Издательство «Знание»

МАТЕМАТИКА КИБЕРНЕТИКА

Новое в жизни, науке, технике

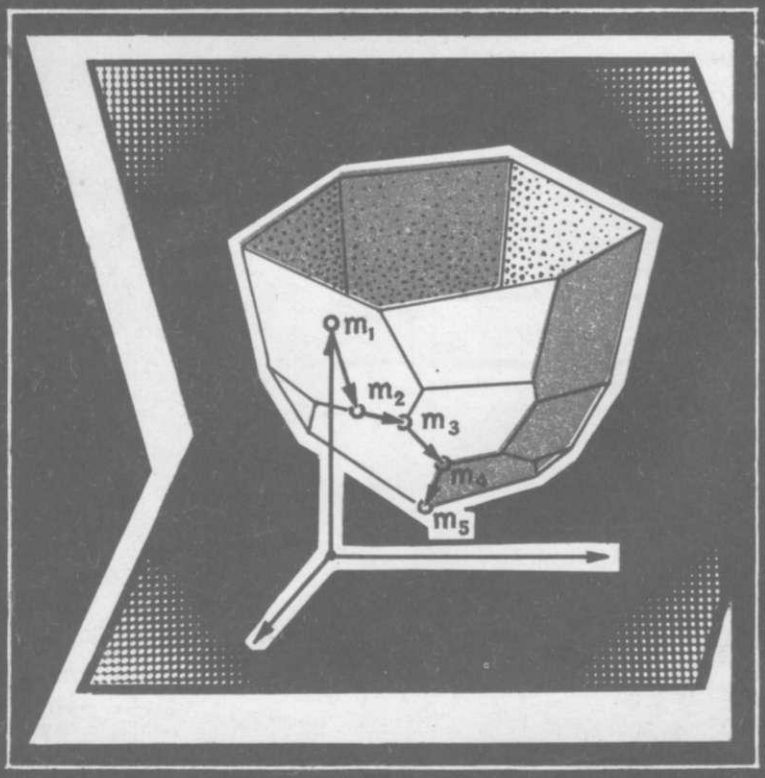
Подписная научно-популярная серия

Издается ежемесячно с 1967 г.

Л. Г. Хачиян

Сложность задач линейного программирования

1987/10



Новое
в жизни,
науке,
технике

МАТЕМАТИКА
КИБЕРНЕТИКА

Издается
ежемесячно
с 1967 г.

10/1987

Л. Г. Хачиян

СЛОЖНОСТЬ ЗАДАЧ ЛИНЕЙНОГО ПРОГРАММИРОВАНИЯ

Подписная
научно-
популярная
серия

СОДЕРЖАНИЕ

Издательство
«Знание»
Москва
1987

ББК 22.18
Х29

Автор: Леонид Генрихович ХАЧИЯН, доктор физико-математических наук, ведущий научный сотрудник Вычислительного центра АН СССР.

Рецензент: С. А. Ашманов, доктор физико-математических наук.

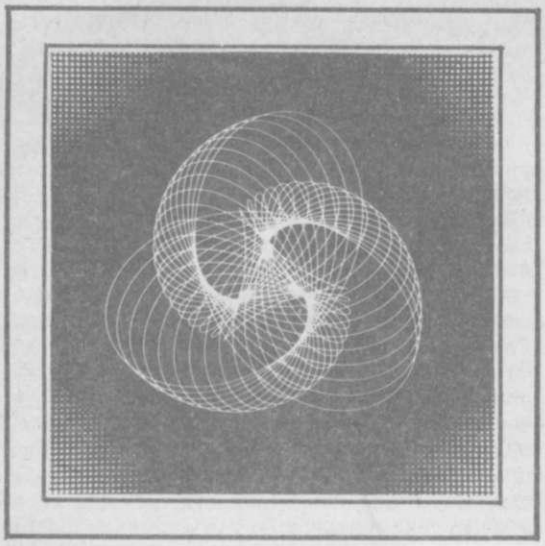
Хачиян Л. Г.

Х29 Сложность задач линейного программирования.— М.: Знание, 1987.— 32 с.— (Новое в жизни, науке, технике. Сер. «Математика, кибернетика»; № 10).

11 к.

Линейное программирование — широко применяемый аппарат прикладной математики, основы которого сейчас включены даже в школьные программы. Несмотря на традиционность предмета (первые теоретические работы относятся еще к началу прошлого века) и множество публикаций по теории и приложениям, исследования в этой области прикладной математики продолжаются и в наши дни. В брошюре рассказывается о некоторых полученных в последние годы результатах, связанных с изучением сложности решения задач линейного программирования на ЭВМ.

1402040000 **ББК 22.18**



ВВЕДЕНИЕ

Предмет линейного программирования

Линейным программированием называется раздел прикладной математики, изучающий теорию, приложения и методы решения конечных систем линейных неравенств

$$\begin{aligned} a_{11}x_1+a_{12}x_2+\ldots+a_{1n}x_n &\leqslant b_1,\\ a_{21}x_1+a_{22}x_2+\ldots+a_{2n}x_n &\leqslant b_2,\\ \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\\ a_{m1}x_1+a_{m2}x_2+\ldots+a_{mn}x_n &\leqslant b_m \end{aligned} \qquad (1)$$

с конечным числом вещественных неизвестных $x_1, x_2, \ldots, x_n$. Общая задача линейного программирования состоит в нахождении такого решения системы неравенств (1), которое обеспечивало бы максимизацию заданной линейной функции

$$c_1x_1+c_2x_2+\ldots+c_nx_n \to \max \qquad (2)$$

неизвестных. Задача линейного программирования (1)—(2) с n неизвестными и m ограничениями называется задачей размерности (n, m). Такая задача, следовательно, вполне задается числовой таблицей

$$\left|\begin{array}{cccc} a_{11} & a_{12} & \ldots a_{1n} & b_1\\ a_{21} & a_{22} & \ldots a_{2n} & b_2\\ \ldots & \ldots & \ldots & \ldots\\ a_{m1} & a_{m2} & \ldots a_{mn} & b_m\\ c_1 & c_2 & \ldots c_n & 0 \end{array}\right| \qquad (3)$$

своих коэффициентов.

В частном случае $c_1=c_2=\ldots=c_n=0$ множество решений задачи линейного программирования (1)—(2) совпадает с множеством решений системы линейных неравенств (1). Поэтому умение решать задачи линейного программирования подразумевает умение решать системы линейных неравенств. Существующая в линейном программировании теория двойственности (см. § 5) позволяет провести обратное сведение.

Из истории линейного программирования

Системы линейных неравенств изучались в математике давно. Первый способ их решения — метод исключения неизвестных — был предложен Ж. Фурье еще в 20-х годах прошлого столетия. Ж. Фурье также описал в геометрической форме метод решения оптимизационной задачи линейного программирования, который через сто с лишним лет стал широко применяться на практике и получил название симплекс-метода. Вслед за Фурье изучением систем линейных неравенств в связи с их приложениями в механике занимались в середине и конце прошлого века М. В. Остроградский и Г. Фаркаш. Современная теория двойственности в линейном программировании является прямым развитием исследований Г. Фаркаша и Г. Минковского, также выполненных еще в прошлом веке. И все же оформление линейного программирования в самостоятельную дисциплину прикладной математики произошло в нашем веке. Оно было связано с осознанием широты приложений линейного программирования и появлением ЭВМ, позволившими решать задачи с такими количествами неизвестных и ограничений, «которые математику представляются почти неправдоподобными» [2]. Первооткрывательские работы в этой области были выполнены в СССР Л. В. Канторовичем [12], предложившим и рассчитавшим в конце 30-х годов ряд линейных оптимизационных моделей планирования производства. В послевоенные годы активные исследования по линейным неравенствам развернулись в США, где в 1948 г. по предложению Т. Купманса и появился сам термин «линейное программирование». Ведущая роль в этих исследованиях принадлежала Дж. Данцигу, который в своих воспоминаниях [27] пишет: «Пришествие или, скорее, надежды, связанные с приходом электронного компьютера, обращение математиков-теоретиков и экономистов к реальным проблемам во время войны, интерес к механизмам планирования, наличие денег для прикладных исследований — все совпало в период 1947—1949 гг.» В области численных методов основным результатом указанных исследований явилась проведенная Дж. Данцигом алгебраизация и внедрение в практику симплекс-метода. В наше время модели и методы линейного программирования стали уже традиционными для исследований по планированию производства, математической экономике, оптимальному управлению, теории игр, комбинаторике, дискретной оптимизации, теории сложности вычислений и многим дру-

гим дисциплинам прикладной математики. Первые понятия о линейном программировании включают теперь даже в школьные учебники.

О чем эта брошюра

Имеется много книг, популярных работ и обстоятельных учебников по линейному программированию. За последние 15 лет, однако, было получено довольно много новых интересных результатов в области сложности методов линейного программирования. Проводимые здесь исследования изучают в той или иной постановке следующий важный для вычислительной математики вопрос: какой объем вычислительной работы необходим для решения систем линейных неравенств и задач линейного программирования? В настоящей брошюре мы попытаемся рассказать о некоторых из этих исследований. В соответствии с этой целью вопросы теории будут освещаться лишь в той мере, в какой они необходимы для анализа методов. Приложения, даже такие интересные, как в дискретной комбинаторной оптимизации, не будут затрагиваться вовсе. Таким образом, брошюра ориентирована на читателя, которому вопрос, как сложно решать системы линейных неравенств, кажется достаточно интересным.

Глава 1 посвящена предварительному изложению сути указанных вопросов и может рассматриваться как продолжение введения.

ГЛАВА 1. СЛОЖНОСТЬ МЕТОДОВ ЛИНЕЙНОГО ПРОГРАММИРОВАНИЯ

1. Эффективность симплекс-метода

1.1 Симплекс-метод

Как упоминалось во введении, результатом начатых после войны исследований Дж. Данцига в области методов линейного программирования явился знаменитый симплекс-метод — предложенная им в 1947 г. итеративная процедура пересчета коэффициентов таблицы (3), приводящая через конечное число шагов к точному решению произвольной задачи линейного программирования (1)—(2).

1.2. Проблема эффективности симплекс-метода

Хотя конечность числа шагов симплекс-метода доказывалась просто, доказательство не давало, по существу, никакой информации о том, насколько хорошим или плохим может оказаться метод на практике. Возникшая здесь ситуация была в некотором смысле гораздо хуже той, с какой мы сталкиваемся при решении систем линейных равенств

$$\begin{aligned} a_{11}x_1+a_{12}x_2+\ldots+a_{1n}x_n&=b_1,\\ a_{21}x_1+a_{22}x_2+\ldots+a_{2n}x_n&=b_2,\\ \ldots\ldots\ldots\ldots\ldots\ldots&\\ a_{m1}x_1+a_{m2}x_2+\ldots+a_{mn}x_n&=b_m, \end{aligned} \qquad (4)$$

или, как принято говорить, систем линейных уравнений. Еще в школе мы учимся решать такие системы методом исключения неизвестных Гаусса, который, как и симплекс-метод, сводится к конечному числу арифметических пересчетов таблицы входных коэффициентов системы. Стоимость одного шага симплекс-метода и метода исключения неизвестных Гаусса по порядку одинакова — в обоих случаях для пересчета таблицы коэффициентов достаточно выполнить $\mathrm{const}\cdot nm$ арифметических операций. Вопросы, связанные с необходимой точностью вычислений, также решаются для обоих методов сходным образом. Однако в оценке сложности метода необходимо учитывать число требующихся шагов. Для метода Гаусса тут имеется полная ясность: число k шагов процедуры при решении общей системы линейных равенств размерности (n, m) равно минимуму из числа уравнений и неизвестных $k=\min(n, m)$, так что полное количество требующихся методу арифметических операций оказывается не большим $\max(n, m)\min^2(n, m)$ и ограниченным полиномом третьей степени от размеров n и m задачи. Приступая, скажем, к решению системы линейных уравнений с $n=m=100$ неизвестными и равенствами, мы заранее знаем, что полное количество требуемых операций не превзойдет 10^6 — скромный объем вычислений даже для компьютеров первых поколений. При переходе к решению систем линейных неравенств и задач линейного программирования симплекс-методом ситуация заметно усложняется. При заданной размерности (n, m) число шагов k симплекс-метода зависит от конкретной задачи, и его не удавалось оценить сверху вообще никакой полиномиальной функцией

$$k\leqslant\mathrm{const}\,[\max(n, m)]^d$$

размерности с некоторой фиксированной степенью d. Ничто, однако, не мешало запрограммировать метод и попробовать его в действии на конкретных задачах. Это было сделано, и к 1948 г. стало выясняться, что, «к счастью, метод работал» [27], подтверждая забытое предсказание Ж. Фурье, что с помощью симплекс-метода «очень быстро» [31] приходят в оптимум*. Оказалось, что для реальных задач число шагов симплекс-метода редко превосходит более чем в несколько раз количество неизвестных и ограничений. С тех пор с помощью симплекс-

* Эта цитата, взятая из [47], полностью приведена в § 4.

метода решено множество прикладных задач линейного программирования, иногда с тысячами неизвестных и ограничений.

1.3. Примеры Кли и Минти

И все же удачная работа метода на множестве примеров еще не есть доказательство его хорошей работы для всех задач. В 1972 г. В. Кли и Дж. Минти указали примеры задач линейного программирования, решить которые даже при скромной размерности симплекс-метод оказался не в состоянии. В своей статье «Как хорош симплекс-алгоритм?» [40] Кли и Минти описали, подтверждая давние опасения исследователей, весьма простые по внешнему виду задачи линейного программирования любой заданной размерности (n, m), на решение которых один из наиболее распространенных вариантов симплекс-метода тратит экспоненциально большое

$$k \geqslant 2^{\min(n, \frac{1}{2}m)}$$

и, следовательно, не ограниченное сверху никакой полиномиальной функцией размерности число шагов. При столкновении с такой плохой задачей размерности (50, 100) симплекс-метод порождает последовательность из 10^{18} арифметических операций, на выполнение которой компьютеру с быстродействием миллиард операций в секунду потребовалось бы несколько десятков лет. Позже такие экспоненциально плохие задачи обнаружились и для других версий симплекс-метода.

