *клики максимального размера*. Размер максимальной клики называют *плотностью графа*. Иначе говоря, плотность графа — это максимальное число вершин, попарно соединенных ребрами.

✓ **ВЕРШИННОЕ ПОКРЫТИЕ**

Существует ли в графе $G(V, E)$ *вершинное покрытие из* k *вершин*? Подмножество $V' \subset V$ называют *вершинным покрытием* графа $G(V, E)$, если каждое ребро G инцидентно некоторой вершине из V'.

---

[1] То есть совпадающий с G_2 при подходящей перенумерации вершин. *Подграфом* графа G называют любой граф $G' \subset G$, вершины и ребра которого являются вершинами и ребрами графа G.

[2] Граф называется *полным*, если содержит все возможные ребра. *Кликой* часто называют также полный подграф максимального размера.

КЛИКА, ВЕРШИННОЕ ПОКРЫТИЕ и еще АНТИКЛИКА (подмножество вершин $V' \subset V$, не соединенных в $G(V, E)$ ни одним ребром), — образуют единый комплект в следующем смысле.

Теорема 1.4.1. *Граф G содержит клику размера k в том случае, если дополнение*[3] $\overline{G}$ *имеет антиклику того же размера, либо $\overline{G}$ обладает вершинным покрытием размера $|V| - k$.*

✓ **РЮКЗАК**

Имеется n предметов, v_i — стоимость i-го, w_i — его вес. Надо выбрать группу предметов с максимальной суммарной стоимостью при ограниченном суммарном весе, т. е. решить задачу

$$\sum_i v_i x_i \to \max, \quad \sum_i w_i x_i \leqslant W,$$

где x_i может принимать значение 1 — «брать», 0 — «не брать».

Задача встречается в различных вариациях. В варианте распознавания остаются два неравенства, которые обычно сводят к одному, приравнивая стоимости весам. Неравенство часто заменяют равенством

$$\sum_i w_i x_i = W,$$

требуя его разрешимости либо в целочисленных переменных, либо в (0–1)-переменных. В первом случае задачу называют **ЦЕЛОЧИСЛЕННЫЙ РЮКЗАК**, во втором — **(0, 1)-РЮКЗАК**. Наконец, частный случай $W = \frac{1}{2}\sum_i w_i$ называют задачей **РАЗБИЕНИЕ**, в силу эквивалентной формулировки: можно ли выделить такое подмножество индексов J, что

$$\sum_{i \in J} w_i = \sum_{i \notin J} w_i.$$

Несмотря на видимую простоту, все «РЮКЗАКИ» NP-*полны*.

✓ **ЦЕЛОЧИСЛЕННОЕ ЛИНЕЙНОЕ ПРОГРАММИРОВАНИЕ** (кратко **ЦЛП**)[4]

$$\sum_{i=1}^{n} v_i x_i \to \max, \quad \sum_i w_{ij} x_i \leqslant W_j, \quad j = 1, \ldots, m,$$

[3] Дополнение $\overline{G}$ имеет то же множество вершин что и G, и содержит те ребра, которых не содержит G.

[4] Вариант распознавания: разрешимость системы неравенств

$$\sum_{i=1}^{n} v_i x_i \geqslant K, \quad \sum_i w_{ij} x_i \leqslant W_j, \quad j = 1, \ldots, m,$$

при заданном K.

где искомые переменные x_i *и коэффициенты* v, w, W — *принимают целочисленные значения*. Задача NP-полна даже в случае (0, 1)-переменных. Если требование целочисленности x_i снимается, ЦЛП трансформируется в обычную задачу **ЛП** — **ЛИНЕЙНОЕ ПРОГРАММИРОВАНИЕ**.

✓ В схему **ЛП** укладывается **ТРАНСПОРТНАЯ ЗАДАЧА** $\in$ P

$$\sum_{i,j} c_{ij}x_{ij} \to \min, \quad \sum_{j} x_{ij} \leqslant a_i, \quad \sum_{i} x_{ij} \geqslant b_j,$$

где переменные x_{ij} при желании могут быть выстроены в цепочку.

✓ Задача **ВЫПОЛНИМОСТЬ** (коротко **ВЫП**, **SAT**)

Дана n*-местная логическая функция* $\varphi(x_1, \ldots, x_n)$ *в конъюнктивной нормальной форме, т. е. в виде произведения сумм (дизъюнкций) переменных или их отрицаний. Например,*

$$\varphi(x_1, \ldots, x_n) = (\bar{x}_1 + x_5 + x_8) \cdot (\bar{x}_2 + \bar{x}_4) \cdot (x_2 + \bar{x}_7 + \bar{x}_8).$$

Существует ли набор значений переменных $\{x_1, \ldots, x_n\}$, *при котором* $\varphi(x)$ *выполнима, т. е.* $\varphi(x_1, \ldots, x_n) = 1$?

То же самое часто излагают иначе. Имеется m дизъюнкций

$$D_1(x), \ldots, D_m(x).$$

Требуется установить существование набора $x = \{x_1, \ldots, x_n\}$, при котором истинны все $D_j(x)$, т. е. $\forall\, j : D_j(x) = 1$. От такой интерпретации удобно отталкиваться при формулировке задачи **max-ВЫП**: *найти* $x = \{x_1, \ldots, x_n\}$, *при котором истинно максимальное число дизъюнкций* $D_j(x)$.

✓ Задача **ИСЧЕРПАНИЕ ВЕРШИН КУБА (ИВК)** $\in$ co-NP

Дана совокупность строчек длины n *вида*

$$*\,1\,*\,0\,*\,*\,*\,*\,1,$$

где звездочку можно заменять на любую цифру 1 *или* 0. *Покрывают ли они все вершины куба* $[0, 1]^n$?

Дополнение ИВК, сводящееся к выяснению существования на кубе непокрытой вершины, — является эквивалентом задачи ВЫПОЛНИМОСТЬ.

✓ Задача **МАШИНА ТЬЮРИНГА**

Существует ли такое v, *что машина* M *на слове* $x = \{v, w\}$ *заканчивает вычисление с результатом «да» за время меньшее* $p(|x|)$? Задача в точности охватывает весь класс NP, в силу того что машина Тьюринга является универсальным вычислительным инструментом.

✓ **РАСКРАСКА ГРАФА В 3 ЦВЕТА**

Можно ли все вершины графа раскрасить в 3 *цвета так, чтобы смежные вершины были окрашены в разные цвета*? Задача NP-полна и остается полной для графов, все вершины которых имеют степень не более 4, а также для *планарных графов.*

11.2. Алгоритмы и вычислимость

✓ Целочисленную функцию $f(n)$ целочисленного аргумента называют *вычислимой* при условии существования *алгоритма*, вычисляющего значения $f(n)$, если они определены.

✓ *Процедура вычислений* отождествляется с программой работы компьютера, представляющей собой запись алгоритма на некотором языке в некотором алфавите $\mathbb{A}$,

$$P = a_1 a_2 \ldots a_N, \quad \text{все} \quad a_j \in \mathbb{A}.$$

✓ Существование невычислимых функций демонстрирует, как далеко может заходить труднорешаемость. Подробная информация имеется в [6, т. 6].

✓ Множество считается *перечислимым*, если существует дающая результат при любом n процедура порождения его элементов. Множество *разрешимо*, если существует эффективная (дающая результат при любом n) процедура для выяснения принадлежности любого n этому множеству. Чаще используется эквивалентное определение: *множество X перечислимо, если представляет собой область значений либо область определения вычислимой функции.*

✓ 2.2.2 **Теорема Поста.** *Для разрешимости множества X необходимо и достаточно, чтобы X и его дополнение $\overline{X}$ были перечислимы.*

Для вычислимости $f(x)$ необходима и достаточна перечислимость графика, т. е. множества пар $\{x, f(x)\}$.

✓ 2.2.3 **Теорема.** *Существует перечислимое, но неразрешимое множество положительных целых чисел.*

✓ 2.3.1 **Теорема.** *Какова бы ни была непротиворечивая теория, содержащая арифметику, существует не имеющий положительных корней полином $Q(x_1, \ldots, x_k)$, отсутствие у которого целых положительных корней недоказуемо.*

✓ 2.3.2 **Теорема.** *Множество $\mathcal{P}$ неразрешимых полиномов $P(x)$ — неперечислимо (тем более, неразрешимо).*

✓ 2.3.5 **Теорема Райса.** *Любое нетривиальное свойство вычислимых функций алгоритмически неразрешимо.*

✓ *Машина Тьюринга* характеризуется тремя функциями F, G и H (2.4), определяющими динамику «вычислений», которые показывают: F — «что в ячейку писать», G — «в какое состояние переходить», H — «куда сдвигаться, влево или вправо». При этом среди внутренних состояний машины $q_1, \ldots, q_n$ выделяется начальное — q_1, в котором машина приступает к работе, и конечное — q_n, попадая в которое, машина останавливается.

✓ *Полиномиальное время работы алгоритма* предполагает требуемое число операций $f(x)$ для завершения работы программы равным *О-большому* от x^k, где x — длина описания задачи, т. е.

$$f(x) = O(x^k) \quad \textit{при некотором} \quad k > 0.$$

✓ *Длина описания задачи* — еще говорят «*длина входа*» — также определяется с точностью до умножения на положительную константу, и представляет собой «величину», пропорциональную числу символов в записи условия задачи, — при той или иной кодировке данных. *Длиной описания задачи* можно считать *количество информации* (число *битов*, т. е. двоичных знаков алфавита $\{0, 1\}$), необходимое для описания задачи.

11.3. Проблема $\mathrm{P} \stackrel{?}{=} \mathrm{NP}$

✓ 1.2.2 **Определение.** *Совокупность задач распознавания, которые могут быть решены некоторым полиномиальным алгоритмом, называется* ***классом P.***

✓ 1.2.3 **Класс NP** *определяется как совокупность* ***полиномиально проверяемых*** *задач распознавания, в которых, если решением является ответ «да», то существует «слово» x полиномиальной длины и полиномиальный от x алгоритм, дающий ответ «да».*

Иначе говоря,

1.2.3′ *Язык $L \in$ NP, если существует полиномиально вычислимый предикат R_L, такой что слово (задача) $z \in L$, если*

$$\exists\, x : (z, x) \in R_L.$$

✓ **NP-полнота.** Среди NP-*задач* есть в некотором роде универсальные, как говорят, — NP-*полные*, каковые *полиномиально эквивалентны* друг другу. По определению задача NP-*полна* (NP-*complete*), если к ней полиномиально сводится любая другая NP-задача. Поэтому полиномиальное решение любой NP-*полной задачи* полиномиально решает все остальные NP-*задачи*.