Для объяснения хорошей работы симплекс-метода на практике проводились теоретические исследования числа его шагов в среднем на задачах размерности (n, m) при тех или иных входных статистиках (достаточно «симметричных» для удобства геометрико-вероятностного анализа). Эти исследования показали, что относительно некоторых статистик среднее число шагов симплекс-метода действительно полиномиально по размерности (об этом более подробно см. в § 4).

1.4. Проблема существования алгоритма линейного программирования полиномиальной алгебраической сложности

Вернемся к обсуждению сложности решения задач линейного программирования в терминах не средних, а гарантированных объемов вычислений. В конце концов если для решения любой системы линейных равенств достаточно кубического по размерности (n, m) числа арифметических операций, можно ли предложить метод с такой или хотя бы полиномиальной оценкой числа операций* для решения любых систем линейных неравенств, или, что то же самое, задач линейного программирования? Этот вопрос, волновавший исследователей задолго до того, как В. Кли и Дж. Минти обнаружили «пятна на Солнце», остается нерешенным вот уже несколько десятилетий. Наилучшее достижение здесь — данное Н. Мегиддо [43] построение требуемого полиномиального метода сложности const max (n,m) $\min^3(n,m)$ log min (n,m) операций для случая, когда в каждое неравенство системы входит не более двух неизвестных (в разные неравенства могут входить разные пары неизвестных). Нетрудно видеть, что случай, когда в каждое неравенство входит одна неизвестная, решается школьным «методом интервалов», а случай, когда в неравенства могут одновременно входить три неизвестных, так же труден для вопроса существования полиномиального алгоритма, как общий. Н. Мегиддо [44] также показал, что если считать число неизвестных в задачах фиксированным**, то можно построить требуемый алгоритм с линейным по второму размеру числом операций $c(n)m$. В этой оценке, однако, функция $c(n)$ не только не полиномиальна, но даже растет по n быстрее экспоненты.

Теперь мы рассмотрим несколько иную постановку проблемы существования полиномиального алгоритма линейного программирования и увидим, что в такой постановке она уже имеет положительное решение.

2. Полиномиальная разрешимость линейного программирования

2.1. Битовая сложность

До сих пор мы предполагали, что сложность вычислительного метода измеряется количеством требуемых ему арифметических операций, которые могут выполняться над произвольными вещественными числами. Такой подход к сложности называется алгебраическим. Алгебраический подход игнорирует дискретную структуру данных в вычислительной машине и не рассматривает вопрос о том, как должны выполняться операции в машине и как зависит их стоимость от цифровой длины чисел. Если эта стоимость считается постоянной, то для реалистичности такого предположения на запись чисел должно отводиться фиксированное число цифр, как, например, в обычном компьютере с фиксиро-

* Здесь и в дальнейшем в число арифметических операций включена операция сравнения $\leqslant$ двух чисел

** Как увидит читатель в § 2, случай фиксированного числа ограничений тривиален.

ванной разрядной сеткой. В этом случае неизбежны ошибки округлений и/или эффекты переполнения. Это приводит к тому, что методы, описанные в терминах точных операций, требуют для своей реализации большой доводки: теоретического и экспериментального анализа ошибок округления, возможности роста данных и переполнения, численной устойчивости. Такую доводку удается осуществить далеко не всегда, и насколько это непростое дело уже для систем линейных уравнений, можно узнать из книг по вычислительной линейной алгебре (см., в частности, поучительное введение в книгу В. В. Воеводина [4]), мы же ограничимся следующим простым примером [5]: «если мы решаем систему

$$\begin{aligned} x_1 &= b_1, \\ 2x_1+x_2 &= b_2, \\ 2x_2+x_3 &= b_3, \\ &\ldots \\ 2x_{n-1}+x_n &= b_n \end{aligned} \qquad (5)$$

100-го порядка с 12 десятичными знаками, то одно лишь округление правой части может привести к ошибке, превосходящей само решение в 10^{15} раз... Самое неприятное заключается в том, что внешне такие системы ничем не примечательны...» Таким образом, описания методов в точных операциях и оценки числа таких операций приходится в дальнейшем дополнять анализом необходимого в расчетах числа знаков, причем ясно, что с увеличением цифровой длины чисел и требований к точности увеличивается и объем вычислительной работы.

Для метода, описанного в терминах точных арифметических операций, имеется, правда, возможность и в самом деле выполнять эти операции точно — например, используя в качестве входных данных рациональные числа, над которыми все арифметические операции можно выполнять без ошибок округлений по обычным правилам действий с дробями. Однако в любой системе записи при точном выполнении арифметических операций над ненулевыми вещественными числами возможен неограниченный рост их цифровой длины.

Пример 1. Заменим в системе (5) двойки на целые числа, имеющие в двоичной (или десятичной) записи l знаков, и пусть целые числа $b_1, \ldots, b_n$ также l-значны. Если решать систему в рациональных числах, то числители и знаменатели получающихся в ответе чисел могут иметь в своей записи nl знаков. Можно показать, что несколько большего числа знаков $\min(n,m)\cdot[l+\log\min(n, m)]$ хватит для решения любой системы линейных уравнений (4) с целыми l-значными входными коэффициентами. В дальнейшем будем для определенности считать запись двоичной, а под log понимать двоичный логарифм. Величина $\log\min(n, m)$ растет с увеличением размерности медленно и при $n=m=1000$ не превосходит 10. Таким образом, этот пример показывает, что при точном решении систем линейных уравнений с l-значными коэффициентами возникает необходимость работать, грубо говоря, с $\min(n, m)$ l-значными числами. Существуют специальные приемы (см., в частности, [28]), позволяющие проводить большую часть этой работы с существенно более короткими числами и лишь в конце восстанавливать точный ответ. Техника работы с длинными числами, включая способы быстрого выполнения арифметических операций, в последние десятилетия интенсивно развивается.

Пример 2. Рассмотрим записанную слева от вертикальной черты вычислительную процедуру, формирующую последовательность чисел $\beta_1, \beta_2, \ldots$, сходящуюся к квадратному корню из числа α. Справа записан результат выполнения этой процедуры при $\alpha=2$.

Вход: число α	Вход: 2
Вычислить:	
$\beta_1=(1+\alpha/1)/2$,	$\beta_1=3/2$,
$\beta_2=(\beta_1+\alpha/\beta_1)/2$;	$\beta_2=17/12$,
$\beta_3=(\beta_2+\alpha/\beta_2)/2$,	$\beta_3=577/408$,
$\beta_4=(\beta_3+\alpha/\beta_3)/2$,	$\beta_4=665857/470832$,
. . . . .	. . . . .
$\beta_{k+1}=(\beta_k+\alpha/\beta_k)/2$,	β_{k+1} = число, имеющее более 2^k цифр.

(6)

С точки зрения алгебраического подхода сложность каждого шага этой процедуры постоянна — требуется выполнить три арифметические операции, и поэтому алгебраическая сложность процедуры линейна по числу шагов k. Однако с каждым шагом цифровая длина чисел увеличивается примерно вдвое, и объем вычислительной работы в данной процедуре считать линейным или даже полиномиальным по k нереалистично.

Таким образом, и при точном, и при приближенном выполнении операций алгебраические оценки сложности требуют существенной доводки — получения информации о том, с числами какой цифровой длины эти операции будут выполняться. Подход к оценкам сложности вычислений, требующий для оценки трудоемкости того или иного метода указывать не только количество необходимых ему операций, но и цифровую длину чисел, над которыми они выполняются, называется битовым (цифровым). Этот подход, следовательно, рассматривает все встречающиеся в вычислениях числа как конечные записи из цифр, которые подвергаются в вычислительной машине потактовой обработке поразрядно или порциями длиной в машинное слово. Тактовая стоимость (время) такой обработки при выполнении арифметических операций зависит, естественно, от длины чисел и называется битовой сложностью операции. Для разных операций и алгоритмов их исполнения она меняется в пределах от линейной до квадратичной зависимости от длины чисел-операндов. Так, существуют алгоритмы выполнения всех арифметических операций битовой сложности $\mathrm{const}\, l\cdot\log l\times$ $\times\log\log l$, т. е. стоимости, почти линейной по количеству l цифр. В любом случае битовая сложность арифметических операций

полиномиально ограничена сверху и снизу цифровой длиной чисел.

Пример 3. Битовая сложность процедуры (6), равная суммарной сложности входящих в нее операций, экспоненциально растет с увеличением числа шагов k.

2.2. Полиномиальная разрешимость линейного программирования

Рассмотрим теперь битовую постановку вопроса существования полиномиального алгоритма линейного программирования. Практически во всех приложениях входные коэффициенты задач линейного программирования являются рациональными числами и, значит, после приведения неравенств к общему знаменателю могут быть сделаны целыми. Поэтому будем считать, что входные коэффициенты таблицы (3) с самого начала являются целыми числами, заданными в двоичной позиционной записи. Обозначим через l число разрядов, отводимых на запись одного такого числа, т. е. число знаков во входных коэффициентах. Ясно, что битовая сложность решения задачи (1)—(2) должна зависеть не только от ее размерности (n, m), но и от величины l разрядности коэффициентов, поскольку, например, для решения простейшей задачи $x_1 \leqslant b$, $x_2 \leqslant ax_1$, $x_2 \to \max$ приходится умножать l-разрядные числа. В связи с этим естественно назвать битовой размерностью задачи уже не пару (n, m), а тройку размеров (n, m, l) и поставить вопрос о полиномиальном алгоритме следующим образом: можно ли построить метод решения общей задачи линейного программирования (1)—(2) с целыми коэффициентами, имеющий полиномиальную по (n, m, l) битовую сложность? С содержательной точки зрения этот вопрос близок к поставленному в пункте 1.4, поскольку для большинства приложений количество цифр во входных коэффициентах не превышает одного-двух десятков. Однако, как было показано в 1979 г., битовый вариант вопроса уже имеет положительное решение [15]. Как увидит читатель в третьей главе, с помощью битовой доводки предложенного Д. Б. Юдиным и А. С. Немировским [20] метода эллипсоидов и некоторого простого вспомогательного алгоритма «округления» находимого с помощью этого метода приближенного решения может быть получен следующий результат: для точного решения общей задачи линейного программирования (1)—(2) с целыми коэффициентами достаточно выполнить не более $\text{const} \max(n, m) \min^4(n, m)[l + \log \min(n, m)]$ арифметических операций над $\text{const} \min(n, m)[l + \log \min(n, m)]$-разрядными числами [16].

Следует заметить, что параметр l теперь входит в оценку числа операций, которая оказывается полиномиально-ограниченной битовой, а не обычной размерностью задачи. Кроме того, цифровая длина чисел по порядку не превосходит количества, необходимого для записи значений неизвестных в точном решении систем линейных равенств примера 1 (а значит, и систем неравенств той же битовой размерности).

Итак, в битовом смысле линейное программирование является полиномиально-разрешимым. Теоретико-сложностная интерпретация этого результата такова: если обозначить через L длину входа задачи линейного программирования, т. е. полное количество цифр в записи ее входной информации (3), то вычислительная машина способна точно решить любую задачу за полиномиальное по L время.

Разумеется, на практике чаще всего ищется не точное, а лишь приближенное решение задачи (1)—(2), например, имеющее в каждой компоненте не более l знаков перед запятой, достигающее максимум (2) с точностью l знаков после запятой и удовлетворяющее всем ограничениям (1) также с точностью l знаков после запятой. В этом случае методу эллипсоидов нет необходимости работать с такими же длинными числами, как при точном решении,— можно показать, что для отыскания приближенного решения все операции можно также выполнять приближенно с числами, имеющими до и после запятой не более $\text{const}[l + \log \min(n, m)]$ знаков (для задач линейного программирования $\text{const} < 8$, так что, грубо говоря, хватает учетверенной цифровой точности ответа). Число арифметических операций метода эллипсоидов при заданной цифровой точности l также снижается до оценки $\text{const} \max(n, m) \min^3(n, m) \times [l + \log \min(n, m)]$ четвертой, а не пятой, как при точном решении, степени размерности (n, m). Однако, как уже говорилось, число операций симплекс-метода не зависит от l и в большинстве случаев лишь кубично по размерности, и поэтому метод эллипсоидов оказывается с ним неконкурентоспособным в практическом использовании, где плохие для симплекс-метода задачи встречаются редко. В этом месте начинается самая интересная часть истории о полиномиальных алгоритмах линейного программирования, которая продолжается в наши дни. В течение нескольких лет оказалось возможным предложить сразу несколько новых полиномиальных алгоритмов линейного программирования, и один из них, изобретенный в 1984 г. Н. Кармаркаром [39], по ряду сообщений не только может конкурировать с симплекс-методом, но и позволяет решать ранее неприступные задачи. Причины высокой эффективности метода Кармаркара к настоящему времени не получили полного теоретического

объяснения, поскольку найденные для него до сих пор оценки числа операций при заданной цифровой точности приближенного решения $\text{const}\max^{3,5}(n, m)[l+\log\max(n, m)]$ растут как степень 3,5 размерности, т. е. улучшают оценки метода эллипсоидов лишь в корень из размерности раз. Возможно, однако, что эти оценки могут быть улучшены. Дело в том, что, хотя имеющаяся теория ограничивает число итераций (средней стоимости $\max^{2,5}(n, m)$ операций каждая) метода Кармаркара первой степенью размерности, имеются сообщения, что даже для решения больших задач хватает лишь нескольких десятков итераций. Наконец, следует отметить совсем недавний важный результат Дж. Ренегара, показавшего, что для решения задачи линейного программирования достаточно решить некоторое количество систем линейных уравнений (методом Гаусса за $\max(n, m)\min^2(n, m)$ операций каждую), причем указанное количество систем ограничено при заданной цифровой точности приближенного решения лишь квадратным корнем $\text{const}\max^{0,5}(n, m)[l+\log\min(n, m)]$ размерности*. В общем, можно сказать, что алгоритмика линейного программирования находится в настоящее время на очень интересном этапе своего развития.

ГЛАВА 2. СИМПЛЕКС-МЕТОД

§ 1 Принцип граничных решений

Решение общей задачи линейного программирования

$$cx=c_1x_1+\ldots+c_nx_n\to\max, \qquad (1.1)$$

$$a_ix=a_{i1}x_1+\ldots+a_{in}x_n\leqslant b_i \qquad (1.2)$$

$$i\in\mathbf{M}=\{1, 2, \ldots, m\}$$

с помощью симплекс-метода основывается на использовании так называемого принципа граничных решений**. Для его формулировки выберем в множестве M неравенств задачи некоторое подмножество ограничений $M\subseteq\mathbf{M}$ и рассмотрим систему линейных уравнений $\text{PAB}(M)$

$$a_ix=a_{i1}x_1+\ldots+a_{in}x_n=b_i,\ i\in M, \qquad (1.3)$$

которая получается из задачи отбрасыванием целевой функции и всех неравенств $i\in\mathbf{M}-M$, не включенных в M, с последующей заменой для оставшихся неравенств $i\in M$ знаков $\leqslant$ на $=$. Принцип граничных решений утверждает, что если задача линейного программирования (1.1)—(1.2) разрешима, т. е. имеет хотя бы одно решение, то всегда существует такое $M\subseteq\mathbf{M}$, что в качестве решения задачи линейного программирования (1.1)—(1.2) можно взять любое решение разрешимой системы линейных уравнений $\text{PAB}(M)$.

Доказательство. Нужно показать, что если множество X^* решений задачи линейного программирования не пусто, то для некоторого M из $\mathbf{M}$ множество $X(M)$ решений системы $\text{PAB}(M)$ также не пусто и целиком содержится в X^*. Опишем итеративную процедуру построения требуемого множества M, исходя из того, что задано некоторое решение $x^0\in X^*$ задачи линейного программирования. Подставим x^0 в систему линейных неравенств (1.2). Поскольку x^0 удовлетворяет (1.2), множество $\mathbf{M}$ ограничений этой системы разобьется на два подмножества

$$\begin{aligned} &a_ix^0=b_i,\ i\in M(x^0),\\ &a_ix^0<b_i,\ i\in\mathbf{M}-M(x^0),\end{aligned} \qquad (1.4)$$

в первое из которых, обозначенное $M(x^0)$, войдут «жесткие» в точке x^0 ограничения, выполненные как равенства, а во второе $\mathbf{M}-M(x^0)$ — все оставшиеся ограничения, выполняемые в точке x^0 «с запасом», как строгие неравенства. Положим $M^0=M(x^0)$ и рассмотрим систему линейных уравнений $\text{PAB}(M^0)$. Из первой строки (1.4) видно, что система $\text{PAB}(M^0)$ разрешима — множество $X(M^0)$ ее решений содержит точку x^0. Если $X(M^0)\subseteq X^*$, принцип граничных решений доказан при $M=M^0$, в противном случае найдется решение $\tilde{x}\in X(M^0)$ системы $\text{PAB}(M^0)$

$$a_i\tilde{x}=b_i,\ i\in M^0,$$

которое не является решением задачи линейного программирования (1.1)—(1.2). Покажем прежде всего, что значения целевой функции в точках x^0 и $\tilde{x}$ совпадают: $cx^0=c\tilde{x}$. Действительно, множество $X(M^0)$ решений системы линейных уравнений содержит вместе с точками x^0 и $\tilde{x}$ соединяющую их прямую L, составленную из точек вида $x(t)=x^0+(\tilde{x}-x^0)t$ при всевозможных значениях скалярного параметра t:

$$a_ix(t)\equiv b_i,\ i\in M^0,\ t\in(-\infty, +\infty). \qquad (1.5)$$

При достаточно малых по модулю значениях параметра строгие неравенства второй строки (1.4) продолжают выполняться

$$a_ix(t)<b_i,\ i\in\mathbf{M}-M^0,\ t\in(-\varepsilon, \varepsilon),$$

так что в некотором интервале ненулевой длины 2ε точка $x(t)$ удовлетворяет всем ограничениям (1.2) исходной задачи линейного программирования. Поскольку точка x^0 доставляет максимум целевой функции при ограничениях (1.2), то

$$cx(t)=cx^0+[c\tilde{x}-cx^0]t\leqslant cx^0,\ t\in(-\varepsilon, \varepsilon),$$

откуда следует, что выражение в квадратных скобках должно быть равно нулю. Итак, $cx^0=c\tilde{x}$, следовательно, целевая функция постоянна на всей прямой

$$cx(t)\equiv cx^0,\ t\in(-\infty, +\infty). \qquad (1.6)$$

Представим теперь зависящую от времени $t\in[0, 1]$ точку $x(t)$, движущуюся по прямой L из начального положения $x(0)=x^0$ в конечное положение $x(1)=\tilde{x}$. Как видно из (1.6) и (1.5), при таком движении целевая функция все время постоянна, а ограничения M^0 выполнены как равенства. Поскольку $x(0)$ является решением зада-

---

* К сожалению, автор смог ознакомиться с препринтом Дж. Ренегара [45] слишком поздно, для того чтобы включить в брошюру более подробное изложение указанного результата.

** Здесь и в дальнейшем считаем, что в каждой строке a_i имеется хотя бы один ненулевой коэффициент a_{ij}.

чи линейного программирования, а $x(1)$ нет, где-то на пути какие-то из неравенств

$$a_i x(t) \leqslant b_i,\ i \in \mathbf{M} - M^0, \qquad (1.7)$$

выполненные в начале пути строго, удовлетворяться перестанут. Пусть τ — последний момент времени, когда все неравенства (1.7) еще выполняются, тогда $x(\tau)$ — решение задачи линейного программирования, для которого выполнены как равенства не только ограничения M^0, но еще хотя бы одно неравенство (1.7) из $\mathbf{M} - M^0$. Другими словами, предположив, что система $\mathrm{PAB}(M(x^0))$ не удовлетворяет принципу граничных решений, мы построили по решению задачи линейного программирования x^0 новое решение той же задачи $x^1 = x(\tau)$, для которого множество жестких ограничений строго включает в себя множество жестких ограничений исходного решения $M(x^0) \subset M(x^1)$ и так далее

$$M(x^0) \subset M(x^1) \subset M(x^2) \subset \ldots \subset M(x^k). \qquad (1.8)$$

Ясно, что через некоторое конечное число шагов k рост цепочки (1.8) должен остановиться, и в этот момент система линейных уравнений $\mathrm{PAB}(M)$ при $M = M(x^k)$ уже будет удовлетворять принципу граничных решений [18].

Важным следствием принципа граничных решений является возможность свести решение любой задачи линейного программирования к решению конечного числа систем линейных уравнений $\mathrm{PAB}(M)$ и, следовательно, возможность решения любой задачи линейного программирования с помощью конечного числа арифметических операций. Действительно, если задача линейного программирования (1.1)—(1.2) разрешима, ее решение найдет следующий

Алгоритм полного перебора. Рассматривая по очереди всевозможные подмножества M конечного множества ограничений $\mathbf{M}$, находим по одному решению на каждую разрешимую систему линейных уравнений $\mathrm{PAB}(M)$, а затем в конечном множестве получившихся решений выбираем среди допустимых по ограничениям (1.2) решений то, в котором значение целевой функции (1,1) максимально.

Хотя алгоритм перебора позволяет с принципиальной точки зрения решить любую задачу линейного программирования за конечное число арифметических операций, его ценность в практическом отношении близка к нулю: уже при решении задач линейного программирования с 30 неизвестными и 60 ограничениями такой астрономический перебор осуществить невозможно. В § 3 описываются основные идеи симплекс-метода решения задач линейного программирования, позволяющего в большинстве практических случаев существенно сократить число систем $\mathrm{PAB}(M)$, подлежащих перебору.

§ 2 Базисы и базисные решения

Для описания симплекс-метода удобно наложить на задачу линейного программирования (1.1)—(1.2) одно условие, которое позволит перебирать в принципе граничных решений лишь такие системы линейных уравнений $\mathrm{PAB}(M)$, которые содержат в точности n уравнений и имеют единственное решение. Для формулировки этого условия, называемого условием полноты ранга матрицы ограничений, потребуется известное из линейной алгебры понятие линейной зависимости векторов.

Вектор $a = (a_1, a_2, \ldots, a_n)$ называется линейно-зависимым от векторов $a_i = (a_{i1}, a_{i2}, \ldots, a_{in})$, $i \in M$, если найдутся такие числа λ_i, $i \in M$, что

$$a = \sum_{i \in M} \lambda_i a_i. \qquad (2.1)$$

Рассмотрим, как и в предыдущем параграфе, в качестве векторов a_i строки матрицы A ограничений задачи линейного программирования

$$A = \begin{bmatrix} a_{11}\ a_{12}\ \ldots\ a_{1n} \\ a_{21}\ a_{22}\ \ldots\ a_{2n} \\ \ldots\ \ldots\ \ldots\ \ldots \\ a_{m1}\ a_{m2}\ \ldots\ a_{mn} \end{bmatrix} = \begin{bmatrix} a_1 \\ a_2 \\ \ldots \\ a_m \end{bmatrix}. \qquad (2.2)$$

Минимальная система векторов a_i, $i \in M$, от которых линейно зависят все оставшиеся векторы a_ν, $\nu \in \mathbf{M} - M$, называются строчным базисом A. Поскольку число $|M|$ векторов в строчном базисе, называемое рангом матрицы A, не превышает числа координат, всегда $|M| \leqslant n$. Для описания симплекс-метода удобно считать, что ранг матрицы ограничений в точности равен n, т. е. найдется строчный базис с числом векторов, равным числу неизвестных $|M| = n$. Покажем, что это условие полноты ранга всегда может быть легко получено с помощью простого преобразования исходной задачи линейного программирования (1.1)—(1.2).

Прежде всего с помощью метода Гаусса, затратив кубическое по размерности задачи количество $\max(n, m)\min^2(n, m)$ арифметических операций, можно найти какой-нибудь строчный базис a_i, $i \in M$ матрицы ограничений, а также числа $\lambda_{\nu i}$, с помощью которых все остальные векторы a_ν, $\nu \in \mathbf{M} - M$ линейно выражаются через базисные:

$$a_\nu = \sum_{i \in M} \lambda_{\nu i} a_i. \qquad (2.3)$$

Кроме того, с теми же затратами арифметических операций можно проверить, выражается ли и вектор c целевой функции задачи в виде линейной комбинации векторов базиса:

$$c = \sum_{i \in M} \lambda_i a_i. \qquad (2.4)$$

Если выражение (2.4) получить невозможно, то из принципа граничных решений следует, что задача линейного программирования (1.1)—(1.2) является неразрешимой. Если же выражение (2.4) будет найдено, введем $|M|$ новых неизвестных

$$x_i' = a_i x, \quad i \in M, \tag{2.5}$$

с помощью которых перепишем задачу линейного программирования (1.1)—(1.2) в следующем виде:

$$cx = \sum_{i \in M} \lambda_i a_i x \equiv \sum_{i \in M} \lambda_i x_i' \to \max, \tag{1.1'}$$

$$a_i x \equiv x_i' \leqslant b_i, \quad i \in M, \tag{1.2'}$$

$$a_\nu x = \sum_{i \in M} \lambda_{\nu i} a_i x \equiv \sum_{i \in M} \lambda_{\nu i} x_i' \leqslant b_\nu, \quad \nu \in \mathbf{M} - M.$$

Рассмотрим теперь отдельно задачу линейного программирования в штрихованных неизвестных x_i', которая получится, если убрать все записанное слева от знаков тождества $\equiv$. Эта задача, обозначенная (1.1')—(1.2'), эквивалентна исходной задаче линейного программирования (1.1)—(1.2): поскольку векторы a_i, $i \in M$ базиса являются линейно-независимыми — ни один из них не может быть линейно выражен через другие,— система линейных уравнений (2.5) разрешима при любых заданных значениях левых частей. Другими словами, по любому решению штрихованной задачи можно найти (неоднозначно при $|M| < n$) решение исходной.

Будем теперь решать вместо исходной штрихованную задачу линейного программирования (1.1')—(1.2'). В этой задаче имеется $n' = |M| =$ ранг A неизвестных — не больше, чем в исходной задаче, $n' \leqslant n$. Количество m ограничений в штрихованной задаче такое же, что и в исходной. В то же время n' из m ограничений штрихованной задачи имеют простейший вид $x_i' \leqslant b_i$, $i \in M$, из которого следует, что в штрихованной задаче условие полноты ранга матрицы ограничений A' заведомо выполнено. Действительно, в качестве строчного базиса A' можно взять n' строк с номерами $i \in M$.

Итак, показано, что по любой задаче линейного программирования (1.1)—(1.2) можно достаточно просто построить эквивалентную задачу не большей размерности, для которой выполнено условие полноты ранга матрицы ограничений. В связи с этим до конца § 4 будем без ограничения общности считать, что условие полноты ранга выполнено для исходной задачи линейного программирования (1.1)—(1.2). Отсюда, в частности, следует, что $n \leqslant m$.

Пусть M — некоторое подмножество ограничений задачи линейного программирования (1.1)—(1.2), состоящее в точности из n элементов, $|M| = n$. Назовем множество M базисным, если соответствующая система векторов a_i, $i \in M$ является строчным базисом матрицы (2.2). Из линейной алгебры известно, что базисность M эквивалентна любому из следующих шести утверждений:

1. Система векторов a_i, $i \in M$ является линейно-независимой, т. е. никакой из векторов системы не выражается линейно через другие.

2. Любой n-мерный вектор a может быть единственным образом представлен в виде линейной комбинации (2.1) векторов базиса.

3. Квадратная $n \times n$ матрица A_M, полученная вычеркиванием из матрицы (2.2) всех не включенных в множество M строк, имеет ненулевой определитель $\det A_M \neq 0$.

4. Существует $n \times n$ матрица A_M^{-1}, обратная матрице A_M

$$A_M^{-1} A_M = A_M A_M^{-1} = I.$$

Здесь I — единичная матрица размера $n \times n$, все недиагональные элементы которой равны нулю, а по диагонали записаны единицы.

5. Система линейных уравнений (2.5) имеет при любых заданных левых частях x' единственное решение x. Это решение находится умножением вектора x' на обратную матрицу $x - A_M^{-1} x'$.

6. Система РАВ(M) имеет единственное решение.

Утверждение 6 позволяет дать следующую геометрическую интерпретацию базисности M. Рассмотрим неизвестные $x_1, \ldots, x_n$ задачи линейного программирования как декартовы координаты точки x в пространстве n измерений. Линейное уравнение $a_{i1} x_1 + \ldots + a_{in} x_n = b_i$ определяет тогда в n-мерном пространстве некоторую плоскость Π_i, составленную из точек x с координатами, удовлетворяющими уравнению. Множество $X(M)$ решений системы линейных уравнений РАВ(M) представляет собой пересечение плоскостей Π_i, $i \in M$. Согласно 6 базисность M эквивалентна тому, что n включенных в M плоскостей Π_i, $i \in M$ пересекаются в единственной точке $x_M = X(M)$. Эта точка — единственное решение системы линейных уравнений РАВ(M) — называется базисным решением.