✓ 3.1.1 **Теорема.** *Множество полиномиальных алгоритмов распознавания перечислимо, но неразрешимо.*

✓ 3.3.1 **Определение.** *Функция* f_1 *полиномиально сводится к* f_2, — *пишут* $f_1 \propto f_2$, — *если существует полиномиально вычислимая функция* f, *такая что* $f_1(x) = f_2(f(x))$.

✓ NP-*полной* (*универсальной переборной*) является, например, задача МАШИНА ТЬЮРИНГА: *существует ли такое* v, *что машина* M *на слове* $x = \{v, w\}$ *заканчивает вычисление с результатом «да» за время меньшее* $p(|x|)$? Задача в точности охватывает весь класс NP, в силу того что машина Тьюринга является универсальным вычислительным инструментом. Таким образом, факт существования универсальной переборной задачи становится очевидным, что решает философскую часть проблемы.

✓ *Теорема Кука* устанавливает NP-полноту ВЫПОЛНИМОСТИ, трансформируя тем самым проблематику в область задач, которые можно «пощупать».

✓ NP-полными являются многие популярные задачи. Понимание этого факта было главным достижением *Кука—Карпа—Левина*, в результате которого вскрылся обманчивый характер большого разнообразия прикладной комбинаторики. ВЫПОЛНИМОСТЬ, ЗК, ЦЛП, ИЗОМОРФИЗМ ПОДГРАФУ, КЛИКА, ГАМИЛЬТОНОВ ЦИКЛ — все это оказалось полиномиально эквивалентными лицами универсальной переборной задачи. Список NP-полных задач сегодня насчитывает сотни наименований.

11.4. Вокруг «P или NP»

✓ 1.2.2 Если задача T состоит в выяснении того, существует ли элемент $x \in X$, обладающий свойством $\mathcal{B}$, то *дополнение задачи* T^c заключается в установлении того, что ни один элемент $x \in X$ не обладает свойством $\mathcal{B}$. Совокупность дополнений NP-задач образует *класс* co-NP.

✓ Неизвестно, совпадает ли co-NP с классом NP. Возможное равенство co-NP = NP не исключает NP $\neq$ P. Нерешенной проблемой является также справедливость равенства co-NP $\bigcap$ NP = P.

✓ 4.1.1 **Теорема.** *Если существует такая NP-полная задача* T, *что* $T^c \in NP$, *то* NP = co-NP.

✓ В постановках некоторых комбинаторных задач фигурируют числа, благодаря разбросу которых задачи оказываются NP-полны, а при ограничениях диапазона — переходят из NP в P. Соответствующие алгоритмы называют *псевдополиномиальными*. Задачи — не имеющие псевдополиномиальных алгоритмов решения — называют *сильно* NP-*полными*. Сильно NP-полны КЛИКА, ВЫПОЛНИМОСТЬ, ВЕРШИННОЕ ПОКРЫТИЕ, ИЗОМОРФИЗМ ПОДГРАФУ, — и другие NP-полные задачи, в постановках которых нет чисел.

Задача ЦЛП тоже *сильно* NP-*полна*, хотя и числовая, — ибо заключение ее коэффициентов даже в прокрустово ложе $\{0, 1\}$ оставляет задачу NP-полной. Псевдополиномиальных задач не так много (в основном РЮКЗАКИ и некоторые задачи составления расписаний), но тема популярна в своем негативе.

✓ PSPACE-*класс* охватывает задачи, решаемые с полиномиально ограниченной памятью. PSPACE весьма широк, в него входят P, NP, co-NP и другие задачи. Разговор, конечно, теряет смысл в случае P = PSPACE, что в принципе не исключено, но остается под вопросом.

✓ Совокупность задач, которые могут быть полиномиально решены с помощью *интерактивных доказательств* (раздел 4.9), образует *класс* IP.

11.5. Линейное программирование

✓ Вариант распознавания задачи линейного программирования,

$$\langle c, x\rangle \to \max, \quad Ax \leqslant b, \quad x \geqslant 0,$$

звучит так: *существует ли вектор из* $X = \{x : Ax \leqslant b, \quad x \geqslant 0\}$, *такой что* $\langle c, x\rangle \geqslant K$ *для априори данного* $K \geqslant 0$? Таким образом, распознавание в данном случае — есть *задача* ЛН (ЛИНЕЙНОЕ НЕРАВЕНСТВО) о разрешимости системы неравенств

$$Ax \leqslant b, \quad x \geqslant 0, \quad \langle c, x\rangle \geqslant K.$$

✓ Задачу ЛП традиционно относили к дискретным, несмотря на непрерывность аргумента x, по той причине, что ее решение заведомо достигается в одной из вершин *допустимого многогранника* $X = \{x : Ax \leqslant b, \quad x \geqslant 0\}$. Потом было осознано, что *длину входа* надо оговаривать. Поэтому ЛП как задача комбинаторного анализа предполагает либо целочисленность коэффициентов, либо рациональность при заданной точности, — что равносильно, ибо рациональные коэффициенты в $\sum_j a_{ij}x_j \leqslant b_i$ переходят в целые после умножения неравенств на подходящее целое число N. Аргумент x рационален в любом случае.