Пример. Рассмотрим задачу линейного программирования с $n = 3$ неизвестными и $m = 6$ ограничениями вида

$$\begin{aligned} -x_1 &\leqslant 0, && (1)\\ -x_2 &\leqslant 0, && (2)\\ -x_3 &\leqslant 0, && (3)\\ x_1 + x_2 + x_3 &\leqslant 4, && (4)\\ 2x_1 &\leqslant 3, && (5)\\ 3x_2 &\leqslant 8. && (6) \end{aligned}$$

На рис. 1 показаны 6 плоскостей Π_i, соответствующих ограничениям (1)—(6). Например, плоскость Π_2 задается уравнением $x_2 = 0$, плос-

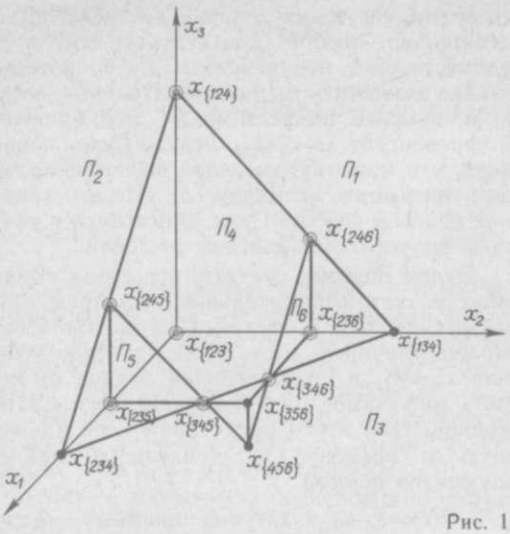


Рис. 1.

кость Π_4 — уравнением $x_1+x_2+x_3=4$ и т. д. Из $m=6$ ограничений можно составить $C_m^n=m(m-1)...(m-n+1)/n(n-1)...1=20$ различных множеств M, содержащих $n=3$ элемента, а именно:

$\{1, 2, 3\}$, $\{1, 2, 4\}$, $\{1, 2, 5\}^*$, $\{1, 2, 6\}^*$, $\{1, 3, 4\}$, $\{1, 3, 5\}^*$, $\{1, 3, 6\}$, $\{1, 4, 5\}^*$, $\{1, 4, 6\}$, $\{1, 5, 6\}^*$, $\{2, 3, 4\}$, $\{2, 3, 5\}$, $\{2, 3, 6\}^*$, $\{2, 4, 5\}$, $\{2, 4, 6\}^*$, $\{2, 5, 6\}^*$, $\{3, 4, 5\}$, $\{3, 4, 6\}$, $\{3, 5, 6\}$, $\{4, 5, 6\}$.

Из этих 20 множеств 12 являются базисными — соответствующие им базисные решения показаны на рисунке жирными точками. Например, базисному множеству $M=\{3, 4, 5\}$ соответствует базисное решение $x_{\{345\}}=\left(\frac{3}{2}, \frac{5}{2}, 0\right)$, компоненты которого можно найти из системы линейных уравнений $\mathrm{PAB}\{3, 4, 5\}$ $-x_3=0$, $x_1+x_2+x_3=4$, $2x_1=3$.

Оставшиеся 8 множеств, отмеченных в списке звездочками, не являются базисными по той причине, что плоскость Π_1 параллельна плоскости Π_5, а плоскость Π_2—Π_6.

Вернемся к рассмотрению общих задач линейного программирования. Как уже указывалось, можно без ограничения общности считать, что в задаче существует некоторое базисное множество ограничений. В этом предположении читатель без труда докажет* следующее уточнение принципа граничных решений.

Если задача линейного программирования (1.1)—(1.2) является разрешимой, то среди систем $\mathrm{PAB}(M)$, удовлетворяющих принципу граничных решений, существует базисная. В частности, если система линейных неравенств (1.2) разрешима, то среди ее решений найдется хотя бы одно базисное (такие базисные решения, удовлетворяющие всем неравенствам (1.2), называются допустимыми).

* или подождет, пока доказательство даст симплекс-метод.

В рассмотренном примере из 12 базисных решений допустимыми являются 8, а недопустимыми — базисные решения $x_{\{134\}}$, $x_{\{234\}}$, $x_{\{356\}}$, $x_{\{456\}}$, нарушающие соответственно неравенства (6), (5), (4) и (3). Таким образом, максимум произвольной линейной функции $c_1x_1+c_2x_2+c_3x_3$ при ограничениях (1)—(6) может быть достигнут в одной из 8 допустимых базисных точек, обведенных на рис. 1 красными кружками.

§ 3 Симплекс-метод

На рис. 1 среди 12 отмеченных точками базисных решений нет совпадающих. Другими словами, различным базисным множествам M ограничений соответствуют различные базисные решения x_M. Задачи линейного программирования, для которых выполнено это условие, называются невырожденными.

Может, однако, случиться так, что в задаче линейного программирования различным базисным множествам ограничений будет соответствовать одно и то же базисное решение, и тогда говорят, что в задаче имеется вырождение. Например, если в ограничении (6) рассмотренного в предыдущем параграфе примера заменить неравенство $3x_2\leqslant 8$ на неравенство $3x_2\leqslant 7{,}5$, то плоскость Π_6 переместится влево и пройдет через «чужую» базисную точку $x_{\{345\}}=\left(\frac{3}{2}, \frac{5}{2}, 0\right)$ (см. рис. 1). В результате 4 различных (в исходном примере) базисных решения выродятся в одно:

$$x_{\{345\}}=x_{\{346\}}=x_{\{356\}}=x_{\{456\}}. \qquad (3.1)$$

Стоит, однако, возмутить неравенство $3x_2\leqslant 7{,}5$ добавлением к его правой части сколь угодно малого числа ε, как плоскость Π_6 чуть-чуть сдвинется, вырождение снимется и базисные решения (3.1) опять разойдутся.

Хотя симплекс-метод может быть описан таким образом, что он будет давать решение произвольных задач линейного программирования с учетом возможности вырождения, для невырожденных задач линейного программирования описание метода выглядит гораздо проще. Поэтому первоначальные варианты симплекс-метода исходили из предположения невырожденности, и лишь затем были разработаны модификации метода, пригодные для общих задач. Следуя этому пути, при описании основных идей симплекс-метода будем считать задачу линейного программирования (1.1)—(1.2) невырожденной. Это означает, что решение x_M любой базисной системы линейных уравнений $\mathrm{PAB}(M)$ не может обращать в равенство никакое «чужое» ограничение $\nu\in\mathbf{M}-M$.

Работа симплекс-метода состоит из двух этапов, на первом из которых ищется допустимое для системы (1.2) базисное решение. Результатом этого этапа является либо получение некоторого допустимого базисного решения x_M, либо доказательство его отсутствия. Последнее означает неразрешимость

ограничений задачи, и тогда метод останавливается. Если некоторое x_M находится, начинается второй этап работы метода, состоящий из однотипных шагов, на каждом из которых текущее допустимое базисное решение заменяется новым с возросшим значением целевой функции. Эта замена осуществляется специальной симплекс-процедурой S улучшения допустимых базисных решений. Через конечное число шагов процедура S либо останавливается на оптимальном базисном решении, либо выясняет, что целевая функция задачи может принимать при ее ограничениях сколь угодно большие значения.

Описание метода удобно начать с процедуры S улучшения допустимого базисного решения x_M. С аналитической точки зрения идею процедуры проще всего проследить с помощью перехода (2.5) к штрихованным координатам, в которых базисные неравенства $a_i x \leqslant b_i$, $i \in M$ принимают простейший вид $x'_i \leqslant b_i$. Рассмотрим этот переход более подробно. Пусть A_M — составленная из строк $i \in M$ квадратная $n \times n$ базисная матрица ограничений (см. утверждения 3—5 §2), тогда замена (2.5) в матричной записи имеет вид $x' = A_M x$. Как следует из утверждения 5 §2, переменные x можно выразить через штрихованные переменные с помощью обратной матрицы:

$$x = A_M^{-1} x'. \tag{3.2}$$

Напишем аналогично §2 задачу линейного программирования в штрихованных переменных

$$cx = [cA_M^{-1}]x' \equiv \lambda x' \to \max, \tag{1.1'}$$

$$a_i x = [a_i A_M^{-1}]x' \equiv x'_i \leqslant b_i, \ i \in M, \tag{1.2'}$$

$$a_\nu x = [a_\nu A_M^{-1}]x' \equiv \lambda_\nu x' \leqslant b_\nu, \ \nu \in \mathbf{M} - M,$$

причем на этот раз получены явные выражения $\lambda = cA_M^{-1}$, $\lambda_\nu = a_\nu A_M^{-1}$ для векторов целевой функции и ограничений этой задачи. Отметим, что если подставить в (1.2′) допустимый базисный вектор x_M, то получится

$$\begin{aligned} a_i x_M &= (x'_M)_i = b_i, \ i \in M, \\ a_\nu x_M &= \lambda_\nu x'_M < b_\nu, \ \nu \in \mathbf{M} - M, \end{aligned} \tag{3.3}$$

поскольку x_M обращает в равенстве базисные ограничения и в силу невырожденности задачи в строгие неравенства небазисные ограничения. В частности, при замене (2.5) вектору x_M соответствует в штрихованной задаче допустимый базисный вектор x'_M, i-я компонента которого есть b_i, $i \in M$. Коротко это можно записать так: $x'_M = b_M$.

Покажем прежде всего, что если все компоненты вектора λ целевой функции в штрихованной задаче неотрицательны

$$\lambda_i = (cA_M^{-1})_i \geqslant 0, \ i \in M, \tag{3.4}$$

то оптимум в обеих задачах достигнут. Действительно, любой допустимый вектор x' удовлетворяет неравенствам $x'_i \leqslant b_i$, которые можно умножить на неотрицательные числа λ_i и сложить по всем $i \in M$. Это приведет к неравенству $\lambda x' \leqslant \lambda b_M$, откуда будет следовать, что максимум целевой функции не может превышать величину λb_M. Но это значение целевой функции уже достигнуто в текущем допустимом базисном решении.

Будем поэтому считать, что среди компонент λ есть отрицательные. Выберем одну из них, скажем $\lambda_p < 0$, $p \in M$, и покажем, как можно улучшить допустимое базисное решение $x'_M = b_M$ в штрихованной задаче и, значит, допустимое базисное решение x_M в исходной. Для этого представим себе зависящую от времени $t \geqslant 0$ точку $x'(t)$, движущуюся по прямой

$$x'_p(t) = b_p - t \ \text{ и } \ x'_i(t) \equiv b_i \ \text{ при } \ i \in M - p.$$

В начальный момент времени точка находится в допустимом базисном решении $x'(0) = b_M$, а затем у нее начинает убывать p-я компонента при неизменных других (рис. 2). Это приводит к тому, что целевая функция во время движения точки $x'(t)$ строго растет. Во время движения точки неравенства $i \in M - p$ выполнены как равенства, а неравенство p при положительных t становится строгим: $x'_p(t) < b_p$. Таким образом, допустимость движущейся точки $x'(t)$ по ограничениям (1.2) задачи эквивалентна ее допустимости по ограничениям $\nu \in \mathbf{M} - M$, которые принимают вид $\lambda_\nu x'_M - t\lambda_{\nu p} \leqslant b_\nu$, где $\lambda_{\nu p}$ есть p-я компонента вектора λ_ν. Из (3.3) видно, что положительная величина

$$\tau_\nu = \begin{cases} +\infty, & \text{если } \lambda_{\nu p} \geqslant 0, \\ (\lambda_\nu x'_M - b_\nu)/\lambda_{\nu p}, & \text{если } \lambda_{\nu p} < 0 \end{cases}$$

задает момент времени, до которого выполняется неравенство ν, и поэтому все неравенства будут выполнены вплоть до момента

$$\tau = \min \tau_\nu, \ \nu \in \mathbf{M} - M. \tag{3.5}$$

Заметим, что если $\tau = +\infty$, можно неограниченно увеличивать целевую функцию, не нарушая ограничений задачи. Будем поэтому считать величину τ конечной, и пусть минимум (3.5) достигается на величине τ_q, т. е. при движении первым по времени нарушается неравенство q (см. рис. 2). Тогда ясно, что вектор $x'(\tau)$ является допустимым базисным решением для множества ограничений $S(M) = q \cup (M - p)$, про которое говорят, что оно получено выводом из исходного базиса неравенства p и вводом неравенства q. Из базисности $S(M)$, кстати, следует, что для невырожденной задачи минимум (3.5) достигается на единственном q. Далее, поскольку $\tau > 0$, допустимому базисному множеству $S(M)$ соответствует в исходной задаче допу-

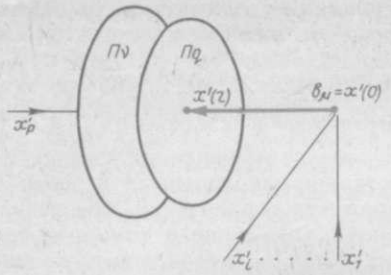


Рис. 2

стимое базисное решение $x_{S(M)}$ с возросшим по сравнению с x_M значением целевой функции, т. е. $S(M)$ лучше M. Теперь, исключая из рассмотрения штрихованные переменные, можно дать следующее описание.

Симплекс-процедура S улучшения допустимого базисного множества неравенств M.

Вход — допустимое базисное множество неравенств M:

1) составить базисную матрицу A_M и вычислить обратную к ней A_M^{-1};

2) вычислить допустимое базисное решение $x_M = A_M^{-1} b_M$;

3) вычислить вектор cA_M^{-1} и проверить, все ли его компоненты $(cA_M^{-1})_i$ неотрицательны, $i \in M$. Если это так, закончить процедуру — множество M и базисное решение x_M оптимальны;

4) выбрать среди компонент вектора cA_M^{-1} некоторую отрицательную $p \in M$;

5) для $\nu \in \mathbf{M} - M$ вычислить величины

$$\tau_\nu = \begin{cases} +\infty, & \text{если } (a_\nu A_M^{-1})_p \geqslant 0, \\ (a_\nu x_M - b_\nu)/(a_\nu A_M^{-1})_p, & \text{если } (a_\nu A_M^{-1})_p < 0. \end{cases}$$

Если все τ_ν бесконечны, закончить процедуру — целевая функция может достигать сколь угодно больших значений, и задача неразрешима;

6) найти q, для которого число τ_q минимально среди чисел τ_ν, $\nu \in \mathbf{M} - M$. Заменить в M ограничение p на q: $p \leftarrow q$.

Для полноты описания процедуры следует еще остановиться на двух моментах. Во-первых, для ее однозначного выполнения в пункте 4 описания необходимо конкретизировать правило выбора отрицательной компоненты p. В теории симплекс-метода известно несколько таких правил, приводящих к различным вариантам метода. Самое простое из них состоит в том, что в качестве p выбирается отрицательная компонента с минимальным номером — правило «первая подходящая». Название другого часто используемого правила — «самая отрицательная» — говорит само за себя. Если в качестве p выбирается компонента, обеспечивающая наибольший прирост целевой функции, говорят о правиле «наибольшего приращения». Это правило требует больших объемов вычислений и используется редко. Еще одно правило «острейшего угла» будет описано в следующем параграфе.

Второй момент уточнения процедуры связан со способом вычисления обратной базисной матрицы в пункте 1 описания. При неоднократном применении процедуры (а именно это происходит в симплекс-методе) базисная матрица меняется каждый раз довольно незначительно — в ней заменяется некоторая строка p на новую строку q. Поэтому было бы неразумно обращать базисную матрицу каждый раз заново за n^3 операций и не использовать обратную матрицу предыдущего шага. Более точно, пусть матрица A_M^{-1} уже известна и требуется вычислить матрицу $A_{S(M)}^{-1}$ для базисного множества $S(M)$, полученного заменой p на q. Если обозначить через $A_M^{-1}[i]$ и $A_{S(M)}^{-1}[i]$ столбцы с номером i указанных матриц, то имеют место следующие формулы:

$$A_{S(M)}^{-1}[q] = A_M^{-1}[p] / (a_q A_M^{-1})_p, \quad (3.6)$$

$$A_{S(M)}^{-1}[i] = A_M^{-1}[i] - A_M^{-1}[p] (a_q A_M^{-1})_i / (a_q A_M^{-1})_p \quad \text{при } i \in M - p.$$

Для их доказательства заметим, что замена переменных (3.2) имеет вид

$$x = \sum_{i \in M} x_i' A_M^{-1}[i] = x_p' A_M^{-1}[p] + \sum_{i \in M - p} x_i' A_M^{-1}[i],$$

откуда с учетом

$a_q A_M^{-1}[i] = i$-я компонента вектора $a_q A_M^{-1} = (a_q A_M^{-1})_i$

следует

$$x_q' = a_q x = x_p' (a_q A_M^{-1})_p + \sum_{i \in M - p} x_i' (a_q A_M^{-1})_i.$$

Выражая из последнего равенства x_p' и подставляя в первое, получим (3.6).

Глядя на формулы (3.6), которые называют формулами одноранговой коррекции, нетрудно понять, что они позволяют вычислять обратную матрицу с затратой порядка n^2 операций. С учетом этого замечания и условия $m \geqslant n$ видно, что для выполнения процедуры S достаточно порядка nm операций.

Вернемся к описанию симплекс-метода. Как уже говорилось, его второй этап состоит в последовательном улучшении с помощью S допустимых базисных решений. Поскольку для невырожденной задачи значение целевой функции с каждым шагом строго растет, а число допустимых базисов конечно (не превосходит C_m^n), через конечное число шагов симплекс-метод приводит к решению задачи.

Для вырожденной задачи линейного программирования это рассуждение о конечности метода неприменимо. Действительно, если какие-либо чужие ограничения $\nu \in \mathbf{M} - M$ обращаются в равенство в базисном решении x_M, то соответствующие им величины τ_ν могут обращать-

ся в нуль при $(a_\nu A_M^{-1})_p<0$. Тогда вывод из M ограничения p и ввод в него одного из тех ограничений q, для которых $\tau_q=0$, приведут к замене базисного множества при неизменном базисном решении — произойдет, как говорят, замена базиса на нулевом уровне. Если не позаботиться о специальных правилах выбора замен $p\leftarrow q$, теоретически такой «бег на месте» может продолжаться неограниченно долго и привести к зацикливанию — периодической смене базисных множеств с одним и тем же базисным решением*. Поэтому пункты 4 и 6 описания симплекс-процедуры должны быть дополнены специальными правилами выбора замен $p\leftarrow q$, гарантирующими отсутствие зацикливания в случае вырождения. Первое правило с требуемым свойством, было описано в 50-х годах. Оно возникло из идеи отслеживания смены базисов при бесконечно малом $\varepsilon\to 0$ степенном возмущении $b_i+\varepsilon^i$ правых частей задачи, снимающем вырождение. В дальнейшем появились и другие правила устранения зацикливания, наиболее замечательным из которых по простоте является правило Блэнда «первое подходящее p, а затем первое подходящее q» [23].

Осталось рассмотреть вопрос об отыскании начального допустимого базисного решения, поиск которого осуществляется на первом этапе работы симплекс-метода. Прежде всего если для системы линейных неравенств известно какое-нибудь решение, не обязательно базисное, можно легко превратить его в допустимое базисное с помощью процедуры, описанной в § 1 при доказательстве принципа граничных решений. Однако часто такое решение не известно и даже не ясно, разрешима ли система ограничений (1.2). В такой ситуации прибегают к методу искусственного базиса, простейший вариант которого состоит в следующем. Сначала, как в § 2, находят произвольное базисное множество M системы неравенств (1.2). Если окажется, что для базисного решения $x_M=A_M^{-1}b_M$ все величины $b_\nu-a_\nu x_M$ неотрицательны, решение допустимо. В противном случае найдем среди величин $b_\nu-a_\nu x_M$ самую отрицательную, номер которой обозначим через m, и рассмотрим множество ограничений $M\cup m$ в следующей вспомогательной задаче линейного программирования с $n+1$ переменной: $x_{n+1}\to\max$,

$$a_i x\leqslant b_i,\ \ i\in M,$$

$$a_\nu x+x_{n+1}\leqslant b_\nu,\ \ \nu\in\mathbf{M}-M. \tag{3.7}$$

Нетрудно видеть, что в этой задаче множество ограничений $M\cup m$ является базисным и допустимым. К задаче (3.7) можно поэтому применять процедуру S улучшения допустимых базисных решений $(x_1, \ldots, x_n, x_{n+1})$ до тех пор, пока не получится решение с положительной целевой функцией $x_{n+1}\geqslant 0$. Тогда первые n компонент этого решения будут давать решение исходной системы неравенств (1.2). Если же S остановится на оптимальном решении задачи (3.7) и окажется, что $x_{n+1}<0$, система (1.2) неразрешима. Итак, первый этап симплекс-метода сводится ко второму. На этом закончим алгебраическое описание метода, заметив, что существуют его варианты, предназначенные для специальных видов задач линейного программирования, в частности для явного учета ограничений в виде равенств [1, 7, 8, 11, 13].

Как следует из описания, симплекс-метод осуществляет направленный перебор базисных множеств ограничений, обеспечивая их монотонное улучшение с помощью одноэлементных замен $p\leftarrow q$. Для невырожденных задач линейного программирования эти замены имеют простую геометрическую интерпретацию.

§ 4 Геометрическая интерпретация симплекс-метода. Задачи с экспоненциальным числом шагов. Оценки числа шагов в среднем

Как отмечалось в § 2, каждому неравенству $i\in M$ в системе линейных ограничений (1.2) соответствует в n-мерном пространстве некоторая плоскость с уравнением $a_i x=b_i$. Эта плоскость Π_i делит пространство на два полупространства, в одном из которых выполняется неравенство $a_i x\leqslant b_i$. Множество X решений системы линейных неравенств представляет собой, следовательно, пересечение конечного числа полупространств, границами которых являются плоскости Π_i. Так, для примера § 2 это множество образовано пересечением шести полупространств и представляет собой трехмерный многогранник, изображенный на рис. 3. Название «многогранное множество» сохраняется для пересечений конечного числа полупространства произвольной размерности n. Для того чтобы дать геометрическую интерпретацию симплекс-метода, потребуется понятие вершины и ребра многогранного множества X. Представим себе, что изображенный на рис. 3 трехмерный многогранник X сделан из твердого материала и выкрашен краской. Если взять в руку плоский лист картона и касаться им многогранника X, то можно получить на картоне отпечатки, представляющие собой либо точки, либо отрезки прямой, либо многоугольники. Восемь вершин $v^1, \ldots, v^8$ многогранника X характеризуются тем, что их можно коснуться первым способом, оставив на картоне точечный след (о вершину можно уколоться). В n-мерном пространстве эта характеризация вершин принимается за их определение. Именно говорят, что элемент $v=(v_1, v_2, \ldots, v_n)$ многогранного множества X является его вершиной, если найдется такое линейное уравнение $c_1x_1+c_2x_2+\ldots+c_nx_n=d$ (положение листа картона), что точка v ему удовлетворяет (оставляет след), а для всех прочих

* О возможности зацикливания в практических задачах см. [41].

точек $X-\{v\}$ многогранного множества выполнено строгое неравенство $c_1x_1+c_2x_2+\ldots+c_nx_n<d$ (все прочие точки расположены по одну сторону от листа картона и не касаются его). Это определение можно сделать короче, если сказать, что вершина — это такая точка $v\in X$, которая может служить единственным решением задачи максимизации на X некоторой линейной функции cx. Глядя на рис. 1 и 3, нетрудно догадаться, что вершинами многогранного множества X решений системы линейных неравенств (1.2) являются допустимые базисные решения x_M, и только они.

Доказательство этого факта несложно. С одной стороны, максимум любой линейной функции при ограничениях (1.2) достигается в одном из допустимых базисных решений, и поэтому вершины содержатся среди точек x_M. С другой стороны, из критерия оптимальности (3.4) видно, что каждое x_M может служить единственным решением задачи (1.1) — (1.2) при подходящей функции cx и потому является вершиной. Напомним, что в вырожденном случае одной вершине могут соответствовать несколько базисов M.

Определение ребра многогранного множества аналогично определению вершины. Каждое из 12 ребер многогранника X на рис. 3 характеризуется тем, что это конечный отрезок прямой, который может быть получен в результате касания многогранника X подходящей плоскостью (о ребро можно порезаться). В общем случае ребром e многогранного множества X называется такой содержащийся в нем конечный отрезок прямой, который может служить множеством решений подходящей задачи линейного программирования на X. Нетрудно видеть, что концевыми точками любого ребра являются вершины $e=[v^iv^j]$, однако не всякий отрезок, соединяющий две вершины, есть ребро (отрезок $[v^4v^8]$ на рис. 3 ребром не является).

В каком случае пара вершин v^i, v^j является смежной, т. е. соединена ребром? Критерием смежности двух вершин может служить существование такой линейной функции cx, что на двух указанных вершинах она принимает одинаковое значение

$$cv^i=cv^j, \qquad (4.1)$$

а на прочих вершинах — строго меньшее:

$$cv^l<cv^i,\ l\neq i,\ j. \qquad (4.2)$$

Другой критерий смежности вершин дает нам симплекс-метод — смежность вершин v^i и v^j эквивалентна тому, что при подходящем выборе целевой функции можно за один шаг симплекс-метода перейти от базисного множества M^i первой вершины к базисному множеству $M^j=S(M^i)$ второй.

Действительно, если симплекс-метод переходит за один шаг из v^i в v^j, то M^j получено из M^i заменой $p\leftarrow q$, и поэтому M^i и M^j отличаются одним элементом. Чтобы доказать, что $[v^iv^j]$ — ребро

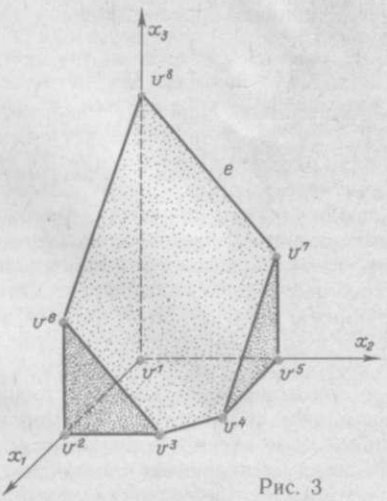


Рис. 3

многогранного множества X решений системы (1.2), достаточно доказать, что $[v^iv^j]$ — ребро многогранного множества, задаваемого $n+1$ неравенством $M^i\cup M^j$, что очевидно (см. «угол» на рис. 2).

Покажем, что и наоборот, любое ребро $[v^iv^j]$ может быть пройдено за один шаг симплекс-методом при подходящем выборе целевой функции. Для этого слегка возмутим функцию cx из первого критерия смежности, чтобы равенство (4.1) заменилось на неравенство $c'v^i<c'v^j$. Поскольку число строгих неравенств (4.2) конечно, при достаточно малом возмущении они будут продолжать выполняться: $c'v^l<c'v^i$, $l\neq i, j$. Возьмем теперь какое-нибудь допустимое базисное множество M^i вершины v^i в качестве начального для симплекс-метода с целевой функцией $c'x$. Поскольку симплекс-метод не может попасть в вершины v^l (они хуже начальной v^i), для попадания в оптимальную вершину v^j ему придется совершить переход по ребру $[v^iv^j]$ (в случае вырождения v^i пробежав, возможно, перед этим несколько шагов на месте).

Из сказанного выше вытекает геометрическая интерпретация симплекс-метода для невырожденных задач линейного программирования: метод строит идущий по смежным вершинам путь, вдоль которого растет целевая функция. Выбор подходящего ребра задается правилом выбора отрицательной компоненты в пункте 4 описания симплекс-процедуры. Так, помимо перечисленных в предыдущем параграфе, известно также правило выбора ребра, образующего с вектором целевой функции острейший угол. В результате работы симплекс-метода получается конечная последовательность вершин, приводящая к оптимуму.