✓ Для практического решения задачи ЛП до сих пор используется *симплекс-метод*, где на каждом шаге осуществляется целенаправленный переход в соседнюю вершину, в которой значение $\langle c, x\rangle$ лучше предыдущего. Алгоритм, как правило, хорошо работает, но есть примеры, когда по необходимости перебираются почти все вершины допустимого многогранника. Поэтому фактически алгоритм переборный, и долгое время многочисленные попытки найти способ решения задачи ЛП, который бы заведомо избавлял от экспоненциального роста объема вычислений, не давали результата. Начинало казаться, что ЛП $\notin$ P.

✓ При задании параметров и переменных рациональными числами линейные задачи меняют свое лицо, ибо возникают полиномиальные зависимости

битовых размеров выхода от входа, т. е. решения от условия. А поскольку битовый размер ограничивает сами числа, появляются принципиальные для вычислений возможности априорных оценок решения задач ЛН и ЛП.

✓ Если битовый размер вектора x не превосходит величины L, т. е. все x_i записываются с помощью не более L двоичных знаков, то $\|x\| \leqslant 2^L$, а точность задания x не более 2^{-L}, т. е. знаменатели координат меньше 2^L.

Этот простой факт определяет главный механизм, обеспечивающий априорные оценки в задачах, связанных с линейной алгеброй, где размеры вычисляемых величин, как правило, зависят полиномиально от размеров входа. В этот же круг задач попадают неравенства и оптимизация.

5.3.1 **Лемма.** *Если система неравенств $Ax \leqslant b$ разрешима, то она имеет решение, размер которого не превосходит $L([A,b])$. Соответственно, норма этого решения ограничена сверху величиной $2^{L([A,b])}$, точность — не более $2^{-L([A,b])}$.*

✓ 5.3.2 **Лемма.** *Если задача ЛП $\langle c, x\rangle \to \max$, $Ax \leqslant b$ имеет конечное решение, то величина оптимума не превосходит $2^{L([A,b,c])}$.*

✓ Что касается вопросов точности порядка 2^{-L}, то подоплека здесь следующая. Если у задачи есть решение, то у нее есть решение x_L с L знаками после запятой, т. е. в одном из узлов ε-сети $\varepsilon = 2^{-L}$. При наличии оценки $x_L \in V$, где область V настолько мала, что не может содержать более одного узла ε-сети, задача сводится к проверке этого узла. Узел не удовлетворяет — решения нет. Именно такой фокус работает в *алгоритме эллипсоидов* (раздел 5.4), который полиномиально решает задачу ЛН, а значит и ЛП.

✓ *Алгоритмы внутренней точки*, также полиномиально решающие задачу ЛП, связаны с движением изображающей точки в относительной внутренности допустимого многогранника X. Все они базируются на итерационных процедурах типа

$$x_{k+1} = x_k + \Delta x_k,$$

где направление и норма корректировки Δx_k выбираются так, чтобы оставаться в X, но двигаться при этом достаточно быстро. Сходимость такой процедуры со скоростью геометрической прогрессии обеспечивает попадание x_k в достаточно малую окрестность оптимальной вершины X за полиномиальное число шагов. А этого «по причинам ограниченной точности» хватает для точного решения задачи.

11.6. Арифметика и криптография

✓ Источником эффективных тестов на простоту служит

6.2.1 **Малая теорема Ферма**. *Для простого N и любого x, взаимно простого с N,*

$$x^{N-1} \equiv 1 \pmod{N}.$$

✓ *Тест Ферма* работает с гарантией только в одном направлении: если $x^{N-1} \equiv 1 \pmod N$ нарушается, то N — составное, т. е. СЧ имеет ответ «да». В противном случае N может быть простым с некоторой вероятностью, каковая возрастает при повторении теста с другим основанием x. Но существуют составные *числа Кармайкла*, которые проходят *тест Ферма* для всех x, не являющихся их делителями.

✓ Тест Ферма можно улучшить, заметив, что для простого числа $N \neq 2$ — квадратный корень из $x^{N-1} \equiv 1 \pmod N$ дает

$$x^{(N-1)/2} = \pm 1 \pmod N,$$

что приводит к *тесту Леманна*. На этом пути возникает

6.2.2 **Тест Рабина—Миллера**, *состоящий в проверке выполнения одного из двух условий*:

$$x^m = 1 \pmod N, \quad \textit{либо} \quad \exists r < k : x^{m2^r} = -1 \pmod N,$$

где N нечетно и $N - 1 = m \cdot 2^k$, где m нечетно.

✓ 6.2.3 **Теорема Миллера**. *Если верна расширенная гипотеза Римана и N проходит тест* 6.2.2 *при любом*

$$x : 1 < x < 2\log^2 N,$$

то N — простое.

Это уже полиномиальный тест на простоту с нулевой вероятностью ошибки, но червоточинка остается в другом обличье — в виде предположения о справедливости *расширенной гипотезы Римана*. Точка была поставлена открытием полиномиального алгоритма AKS — описание в разделе 6.3.

✓ Алгоритмы решения задач распознавания ПЧ, СЧ обходят стороной поиск разложения N на множители. Полиномиальные рецепты до сих пор неизвестны. Ситуация несколько странная, потому что в стандартных комбинаторных задачах из класса P (типа МОД) вариант, на котором достигается ответ «да» или «нет», разыскивается полиномиально. Здесь вопрос повис нерешенным.

✓ Система шифрования RSA на сегодняшний день является крупным потребителем теорем и алгоритмов из области ПЧ, СЧ и *факторизации составных чисел*. Идея, лежащая в основе RSA, проста до гениальности. Цифровой текст x шифруется с помощью легко вычисляемой функции

$$f(x) = x^e \pmod N,$$

где числа e и N, составляющие *открытый ключ*, общедоступны. При определенных требованиях к $\{e, N\}$ функция $f(x)$ обратима, но f^{-1} вычисляется уже не просто.

Точнее говоря, значения f^{-1} легко определяются, если известен секрет, без которого расшифровка практически невозможна. Для расшифровки сообщения $y = f(x)$ надо всего лишь решить уравнение

$$x^e = y \pmod N,$$

но в этом трудность и заключается, ибо возникает необходимость разложения N на простые множители.

✓ Существование *функций с секретом*: $f(x)$ считается быстро, а время вычисления $f^{-1}(x)$ зашкаливает, если не известен *секрет*, с помощью которого значения $f^{-1}(x)$ также быстро вычисляются, — в принципе не доказано, но фактически такой функцией считается $f(x) = x^e \pmod N$, используемая в RSA. Еще одна гипотетическая *функция с секретом* — вычисление *дискретного логарифма* — определение целого x в равенстве

$$a^x = b \pmod N,$$

где N — простое число большее двух, a — порождающий элемент мультипликативной группы вычетов $\mathbb{Z}_N^\times$ и целое $b < N$.

✓ Что касается *систем с нулевым разглашением* (раздел 6.8), то в криптографии это одна из главных линий, а в теории сложности алгоритмов — второстепенная.

11.7. Геометрический подход

✓ Определенный свет на отличие легко- от трудно решаемых комбинаторных задач проливает их геометрическая интерпретация, состоящая в описании

$$c(x) \to \max, \quad x \in X,$$

где X — дискретное множество, а $c(x)$ — целевая функция. В линейном случае, $c(x) = \langle c, x\rangle$, множество X без ущерба для задачи можно заменить выпуклой оболочкой со-X, что приводит *вроде бы* к задаче ЛП[5]:

$$\langle c, x\rangle \to \max, \quad x \in \text{со-}X, \tag{11.1}$$

каковую для краткости обозначают иногда как $[X, c]$.

✓ Принципиальное отличие такой постановки от стандартной задачи ЛП заключено в описании многогранника со-X не определяющими неравенствами, как обычно, а выпуклой оболочкой вершин. Не будь разницы в описании задачи ЛП, ларчик бы открывался совсем просто. Масса NP-полных задач легко

[5] Замена оптимизации на конечном множестве X оптимизацией на сплошном многограннике со-X возможна благодаря линейности критерия $\langle c, x\rangle \to \max$.

записывается в виде (11.1), после чего полиномиальный *алгоритм Хачияна* или *Кармаркара* решает задачу ЛП, и проблема исчерпана. На этом пути, однако, возникает «непреодолимая» загвоздка. Приведение (11.1) к стандартному виду

$$\langle c, x\rangle \to \max, \quad Ax \leqslant b,$$

требует привлечения вычислительных ресурсов, которые могут иметь «экспоненциальную природу», после чего экономия на решении самой задачи ЛП ничего не дает.

✓ Другой ракурс рассмотрения задачи (11.1) дает *конусная интерпретация*, сопоставляющая точкам $x \in X$ конусы

$$K(x) = \{w : \langle w, x\rangle \geqslant \langle w, y\rangle,\ \forall y \in X\}.$$

Содержательно конус $K(x)$ представляет собой множество градиентов $w \in \mathbb{R}^m$, при которых решение задачи ЛП (для $c = w$) достигается в вершине x. Поэтому *задача* (11.1) *равносильна поиску конуса* $K(x)$, *в котором лежит градиент с линейного функционала* $\langle c, x\rangle$.

✓ Дальнейший сюжет развивается вокруг конусного разбиения

$$\bigcup K(x) = \mathbb{R}^m \tag{11.2}$$

для «аномальных» многогранников, у которых все вершины смежные[6]. Конусный букет (11.2) такого многогранника обладает тем свойством, что любые два конуса граничат друг с другом по $(m-1)$-мерной грани. В результате, если алгоритм бракует часть конусов, деля пространство $\mathbb{R}^m$ на две части плоскостями, то на каждом шаге может быть отброшен лишь один конус — и полного перебора избежать принципиально невозможно. На этом феномене высокой *плотности графа* со-X базируется идея отделения простых задач от сложных.

✓ Высокая *плотность графов* многогранников при больших размерностях является скорее правилом, нежели исключением. Вот один из вариантов точного утверждения. Выберем наугад (равновероятно) k вершин m-мерного куба $[0, 1]^m$, и пусть X обозначает их выпуклую оболочку, а p_{km} — вероятность того, что все вершины многогранника X попарно смежны.

При $m \geqslant 4$ *и* $k \leqslant 2^m$ *справедлива оценка*

$$p_{km} > 1 - \frac{k^4}{4}\left(\frac{5}{8}\right)^m.$$

[6] *В* $\mathbb{R}^m$ *при* $m \geqslant 4$ *существуют выпуклые многогранники с произвольным количеством смежных вершин.* Такими многогранниками являются — *циклические.*

✓ Комбинаторные алгоритмы обычно работают на основе линейных сравнений. Решая задачу $[X, c]$, т. е. (11.1), алгоритм на каждом шаге проверяет неравенство $\langle f_i, c\rangle > 0$ и переходит к следующему тесту $\langle f_{i+1}, c\rangle > 0$, где выбор функционала f_{i+1} происходит дихотомически — в зависимости от ответа «да» или «нет» на вопрос о справедливости неравенства $\langle f_i, c\rangle > 0$. Это, собственно, каркас работы алгоритма — его *линейное разделяющее дерево* (ЛРД). Детали, иногда громоздкие, остаются за кулисами. *Жадный алгоритм*, например, решающий задачу МОД, помимо линейных сравнений, проверяет постоянно, не образуются ли циклы при добавлении ребер, и не охвачены ли уже все узлы (правило остановки), — что находится за рамками *бинарного разделяющего каркаса*. Но именно *глубина оптимального ЛРД* свидетельствует о трудоемкости алгоритма — по крайней мере, об оценке трудоемкости снизу.