Здесь уместно привести отрывок из сочинения Ж. Фурье [31], относящегося к 20-м годам прошлого столетия. Речь в нем идет о приближенном решении систем линейных уравнений $a_ix=b_i$, $i\in\mathbf{M}$ по критерию минимизации уклонения $\max|a_ix-b_i|$, или, на языке линейного программирования, о ре-

шении задачи вида $x_{n+1}\to\min$,

$$a_i x - b_i \leqslant x_{n+1},$$
$$-a_i x + b_i \leqslant x_{n+1},\ i\in\mathbf{M}. \qquad (4.3)$$

Если направить ось x_{n+1} «вверх», решением $(x_1,\ldots,x_n, x_{n+1})$ задачи будет самая низкая точка, расположенная выше $2m$ плоскостей, соответствующих ограничениям (4.3), т. е. самая низкая из точек, расположенных внутри многогранной чаши, дно которой состоит из кусков экстремальных плоскостей (см. рисунок на обложке). Описывая метод поиска решения этой задачи для наглядности в трехмерном пространстве, Фурье писал: «Чтобы быстро достичь самой низкой точки чаши, поднимают какую-нибудь точку горизонтальной плоскости, например начало координат x_1 и x_2, по вертикали до пересечения с самой верхней плоскостью, т. е. среди всех точек пересечения, которые находятся на этой вертикали, выбирают самую удаленную от плоскости x_1 и x_2. Пусть эта точка пересечения, расположенная на экстремальной плоскости, будет m_1. По той же плоскости спускаются из точки m_1 до точки m_2 какого-нибудь ребра многогранника и, следуя по этому ребру, спускаются из точки m_2 до вершины m_3, общей для трех экстремальных плоскостей. Начиная с точки m_3 продолжают спускаться по другому ребру до новой вершины m_4 и продолжают применять тот же прием, следуя всегда по тому из двух ребер, которое ведет в более низкую вершину. Таким образом, очень быстро приходят в самую низкую точку многогранника». В этом отрывке дано геометрическое описание обоих этапов симплекс-метода, вполне достаточное, как отмечает Фурье, для понимания метода в пространствах произвольного числа измерений.

Начавшиеся через 125 лет вместе с приходом компьютерной эры вычислительные эксперименты показали, что для большинства практических задач предсказание Фурье о быстрой работе метода действительно подтверждается — как правило, число шагов процедуры не более чем в несколько раз превосходит размерность задачи. Однако еще через 25 лет выяснилось, что так хорошо дело обстоит отнюдь не всегда, и в некоторых случаях для решения, например, задач с n переменными и $m=2n$ ограничениями симплекс-методу требуется 2^n минус один шаг, так что при нескольких десятках переменных процесс достижения дна чаши может затягиваться на годы.

Принадлежащая В. Кли и Дж. Минти [40] идея построения таких плохих для симплекс-метода задач может быть описана следующим образом. Обозначим через $H(n, m)$ максимальное число шагов, которое может потребоваться симплекс-методу для решения задач размерности (n, m), и предположим, что в размерности (n, m) плохой для симплекс-метода пример уже построен. Это означает, что нам известен n-мерный многогранник P, заданный m линейными ограничениями, в котором имеется путь $v^0v^1v^2\ldots v^k$ из $k=H(n, m)$ смежных вершин, вдоль которого растет некоторая линейная функция cx,— без ограничения общности эту функцию можно считать равной последней координате $cx=x_n$ (левая часть рис. 4). Добавим еще одну новую «вертикальную» координату x_{n+1} и рассмотрим в расширенном пространстве те же m ограничений, которые описывали многогранник P. Поскольку в эти ограничения переменная x_{n+1} не входит, множество решений получившейся системы в расширенном пространстве будет представлять собой многогранный цилиндр с сечением P в горизонтальной плоскости $x_{n+1}=\text{const}$ и вертикальными образующими — прямыми, исходящими из вершин P (правая часть рис. 4). Далее рассмотрим в расширенном пространстве график функции $x_{n+1}=cx$, который можно представить себе как наклоненный к горизонтальной плоскости лист картона, пересекающий наискось цилиндр,— на рис. 4 это пересечение обозначено сплошной штриховкой. Возьмем также большое число B и рассмотрим график функции $x_{n+1}=B-cx$, сечение которого с цилиндром обозначено более слабой штриховкой (число B выбирается таким образом, чтобы два заштрихованных на рисунке сечения не пересекались между собой в цилиндре). Если теперь добавить к m описывающим многогранник P ограничениям два новых $x_{n+1}\geqslant cx$ и $x_{n+1}\leqslant B-cx$, то в расширенном пространстве получится многогранник Q, представляющий собой часть цилиндра, заключенную между двумя наклонными заштрихованными плоскими основаниями. При этом каждой вершине v^i многогранника P будут соответствовать в многограннике Q две вершины u^i и w^i, первая из которых лежит на нижнем основании, а вторая — на верхнем. Ясно, что вдоль пути $u^0u^1u^2\ldots u^kw^k\ldots w^2w^1w^0$ длины $2k+1$ все время растет вертикальная координата x_{n+1}, поэтому

$$H(n+1, m+2)\geqslant 2H(n, m)+1.$$

С учетом того, что $H(1, 2)=1$ — для решения задачи линейного программирования на одномерном отрезке требуется один шаг,— получаем, в частности, $H(2, 4)\geqslant 3$, $H(3, 6)\geqslant 7$ и вообще

$$H(n, 2n)\geqslant 2^n-1.$$

Интересно отметить, что в рассмотренной индуктивной конструкции $P\to Q$ можно заменить неравенства $x_{n+1}\geqslant x_n$ и $x_{n+1}\leqslant B-x_n$ на неравенства $x_{n+1}\geqslant\varepsilon x_n$ и $x_{n+1}\leqslant 1-\varepsilon x_n$, где ε — произвольно малое положительное число из интервала $(0, 1/2)$, что приведет к сколь угодно малому наклону заштрихованных на рис. 4 оснований к горизонтальной плоскости. Если начать теперь индукцию с единичного отрезка, то линейное описание «плохого» многогранника будет выглядеть очень просто:

$$0\leqslant x_1\leqslant 1,$$
$$\varepsilon x_1\leqslant x_2\leqslant 1-\varepsilon x_1,$$
$$\ldots\ldots\ldots\ldots\ldots$$
$$\varepsilon x_{n-1}\leqslant x_n\leqslant 1-\varepsilon x_{n-1}$$

и входящие в него плоскости будут получаться малым поворотом и сдвигом плоскостей в описании обычного единичного куба, которое получается при $\varepsilon=0$. На рис. 5 изображен по-

Рис. 4

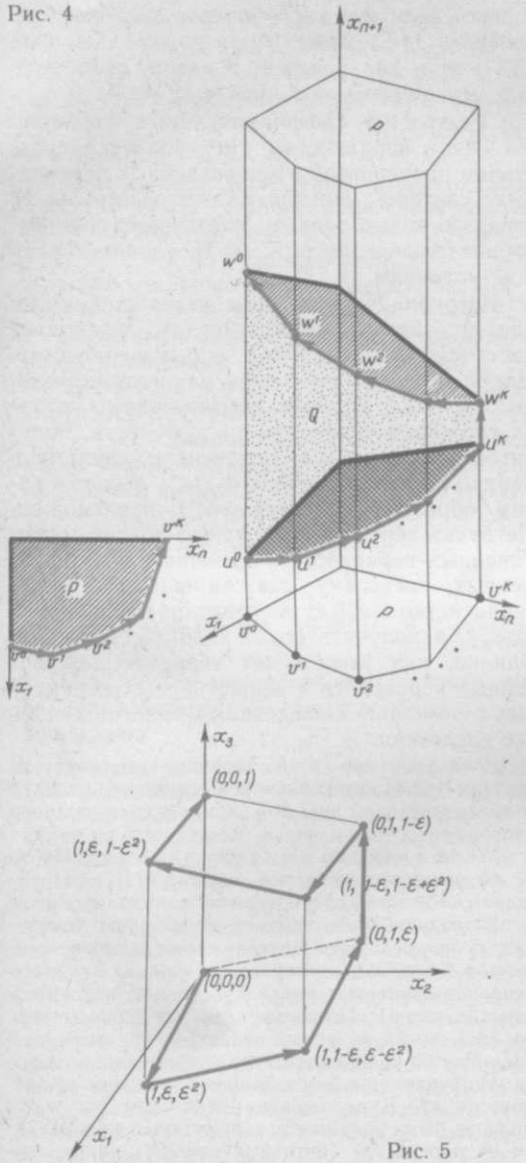


Рис. 5

лученный малой деформацией куба трехмерный «плохой» многогранник с $2^n=8$ вершинами, а также проходящий по всем вершинам путь из $2^n-1=7$ ребер, по которому все время растет вертикальная координата.

В. Кли и Дж. Минти также показали, что при фиксированном n величина $H(n, m)$ растет по m не медленнее, чем $c(n)m^{[n/2]}$, где $c(n)$ — зависящая от n константа. Другими словами, при фиксированном числе переменных рост числа шагов симплекс-метода по количеству ограничений может быть отнюдь не линейным, а полиномиальным степени $[n/2]$.

У читателя, возможно, возник вопрос: почему симплекс-метод должен отправляться в долгий путь $u^0u^1u^2\ldots u^kw^k\ldots w^2w^1w^0$ (см. рис. 4), когда имеется возможность достичь оптимума всего за один шаг переходом по ребру u^0w^0? Ответ зависит от того, какое правило выбора подходящего ребра принято в рассматриваемом варианте симплекс-метода. Фактически из рассмотренных примеров просто вывести, что симплекс-метод может отправляться в долгий путь при правилах «первая подходящая» и «самая отрицательная». Однако в дальнейшем соответствующие плохие примеры с экспоненциальным числом шагов были построены и для всех других известных вариантов симплекс-метода [35, 38], так что в настоящее время возможность построения симплекс-метода с полиномиальным и тем более линейным по размерности числом шагов оценивается довольно пессимистично. Между прочим, до сих пор остается недоказанной высказанная в 1957 г. М. Хиршем гипотеза, согласно которой между любыми двумя вершинами ограниченного многогранника размерности (n, m) всегда существует путь из смежных вершин, содержащий не более $m-n$ ребер. Доказательство этой гипотезы означало бы по крайней мере принципиальную возможность перехода от одного допустимого базиса к другому по цепочке допустимых с помощью линейного по размерности числа одноэлементных замен $p \leftarrow q$.

Для объяснения удачной работы симплекс-метода оценивалось его число шагов в среднем. Так, в работах [3, 46] для одной из версий симплекс-метода (параметрического самодвойственного алгоритма Данцига) были получены асимптотики среднего числа шагов по n при фиксированном m для решения задачи (1.1)—(1.2) в неотрицательных переменных в предположении, что входные коэффициенты задач берутся из стандартного нормального распределения. Эти оценки, однако, были «чересчур асимптотическими» и неполиномиальными при всех значениях n и m. Первая полиномиальная оценка $\text{const}\, n^3m^{1/(n-1)}$ среднего числа шагов специальной параметрической версии симплекс-метода (Schatteneckenalgorithmus) была получена в [25] для решения задачи (1.1)—(1.2) с единичными правыми частями и векторами $a_1,\ldots,a_m, c$, имеющими независимые, одинаковые и инвариантные относительно вращений распределения (в частности, нормальные). Далее в работах [21, 60, 22] была использована более мощная параметрическая версия симплекс-метода со степенным возмущением и для задач (1.1)—(1.2) с неотрицательными переменными при довольно слабых предположениях о невырожденности и инвариантности входной статистики относительно обращения знаков в неравенствах была получена квадратичная по $\min(n, m)$ оценка среднего числа шагов, причем в [21] и сверху, и снизу. Наконец, имеются сообщения о неопубликованной работе [36], где при аналогичных предположениях о статистике задач (1.1)—(1.2) получена при некотором случайном входном параметре* оценка сред-

* распределение которого зависит от входных данных и не поддается простому машинному моделированию.

него числа шагов, линейная по $\min(n, m)$.

Несомненно, эти работы проливают некоторый свет на феномен симплекс-метода. Однако утверждать, что оценки среднего числа шагов действительно объясняют его практическое поведение было бы, по-видимому, чрезмерно обобщающей интерпретацией этих результатов. Серьезным аргументом против такой интерпретации служит, в частности, то обстоятельство, что по крайней мере для некоторых из используемых статистик линейное программирование не отличается [50] по оценкам среднего числа шагов от задачи о линейной дополнительности при их решении одним и тем же методом*). Между тем на практике — при любом понимании этого слова — указанные задачи радикально отличаются по своей сложности, и решение *NP*-трудной задачи о линейной дополнительности с той же эффективностью, с какой решаются задачи линейного программирования, можно было бы сравнить с революцией в мире вычислений. Таким образом, загадка симплекс-метода остается во многом неразгаданной.

В заключение главы остановимся вкратце на теории двойственности в линейном программировании, результаты которой потребуются в § 8.

§ 5 Двойственность в линейном программировании

Говорят, что линейное неравенство

$$cx \leqslant d \qquad (*)$$

является следствием разрешимой системы линейных неравенств

$$a_i x \leqslant b_i,\ i \in \mathbf{M}, \qquad (1.2)$$

если для любого вещественного вектора $x=(x_1, \ldots, x_n)$, удовлетворяющего (1.2), выполняется $(*)$. Если умножить i-е неравенство системы (1.2) на произвольное неотрицательное число λ_i и сложить по $i \in \mathbf{M}$ все результаты, то получится неравенство

$$\Big[\sum_{i\in\mathbf{M}} \lambda_i a_i\Big] x \leqslant \Big[\sum_{i\in\mathbf{M}} \lambda_i b_i\Big],$$

которое, очевидно, является следствием (1.2). Это неравенство останется следствием (1.2) и в том случае, если несколько увеличить его правую часть. Другими словами, пусть c и d — вектор и скаляр, удовлетворяющие условиям

$$\sum_{i\in\mathbf{M}} \lambda_i a_i = c, \quad \sum_{i\in\mathbf{M}} \lambda_i b_i \leqslant d \qquad (5.1)$$

при некоторых неотрицательных числах $\lambda_i \geqslant 0$, $i \in \mathbf{M}$, тогда $(*)$ есть следствие (1.2). Важнейший результат теории линейных неравенств состоит в том, что и, наоборот, всякое следствие $(*)$ системы линейных неравенств (1.2) может быть получено в виде (5.1) при подходящем образом подобранных неотрицательных числах λ_i, $i \in \mathbf{M}$.

Теорема (аффинная лемма Фаркаша Линейное неравенство $(*)$ является следствием разрешимой в вещественных переменных системы линейных неравенств (1.2) тогда и только тогда, когда найдутся неотрицательные числа λ_i, $i \in \mathbf{M}$, удовлетворяющие условиям (5.1).

Нетривиальной в этом утверждении является часть «только тогда», поскольку часть «тогда» верна для любых неравенств. Например, если умножить на неотрицательные λ_i квадратичные неравенства и после их сложения увеличить правую часть, получится квадратичное неравенство-следствие, однако из неравенства $x_1^2 + x_2^2 \leqslant 1$ вывести таким образом следствие $x_1^2 \leqslant 1$ невозможно. Не будет верна часть «только тогда» и для линейных неравенств в целочисленных переменных, поскольку, скажем, в этом случае неравенство $x_1 \leqslant 0$ есть следствие неравенства $2x_1 \leqslant 1$, а получить его в виде (5.1) нельзя. Однако, как утверждает теорема, для линейных неравенств в вещественных переменных с помощью λ-следствий можно получить все следствия.

Доказательство этой теоремы вытекает из критерия (3.4) оптимальности симплекс-метода с учетом перехода, как в § 2, к задаче полного ранга.

Механическая интерпретация. Пусть X — множество решений системы (1.2). Представим себе материальную точку единичной массы в однородном поле тяжести с вектором ускорения c, находящуюся внутри многогранного множества X, граница которого сделана из гладкого непроницаемого материала (идеальные неудерживающие связи). Поскольку для $x \in X$ справедливо $(*)$, точка не может двигаться внутри многогранника по полю тяжести неограниченно, и имеется устойчивое или безразличное положение ее равновесия x^*. В положении равновесия вес точки должен быть уравновешен реакциями касающихся ее плоскостей (активных связей), направленных к ним по нормали, что дает первое из условий (5.1). Умножая это равенство сил на x^*, получаем из границы на потенциальную энергию $cx^* \leqslant d$ второе условие (5.1). Наконец, условие $\lambda_i \geqslant 0$ интерпретируется как направленность реакций связей внутрь многогранника по принципу виртуальных перемещений. Именно эта механическая интерпретация явилась причиной появления в конце прошлого века однородного варианта леммы Фаркаша.

Рассмотрим теперь случай неразрешимой системы линейных неравенств (1.2) с пустым множеством решений X. Оказывается, что лемма Фаркаша будет справедливой и для таких систем неравенств, если условиться считать их следствием неразрешимое неравенство $0 \cdot x \leqslant -1$. Именно имеет место следующий

---

*) алгоритмом Лемке, который на задачах линейного программирования совпадает с самодвойственным алгоритмом Данцига.

Вариант леммы Фаркаша о неразрешимости. Система линейных неравенств (1.2) неразрешима тогда и только тогда, когда разрешима система условий.

$$\sum_{i\in\mathbf{M}} \lambda_i a_i = 0,\ \sum_{i\in\mathbf{M}} \lambda_i b_i \leqslant -1,\ \lambda_i \geqslant 0,\ i\in\mathbf{M}. \quad (5.2)$$

Для доказательства этого варианта нужно лишь заметить, что из разрешимой системы линейных неравенств с одной новой переменной

$$a_i x + x_{n+1} \leqslant b_i,\ i\in\mathbf{M}$$

вытекает неравенство-следствие $x_{n+1} \leqslant -\varepsilon^*$ с некоторым строго отрицательным числом $-\varepsilon^*$

Обратимся теперь к теореме двойственности в линейном программировании.

Теорема двойственности. Задача линейного программирования на максимум (1.1)—(1.2) разрешима тогда и только тогда, когда разрешима задача линейного программирования на минимум

$$\sum_{i\in\mathbf{M}} \lambda_i b_i \to \min \quad (1.1^0)$$

при ограничениях

$$\sum_{i\in\mathbf{M}} \lambda_i a_i = c,\ \lambda_i \geqslant 0,\ i\in\mathbf{M}. \quad (1.2^0)$$

В случае разрешимости оптимальные значения целевых функций в обеих задачах совпадают.

Доказательство. Пусть задача (1.1)—(1.2) разрешима и d^* — максимальное значение ее целевой функции. Тогда неравенство $cx \leqslant d$ является следствием разрешимой системы (1.2) при $d \geqslant d^*$ и не является таковым при $d < d^*$. По лемме Фаркаша это означает разрешимость системы (5.1) в неотрицательных $\lambda = \lambda_1, \lambda_2, \ldots, \lambda_m$) при $d \geqslant d^*$ и неразрешимость при $d < d^*$. Но это не что иное, как разрешимость задачи (1.1^0)—(1.2^0) с минимальным значением целевой функции d^*. Наоборот, пусть разрешима задача (1.1^0)—(1.2^0), тогда, очевидно, система (5.2) не является разрешимой, так как в этом случае можно было бы при ограничениях (1.2^0) неограниченно уменьшать целевую функцию (1.1^0). Из неразрешимости (5.2) следует разрешимость (1.2). Применяя еще раз лемму Фаркаша, завершаем доказательство теоремы.

Таким образом, задаче линейного программирования (1.1)—(1.2), имеющей в матричной записи вид

$$cx \to \max,\ Ax \leqslant b, \quad (P)$$

соответствует двойственная задача линейного программирования (1.1^0)—(1.2^0), имеющая в матричной записи вид

$$\lambda b \to \min,\ \lambda A = c,\ \lambda \geqslant 0. \quad (D)$$

Теперь из теоремы двойственности видно, что вместо того чтобы решать прямую (P) и двойственную (D) задачи отдельно, можно решать их вместе, что эквивалентно решению системы линейных неравенств

$$\lambda b = cx,\ Ax \leqslant b,\ \lambda A = c,\ \lambda \geqslant 0, \quad (PD)$$

в которой всякая оптимизация уже пропала*.

Итак, как указывалось во введении, оптимизация в линейном программировании является в некотором смысле избыточной, и с помощью теории двойственности решение оптимизационных задач может быть сведено к решению систем линейных неравенств.

Отсюда следует, что решение задач линейного программирования может быть также сведено к решению систем линейных уравнений в неотрицательных переменных.

ГЛАВА 3. ПОЛИНОМИАЛЬНЫЕ АЛГОРИТМЫ ЛИНЕЙНОГО ПРОГРАММИРОВАНИЯ

В этой главе будут рассмотрены задачи линейного программирования с целыми коэффициентами и описаны два алгоритма точного решения с числом арифметических операций, ограниченным полиномом от битовой размерности задачи. Важно подчеркнуть, что дальнейший анализ этих алгоритмов показывает и полиномиальность числа знаков в числах, с которыми выполняются арифметические операции, доказывая тем самым возможность решения задач линейного программирования за полиномиальное по их битовой размерности время. К сожалению, анализ битовой длины чисел в указанных алгоритмах является довольно кропотливым делом, и поэтому, чтобы упростить изложение, мы будем считать, что в компьютере реализуется точная арифметика, лишь конспективно указывая на те изменения в описаниях алгоритмов, которые должны быть внесены для учета эффектов конечноразрядной компьютерной арифметики.

§ 6 Границы решений и мера несовместности задач с целыми коэффициентами Нахождение точного решения по приближенному

В этом параграфе мы рассмотрим некоторые вспомогательные факты о задачах линейного программирования с целыми коэффициентами и алгоритм, позволяющий по достаточно хорошему приближенному решению таких задач определить их точное решение с затратой кубического по n и m числа арифметических операций [16]. Всюду в дальнейшем требование целочисленности коэффициентов задачи линейного программирования подразумевается и для целочисленной матрицы D через $\Delta(D)$ обозначается положительная константа, ограничивающая сверху модули все-

* Мы говорим о системе неравенств, потому что входящие в (PD) равенства могут быть записаны в виде противоположных неравенств.

возможных определителей, которые можно получить вычеркиванием каких-либо строк или столбцов матрицы: $\Delta(D) \geqslant |\det D'|$, D' — квадратная подматрица D. Через $\|x\| = (x_1^2 + x_2^2 + \ldots + x_n^2)^{1/2}$ обозначается евклидова норма n-мерного вектора x. Кроме того, в соответствии с § 2 можно при желании считать $m \geqslant n$. Прежде всего потребуется

Теорема о границах решений задач линейного программирования. Если задача линейного программирования размерности (n, m)

$$cx \to \max, \quad Ax \leqslant b \qquad (1.1), (1.2)$$

разрешима, у нее существует рациональное решение x^* в шаре

$$\|x\| \leqslant R = n^{1/2}\Delta([A, b]), \qquad (6.1)$$

и при этом максимальное значение $d^* = cx^*$ целевой функции является рациональным числом $d^* = t/s$, t и s — целые, со знаменателем

$$|s| \leqslant \Delta(A). \qquad (6.2)$$

Доказательство. По принципу граничных решений в качестве x^* можно взять любое решение некоторой системы линейных уравнений РАВ (M). По правилу Крамера для системы линейных уравнений существует решение, каждая компонента которого имеет вид $x_j^* = \det D_j / \det D'$, где D_j — некоторая квадратная подматрица матрицы $[A, b]$ или нуль, и D' — некоторая квадратная подматрица А. Поскольку D' целочисленна, ее определитель является целым числом $1 \leqslant |\det D'| \leqslant \Delta(A)$, что доказывает (6.2). Далее, $|x_j^*| \leqslant |\det D_j| \leqslant \Delta([A, b])$, что доказывает (6.1) и всю теорему.

Пусть ε — некоторое положительное число. Будем говорить, что вектор x^0 является ε-приближенным решением системы линейных неравенств (1.2), если этот вектор нарушает каждое неравенство системы не более чем на ε,

$$a_i x^0 \leqslant b_i + \varepsilon, \; i \in \mathbf{M} = \{1, 2, \ldots, m\}. \quad (1.2\,\varepsilon)$$

Теорема о мере несовместности систем линейных неравенств. Положим

$$\varepsilon_1 = \frac{1}{(n+2)\,\Delta(A)}. \qquad (6.3)$$

Если система линейных неравенств (1.2) имеет ε_1-приближенное решение x^0, то она является разрешимой и имеет точное решение x^*.

Доказательство. Предположим, что система (1.2) неразрешима, и рассмотрим задачу линейного программирования $\varepsilon \to \min$, при ограничениях (1.2 ε). Эта задача разрешима. Пусть ε^* минимальное значение целевой функции в этой задаче, т. е. минимально возможное ε, для которого неразрешимая система (1.2) имеет ε-приближенное решение (число ε^* называется мерой несовместимости системы неравенств). Ясно, что $\varepsilon^* > 0$, и поэтому из (6.2) вытекает $\varepsilon^* = t/s \geqslant 1/|s| \geqslant 1/\Delta([A, -\mathbf{1}])$, где через $\mathbf{1}$ обозначен вектор-столбец из m единиц. По правилу вычисления определителей по столбцу имеем $\Delta([A, -\mathbf{1}]) \leqslant (n+1)\Delta(A)$, поскольку любая квадратная подматрица матрицы $[A, -\mathbf{1}]$ имеет не более $n+1$ строк. Следовательно, $\varepsilon^* > \varepsilon_1$, и теорема доказана.

Отметим, что, поскольку любая система линейных уравнений РАВ(M) может быть записана в виде $2|M|$ линейных неравенств $a_i x \leqslant b_i$ и $-a_i x \leqslant -b_i$, $i \in M$, из существования ε_1-приближенного решения системы линейных уравнений $|a_i x^0 - b_i| \leqslant \varepsilon_1$ вытекает ее точная разрешимость.

Опишем теперь простой алгоритм нахождения x^* по x^0. Этот алгоритм весьма близок к тому итеративному процессу, который использовался при доказательстве принципа граничных решений в § 1. Именно подставим известное приближенное решение x^0 в систему линейных неравенств (1.2) и разобьем множество $\mathbf{M}$ ограничений этой системы на два подмножества

$$|a_i x^0 - b_i| \leqslant \varepsilon_1, \quad i \in M(x^0),$$
$$a_i x^0 - b_i < -\varepsilon_1, \; i \in \mathbf{M} - M(x^0),$$

отнеся в первое из них ограничения, выполняемые в точке «почти как равенства», а во второе — ограничения, выполненные «с запасом». Положим $M = M(x^0)$. По теореме о мере несовместности система РАВ(M^0) является разрешимой, и поэтому с помощью, например, алгоритма Гаусса можно найти ее точное решение $\bar{x}$, не обращая внимания на оставшиеся неравенства $i \in \mathbf{M} - M^0$. Если при этом окажется, что $\bar{x}$ удовлетворяет всем неравенствам $\mathbf{M} - M^0$, этот вектор будет являться точным решением системы (1.2), и можно закончить алгоритм. В противном случае обозначим через M^+ множество нарушенных в точке $\bar{x}$ ограничений

$$a_i \bar{x} - b_i = 0, \; i \in M^0,$$
$$a_i \bar{x} - b_i > 0, \; i \in M^+,$$
$$a_i \bar{x} - b_i \leqslant 0, \; i \in \mathbf{M} - (M^0 \cup M^+)$$

и рассмотрим прямолинейный отрезок $[x^0 \bar{x}]$, соединяющий две известные нам точки x^0 и $\bar{x}$, т. е. множество точек вида $x(t) = (1-t)x^0 + t\bar{x}$ при значениях параметра $t \in [0, 1]$. Для любой точки $x(t)$ отрезка будем иметь $|a_i x(t) - b_i| \leqslant \varepsilon_1$ при $i \in M^0$ и $a_i x(t) - b_i \leqslant 0$ при $i \in \mathbf{M} - (M^0 \cup M^+)$, так что все неравенства системы, кроме M^+, на отрезке будут выполнены, причем неравенства M^0 будут выполняться на всем отрезке «почти как равенства» с точностью ε_1. Вычислим теперь для всех $i \in M^+$ величины

$$\tau_i = [b_i - a_i x^0] / [a_i \bar{x} - a_i x^0],$$

тогда неравенство $i \in M^+$ системы будет выполняться строгим образом $a_i x(t) - b_i < 0$ при $t \in [0, \tau_i)$, обращаться в равенство при

$t=\tau_i$ и нарушаться при $t\in(\tau_i, 1]$. Это означает, что если обозначить через τ минимальную из величин τ_i, $i\in M^+$ и рассмотреть вектор $x^1=x(\tau)$, то этот последний, как и x^0, будет являться ε_1-приближенным решением системы (1.2), и при этом множество ограничений, выполняемых в точке x^1 «почти как равенства», увеличится по сравнению с $M(x^0)$ по крайней мере на одно ограничение: $M(x^0)\subset M(x^1)$. Продолжая действовать подобным образом, через не более чем n шагов придем к системе $\mathrm{PAB}(M^k)$, точное решение которой даст точное решение исходной системы линейных неравенств (1.2). Поскольку наиболее трудоемкой частью алгоритма является решение систем линейных уравнений $\mathrm{PAB}(M^0)$, $\mathrm{PAB}(M^1)$, ..., $\mathrm{PAB}(M^k)$ с вложенными друг в друга множествами уравнений, число арифметических операций, требующихся для нахождения точного решения x^* по приближенному x^0, будет ограничено по порядку величиной n^2m.