✓ ЛРД разбивает $\mathbb{R}^m$ плоскостями $\langle f_i, c\rangle = 0$, и любая ветвь β, заканчивающаяся в висячей вершине, отвечает последовательности полупространств $\langle f_i, c\rangle \leqslant 0$, вырезающих в $\mathbb{R}^m$ конус K_β, в который попадает вектор $c \in K_\beta$. Причем $K_\beta \subset K(x)$, где $K(x)$ — конус разбиения. В итоге $c \in K_\beta \subset K(x)$, — и алгоритм, основанный на ЛРД, обеспечивает выбор решения $x \in X$ для задачи $[X, c]$.

7.4.1 **Теорема Мошкова**. *Оптимальное ЛРД для любой задачи* $[X, c]$ *имеет полиномиальную глубину*

$$D(X) = O\big(m^3 \log |X|\big).$$

✓ Теорема 7.4.1 воспринимается поначалу как NP = P, ибо число необходимых сравнений оказывается полиномиальным, а это как раз и составляет основной вклад в трудоемкость во всех известных алгоритмах. Беда заключается в том, что ЛРД — это не алгоритм, а виртуальный каркас. Но в принципе — существование полиномиального ЛРД говорит о том, что на пути создания полиномиального алгоритма нет препятствий принципиального характера, каковые могли бы перечеркнуть надежды. И если уж говорить об ожиданиях, то на данном уровне непонимания — возникают аргументы, скорее, в пользу NP = P, хотя градус алхимического стиля мышления при этом — довольно высок.

✓ Известные алгоритмы решения комбинаторных задач базируются на линейных сравнениях[7)], но в более приземленном исполнении, чем это подразумевается за кадром ЛРД. Суть дела удобнее всего объяснить, опираясь на конусное разбиение (11.2).

Движение по ветви ЛРД подразумевает исключение каждый раз одного из полупространств $\langle f_i, c\rangle > 0$ или $\langle f_i, c\rangle < 0$. *Алгоритм прямого типа* также проводит линейное сравнение

$$\langle f_i, c\rangle > 0 \quad (?),$$

[7)] Кроме ЛИНЕЙНОГО ПРОГРАММИРОВАНИЯ и ПРОСТОГО ЧИСЛА.

но исключает не все полупространство $H^- = \{c : \langle f_t, c\rangle < 0\}$, а лишь конусы $K(x)$ разбиения (11.2), лежащие в H^-. И если конусное разбиение (11.2) таково, что все конусы попарно граничат друг с другом, — отсечь более одного конуса за один раз не удается. Это, разумеется, крайняя ситуация. Но алгоритм остается переборным и в том случае, когда бóльшая часть конусов $K(x)$ попарно граничат друг с другом — иначе говоря, когда плотность графа многогранника со-X экспоненциальна.

✓ Из сказанного ясно, что *алгоритмы прямого типа принципиально не могут решать за полиномиальное время задачи, имеющие высокую плотность графа многогранника* со-X. Последнее (высокая плотность графа) установлено для многих NP-полных задач.

✓ Если в ограничениях комбинаторных задач целочисленным переменным разрешить принимать непрерывные значения, возникают описания так называемых *релаксационных многогранников*. Вот один из простейших примеров.

Пусть координаты вектора $x \in \mathbb{R}^m$, $m = n^3$, имеют трехиндексную нумерацию. Рассмотрим многогранник M_n, определяемый системой ограничений

$$\sum_{i,j=1}^{n} x_{ijk} = \sum_{i,k=1}^{n} x_{ijk} = \sum_{j,k=1}^{n} x_{ijk} = 1, \quad x_{ijk} \geqslant 0. \tag{11.3}$$

При условии целочисленности вершины многогранника M_n образуют множество X_n, соответствующее задаче ТРЕХМЕРНОЕ СОЧЕТАНИЕ, — и граф со-X_n имеет экспоненциальную плотность. В то же время M_n имеет и нецелочисленные вершины. При этом ребра многогранника со-X_n являются ребрами M_n, т. е. граф со-X_n — подграф графа M_n. Поэтому *граф M_n также имеет экспоненциальную по n плотность* — и это весьма примечательный факт, ибо *задача ЛП*

$$\langle c, x\rangle \to \max, \quad x \in M_n,$$

при фасетном описании (7.10) *имеет допустимый многогранник с экспоненциальной плотностью вершин, но решается — полиномиально, как любая задача ЛП.*

11.8. Вероятностные алгоритмы

✓ Ни один вероятностный алгоритм не привел к скачку из NP в P.

✓ Дополнение *вероятностной машины Тьюринга* (ВМТ) *Оракулом* $\mathcal{O}$ порождает иногда принципиально новые возможности, возникающие, в частности, в *интерактивных системах* (раздел 4.9), где вместо ВМТ предпочитают говорить о *Вычислителе* $\mathcal{V}$.

Такова, например, *система с нулевым разглашением*, основанная на игре двух лиц, в которой *Оракул* $\mathcal{O}$ знает решение некой данной задачи, и в процессе

диалога пытается доказать, возможно, блефуя, «что знает», — другому игроку, *Вычислителю* $\mathcal{V}$. Последний в проверках ограничен «полиномиальными средствами». Иллюстрация на ИЗОМОРФИЗМЕ ГРАФОВ рассматривается в основном тексте. Уникальность ситуации в том, что $\mathcal{O}$ убеждает $\mathcal{V}$ в знании изоморфизма φ, не предъявляя самого φ. И если изоморфизм взять в качестве пароля, ключа, — появляются невиданные криптографические возможности.

✓ Среди интерактивных систем наиболее известен *класс* PCP, определяемый как совокупность задач, решаемых специальной интерактивной системой (раздел 8.3). PCP-*теорема* утверждает равенство

$$\boxed{\mathrm{NP} = \mathrm{PCP},}$$

дающее на NP взгляд с другой стороны.

11.9. Квантовые вычисления

✓ Если в каждую ячейку регистра поместить *квантовую частицу*, способную пребывать в одном из двух возможных состояний, — условно обозначаемых 0 или 1, — то *в одном регистре будут присутствовать все* 2^n *возможностей* (*чисел*). Потому что квантовая система как бы пребывает во всех возможных состояниях одновременно, и есть лишь вероятности обнаружить ее при измерении в том или ином конкретном состоянии.

✓ Квантовому компьютеру для записи 2^n чисел требуется всего n ячеек. Следующий трюк еще более удивителен. В квантовом случае достаточно выполнить один раз преобразование f исходного регистра, где содержатся все n-разрядные x, и функция $f(x)$ готова — для ее записи потребуется $m+n$ ячеек. В этом, собственно, заключается главный «фокус».

✓ Состояние частицы в ячейке «выясняется» только после измерения, а *латентное состояние* представляет собой линейную смесь

$$\lambda_0|0\rangle + \lambda_1|1\rangle, \tag{11.4}$$

где λ_0, λ_1 — *комплексные амплитуды*, квадраты модулей которых являются вероятностями обнаружения частицы при измерении в соответствующем — $|0\rangle$ или $|1\rangle$ — состоянии. Состояния $|0\rangle$, $|1\rangle$ образуют *базис*. Базис могут составлять и два других ортогональных состояния вида $\alpha|0\rangle + \beta|1\rangle$.

✓ Единичный вектор (11.4) называют *квантовым битом*, или *q-битом* (*кубитом*). Обычный классический бит, как единица информации, есть по сути двоичный разряд, ячейка, куда помещается 1 или 0. Кубит — это ячейка, куда помещается частица в состоянии $|\psi\rangle = \lambda_0|0\rangle + \lambda_1|1\rangle$.

✓ Система из двух *q-битов* может находиться в любом состоянии

$$\lambda_{00}|00\rangle + \lambda_{01}|01\rangle + \lambda_{10}|10\rangle + \lambda_{11}|11\rangle,$$

где λ_{ij} — по-прежнему *комплексные амплитуды*, квадраты модулей которых являются вероятностями обнаружения теперь уже пары частиц при измерении в соответствующем $|ij\rangle$-состоянии. «Новость» заключается в том, что набор коэффициентов λ_{ij} не обязательно получается в виде обычной композиции простых состояний, — и это отвечает *сцепленным состояниям*.

✓ *Вычислительные квантовые процессы представляют собой унитарные преобразования ψ-состояний.*

✓ Особое внимание уделяется унитарным преобразованиям, полезным с точки зрения организации вычислительного процесса. Таковым является, например, однокубитовое *преобразование Уолша—Адамара*

$$H = \frac{1}{\sqrt{2}}\begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix},$$

трансформирующее состояние $|0\rangle$ в суперпозицию состояний $\frac{1}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}|1\rangle$, а $|1\rangle$ — в суперпозицию $\frac{1}{\sqrt{2}}|0\rangle - \frac{1}{\sqrt{2}}|1\rangle$. Применяя H к n кубитам по отдельности, получаем суперпозицию всевозможных 2^n состояний,

$$\frac{1}{\sqrt{2^n}}(|00\ldots00\rangle + |00\ldots01\rangle + \ldots + |11\ldots11\rangle),$$

которая обратным преобразованием переводится в состояние $|00\ldots0\rangle$.

✓ Для получения функции $f(x)$ с n-разрядным входом и m-разрядным выходом вводится оператор

$$U_f : |x, y\rangle \to |x, y \oplus f(x)\rangle,$$

где $\oplus$ обозначает поразрядное сложение по модулю 2. Применение U_f к $|x, 0\rangle$ дает $|x, f(x)\rangle$ в регистре из $(n+m)$ ячеек. А поскольку U_f *линейный унитарный оператор*, — суперпозиция всевозможных состояний по x порождает суперпозицию всевозможных состояний $f(x)$. Тем самым вся информация о функции загоняется в регистр.

✓ Задача поиска решения уравнения $f(x) = 1$, где функция $f(x)$ принимает значения из $\{0, 1\}$, — охватывает массу практических ситуаций, и в неблагоприятных ситуациях требует $O(N)$ шагов, потому что в худшем случае все N значений x надо перебрать. Квантовый *алгоритм Гровера* позволяет осуществлять поиск за $O(\sqrt{N})$ шагов (раздел 10.5). *Алгоритм Шора* (раздел 10.7) решает более впечатляющую задачу, добиваясь за полиномиальное время факторизации составного числа.

Сокращения и обозначения

◄ и ► — начало и конец рассуждения, темы, доказательства

(?) — предлагает проверить или доказать утверждение в качестве упражнения, либо довести рассуждение до «логической точки», — но не является вопросом «правильно или неправильно?»

(!) — предлагает обратить внимание

«в томм случае» — «в том и только том случае»

НОД — наибольший общий делитель

НОК — наименьшее общее кратное

$A \Rightarrow B$ — из A следует B

$x \in X$ — x принадлежит X

$X \cup Y$, $X \cap Y$, $X \setminus Y$ — объединение, пересечение и разность множеств

$X \subset Y$ — X подмножество Y, в том числе имеется в виду возможность $X \subseteq Y$, т. е. между $X \subset Y$ и $X \subseteq Y$ различия не делается

$\mathbb{N}$ — множество натуральных чисел $\{1, 2, \ldots\}$

$\mathbb{Z}$ — множество целых чисел $\{\ldots, -1, 0, 1, \ldots\}$

$\mathbb{Q}$ — множество рациональных чисел

$\mathbb{R} = (-\infty, \infty)$ — вещественная прямая

$\mathbb{C}$ — комплексная плоскость

$x = a \pmod{p}$ — «x при делении на p дает в остатке a»

$\lfloor a \rfloor$ — округление числа a до целого в меньшую сторону

$\lceil a \rceil$ — округление числа a до целого в бóльшую сторону

$|C|$ — мощность множества C, число элементов в C

$\langle x, y \rangle$ — скалярное произведение векторов x и y,

$$\langle x, y \rangle = x_1 y_1 + \ldots + x_n y_n.$$

Литература

1. *Айгнер М.* Комбинаторная теория. М.: Мир, 1982.
2. *Ахо А., Хопкрофт Дж., Ульман Дж.* Построение и анализ вычислительных алгоритмов. М.: Мир, 1979.
3. *Бондаренко В. А.* Полиэдральные графы и сложность в комбинаторной оптимизации. Ярославль: Изд-во Яросл. гос. ун-та, 1995.
4. *Бондаренко В. А., Максименко А. Н.* Геометрические конструкции и сложность в комбинаторной оптимизации. М.: Издательство ЛКИ/URSS, 2008.
5. *Босс В.* Интуиция и математика. М.: Айрис-Пресс, 2003; 2-е изд. М.: КомКнига/URSS, 2008.
6. *Босс В.* Лекции по математике. Т. 1: Анализ; Т. 2: Дифференциальные уравнения; Т. 3: Линейная алгебра; Т. 4: Вероятность, информация, статистика; Т. 5: Функциональный анализ; Т. 6: От Диофанта до Тьюринга; Т. 7: Оптимизация; Т. 8: Теория групп; Т. 9: ТФКП. М.: URSS, 2004–2007.
7. *Гэри М., Джонсон Д.* Вычислительные машины и труднорешаемые задачи. М.: Мир, 1982.
8. *Деза М. М., Лоран М.* Геометрия разрезов и метрик. М.: МЦНМО, 2001.
9. *Емеличев В. А., Ковалев М. М., Кравцов М. К.* Многогранники, графы, оптимизация. М.: Наука, 1981.
10. Кибернетический сборник. Вып. 12. М.: Мир, 1975.
11. *Кнут Д.* Искусство программирования для ЭВМ. Т. 3: Сортировка и поиск. М.: Мир, 1978.
12. *Кофман А., Дезебай Г.* Сетевые методы планирования. М.: Прогресс, 1968.
13. *Левин Л. А.* Универсальные задачи перебора // Проблемы передачи информации. 1973. **IX**, вып. 3. 115–116.
14. Линейные неравенства и смежные вопросы. М.: ИЛ, 1959.
15. *Минский М.* Вычисления и автоматы. М.: Мир, 1971.
16. *Оре О.* Теория графов. М.: Наука, 1980 ; 4-е изд. М.: Издательство ЛКИ/URSS, 2008.

17. *Пападимитриу Х.*, *Стайглиц К.* Комбинаторная оптимизация. Алгоритмы и сложность. М.: Мир, 1985.
18. *Риордан Дж.* Введение в комбинаторный анализ. М.: ИЛ, 1963.
19. *Рокафеллар Р.* Выпуклый анализ. М.: Мир, 1973.
20. *Хачиян Л. Г.* Полиномиальные алгоритмы в линейном программировании // ЖВМ и МФ. 1980. **20**, № 1. 51–69.
21. *Agrawal M.*, *Kayal N.*, *Saxena N.* Primes is in P // Annals of Math. 2004. **160** (2). 781–793.
22. *Borgwardt K. H.* The Simplex Method — A Probabilistic Analysis. Heidelberg: Springer-Verlag, 1987.
23. *Cook S. A.* The complexity of theorem-proving procedure // Proc. 3-d Annual ACM Symposium on the Theory of Computing. Shaker Heights, Ohio. 1971. 151–159 (есть русский перевод в сб. [10]).
24. *Karp R. M.* Reducibility among Combinatorial problems // Complexity of computer computations. Proc. Symp. March. 1972. 85–103 (есть русский перевод в сб. [10]).

Предметный указатель