Итак, нами описан алгоритм «округления» $x^0\to x^*$, позволяющий по ε_1-приближенному решению x^0 системы линейных неравенств достаточно быстро находить ее точное решение x^*, представляющее собой рациональный вектор со знаменателем $\leqslant\Delta(A)$. Распространим теперь этот результат и на оптимизационный вариант задачи, считая для удобства, что речь идет о максимизации функции x_n, равной последней компоненте вектора переменных (к этому случаю легко перейти, введя в задачу новую переменную, равную cx). Здесь прежде всего потребуется следующее простое

У т в е р ж д е н и е. Пусть $\varepsilon\leqslant\varepsilon_1$ и вектор x^0 удовлетворяет системе $\mathrm{PAB}(M)$ с точностью ε: $|a_ix^0-b_i|\leqslant\varepsilon$; $i\in M$. Тогда существует точное решение $\bar{x}$ системы $\mathrm{PAB}(M)$, мало отличающееся в n-й компоненте от x^0:

$$|\bar{x}_n-x^0_n|\leqslant\varepsilon n\Delta(A). \qquad (6.4)$$

Действительно, компонента x_n может принимать на множестве решений системы $\mathrm{PAB}(M)$ либо все значения, либо одно $\bar{x}_n=\mathrm{const}$. В первом случае левую часть (6.4) выбором $\bar{x}$ можно сделать нулевой, во втором условие $x_n=\mathrm{const}$ есть следствие $\mathrm{PAB}(M)$, и поэтому $c=[0, ..., 0, 1]=\sum\lambda_ia_i$ по $i\in M$, где можно считать $|M|\leqslant n$, и по теореме о границах решений $|\lambda^i|\leqslant\Delta(A)$. Умножая это равенство на $\bar{x}$ и x^0, получаем (6.4).

Определим теперь для оптимизационной задачи понятие ε-приближенного решения x^0, добавив к условиям (1.2ε) малости нарушения ограничений требование

$$x^0_n\geqslant d^*-\varepsilon \qquad (1.1\varepsilon)$$

достижения с той же точностью максимального значения d^* последней компоненты. Если положить

$$\varepsilon_2=\frac{1}{2n^2\Delta^3(A)}, \qquad (6.5)$$

то, найдя ε_2-приближенное решение задачи, можно будет по-прежнему за n^2m арифметических операций «округлиться» в точное решение. Для этого достаточно на шагах старой процедуры $x^0\to x^*$ каждый раз выбирать точное решение $\bar{x}$ системы $\mathrm{PAB}(M)$ из условия (6.4) по отношению к текущему приближенному решению. В самом деле, через не более чем n шагов процедуры будет получено точное рациональное решение x^* системы ограничений (1.2), для которого из (6.4) и (1.1ε) будет следовать

$$d^*\geqslant x^*_n\geqslant x^0_n-n\varepsilon_2n\Delta(A)=$$
$$=x^0_n-1/(2\Delta^2(A))>d^*-1/\Delta^2(A),$$

и поэтому

$$|d^*-x^*_n|<1/\Delta^2(A). \qquad (6.6)$$

Осталось заметить, что d^* и x^*_n являются рациональными числами t/s и t_1/s_1 со знаменателями $\leqslant\Delta(A)$, и если они различны, то

$$|d^*-x^*_n|=|(ts_1-t_1s)|/|ss_1|\geqslant$$
$$\geqslant 1/|ss_1|\geqslant 1/\Delta^2(A),$$

что противоречит (6.6). Следовательно, полученный в результате процедуры «округления» вектор x^* является точным решением оптимизационной задачи.

Здесь необходимо сделать одно важное замечание относительно числа разрядов в числах, с которыми приходится оперировать алгоритму округления $x^0\to x^*$. Если ε_2-приближенное решение задачи линейного программирования лежит в шаре (6.1), то для двоичной записи каждой его компоненты будет достаточно порядка $\log(n\Delta)$ разрядов до и после запятой, где через $\Delta=\Delta(D)$ здесь и в дальнейшем будет обозначаться константа, мажорирующая модули определителей расширенной матрицы коэффициентов

$$D=\begin{bmatrix} c & 0 \\ A & b \end{bmatrix} \qquad (6.7)$$

задачи линейного программирования (читатель может убедиться в том, что величина Δ не меняется при переходе к задаче с новой переменной $x_{n+1}=cx$). Оказывается, и это требует самостоятельного доказательства (см. [6], с. 43) — указанного числа разрядов заведомо хватит и для точного решения систем $\mathrm{PAB}(M)$ методом Гаусса на шагах алгоритма округления*.

* Если реализовывать алгоритм Гаусса в рациональной арифметике, решая линейные уравнения с целыми коэффициентами точным образом «в дробях», как это делают в младших классах школы. На самом деле большую часть вычислений при точном решении системы линейных уравнений можно выполнять с существенно более короткими числами, например с помощью p-адической арифметики [28], однако для дальнейшего эта экономия не существенна.

В заключение параграфа установим некоторые полезные для дальнейшего оценки сверху величины $\log(n\Delta)$. Первая из них является простой и весьма грубой

$$\log(n\Delta)<L, \qquad (6.8)$$

где L — длина входа задачи, т. е. количество символов 0 и 1, необходимых для двоичной записи входной информации D. Действительно, для двоичной записи натурального числа a требуется не менее $\log(a+1)$ разрядов, и поэтому с учетом знаков и разделителей для записи всех коэффициентов матрицы D потребуется

$$L>\sum\log(|a_{ij}|+1)+\sum\log(|b_i|+1)+\\+\sum\log(|c_j|+1)+n+m \qquad (6.9)$$

битов. Но для модуля любого определителя подматрицы D' матрицы D по определению функции det имеем

$$2^L>2^{n+m}\prod(|a_{ij}|+1)\prod(|b_i|+1)\prod(|c_j|+1)>\\>n|\det D'|,$$

что совпадает с (6.8).

Более разумную оценку величины $\log(n\Delta)$ можно получить из неравенства Адамара: определитель матрицы не превосходит произведения евклидовых длин ее строк (или столбцов). Геометрически это неравенство означает, что определитель матрицы, являясь объемом параллелепипеда, образованного векторами-строками (или столбцами), достигает своего максимального по модулю значения, равного произведению длин этих векторов, когда они взаимно перпендикулярны. Если обозначить через

$$h=\max(|a_{ij}|, |b_i|, |c_j|) \qquad (6.10)$$

максимум модулей коэффицентов задачи линейного программирования*, то евклидова длина любой строки подматрицы D не будет превосходить $n^{1/2}h$, поэтому окажется $\Delta\leqslant n^{n/2}h^n$, так что при $n\geqslant 2$ будет справедлива оценка

$$\log(n\Delta)\leqslant n\log(nh). \qquad (6.11)$$

Подведем итог сказанному в параграфе. Для того чтобы точно решить задачу линейного программирования с целыми коэффициентами, достаточно найти в шаре $\|x\|\leqslant n^{1/2}\Delta$ ее $1/(2n^2\Delta^3)$-приближенное решение. После этого с помощью «округления» можно будет быстро найти точное решение. Трудоемкость алгоритма округления по порядку совпадает с трудоемкостью алгоритма Гаусса — n^2m арифметических операций. Двоичная длина чисел, с которыми работает алгоритм округления, по порядку составляет $\log(n\Delta)$ разрядов и ограничена битовой размерностью задачи.

* Величину h принято называть высотой задачи, $h\leqslant 2^l$, где l — цифровая длина входных коэффициентов.

§ 7 Метод эллипсоидов Полиномиальная разрешимость линейного программирования

Представим себе, что в нашем распоряжении имеется некоторый метод — назовем его $M(\varepsilon, R)$ — нахождения ε-приближенных решений задач линейного программирования

$$cx^0\geqslant d^*-\varepsilon,\ a_ix^0\leqslant b_i+\varepsilon,\ i\in M=\{1, 2, \ldots, m\}$$

в шаре $(x_1^2+\ldots+x_n^2)^{1/2}\leqslant R$ с полиномиальной по n, m и параметру $\log(Rnh/\varepsilon)$ оценкой числа арифметических операций. Если использовать этот метод при $R=n^{1/2}\Delta$ и $\varepsilon=1/(2n^2\Delta^3)$ для решения произвольной задачи линейного программирования с целыми коэффициентами, то поскольку

$$\log(Rnh/\varepsilon)\leqslant\log(Rn\Delta/\varepsilon)=\\=\log(2n^{3,5}\Delta^5)<5\log(n\Delta), \qquad (7.1)$$

количество требуемых операций окажется полиномиальным по n, m и $\log(n\Delta)$ и, значит, ограниченным полиномом от битовой длины входа L задачи (см. (6.9)). Если, кроме того, удастся показать, что для выполнения операций в методе $M(\varepsilon, R)$ достаточно полиномиального по n, m и $\log(Rnh/\varepsilon)$ количества разрядов, то, комбинируя $M(\varepsilon, R)$ с алгоритмом округления, можно будет за полиномиальное по L время точно решить любую задачу линейного программирования с целыми коэффициентами. Это простое соображение было использовано в 1979 г. [15] для доказательства полиномиальной разрешимости линейного программирования с помощью предложенного несколькими годами ранее [20, см также 19] метода поиска приближенного решения $M(\varepsilon, R)$, получившего название метода эллипсоидов.

В основе метода эллипсоидов лежит следующее геометрическое

Предложение. Пусть E — произвольный эллипсоид в n-мерном пространстве с ненулевым объемом $\mathrm{vol}\,E$ и ξ — его центр. Рассмотрим произвольную проходящую через точку ξ плоскость Π, заданную ненулевым вектором нормали g, и пусть $E^-(g)=E\cap\{g(x-\xi)\leqslant 0\}$ — один из двух полуэллипсоидов, на которые разбивается эллипсоид E плоскостью Π. Полуэллипсоид $E^-(g)$ можно целиком заключить в новый эллипсоид E', имеющий объем, строго меньший объема исходного эллипсоида

$$\mathrm{vol}\,E'/\mathrm{vol}\,E<e^{-1/(2n+2)},\ e=2{,}71828\ldots \qquad (7.2)$$

При этом эллипсоид E' можно вычислить по полуэллипсоиду $E^-(g)$ с затратой порядка n^2 операций.

На рис. 6 полуэллипсоид $E^-(g)$ заштрихован, а граница описанного вокруг него эллипсоида E' обозначена пунктиром. Чтобы убедиться в справедливости (7.2) и явно указать описанный эллипсоид E', рассмотрим сначала случай, когда исходный эллипсоид E представляет собой шар единичного радиуса $\|x\|\leqslant 1$ с центром ξ в начале координат, а полуэллипсоид $E^-(g)$ — его «верхний» полушар $x_n\geqslant 0$, заштрихованный на рис. 7. Поместим центр описанного эллипсоида в точку $\xi'=(0, \ldots, 0, 1/(n+1))$, так что E' будет задаваться квадратичным неравенством

$$x_1^2/\beta^2+x_2^2/\beta^2+\ldots+x_{n-1}^2/\beta^2+(x_n-1/(n+1))^2/\alpha^2\leqslant 1,$$

где через $\alpha=n/(n+1)$ обозначена меньшая полуось E'. Величина β других полуосей находится из уравнения $1/\beta^2+(0-1/(n+1))^2/\alpha^2=1$, что приводит к значению* $\beta=n/(n^2-1)^{1/2}$. Отношение объема описанного эллипсоида E' к объему E равно произведению n полуосей $\alpha\beta^{n-1}$, откуда с учетом неравенств

$$\alpha=1-1/(n+1)<e^{-1/(n+1)},$$
$$\beta^2=1+1/(n^2-1)<e^{1/(n^2-1)}$$

сразу получается (7.2) для случая, когда исходный эллипсоид E представляет собой шар. Однако любой эллипсоид E можно превратить в шар с помощью соответствующего аффинного преобразования координат, а поскольку отношение объемов тел остается при таких преобразованиях неизменным, неравенство (7.2) оказывается справедливым в общем случае. Далее, пусть эллипсоид E задается в памяти ЭВМ с помощью своего центра $\xi=(\xi_1, \ldots, \xi_n)$ и квадратной $n\times n$ матрицей $Q=[q_{ij}]$, которые вместе кодируют соответствующее аффинное преобразование

$$E=\{x\,|\,x=\xi+Qz,\ \|z\|\leqslant 1\}.$$

Другими словами, пусть исходный эллипсоид $E\sim(\xi, Q)$ представляет собой сдвинутый в точку ξ образ единичного шара $\|z\|\leqslant 1$ при преобразовании Q. Тогда при заданном векторе отсечения g параметры $E'\sim(\xi', Q')$ нового эллипсоида с меньшим объемом, описанного вокруг $E^-(g)$, могут быть вычислены по формулам

$$\xi'=\xi-\frac{1}{n+1}Q\eta,$$

$$Q'=\frac{n}{(n^2-1)^{1/2}}\left[Q+\left(\sqrt{\frac{n-1}{n+1}}-1\right)Q\eta\eta^t\right], \quad (7.3)$$

в которых через η обозначен вектор единичной длины $Q^tg/\|Q^tg\|$, знак t означает транспонирование, а $Q\eta\eta^t$ — квадратная матрица, ij-й элемент которой равен произведению i-й компоненты вектора $Q\eta$ на j-ю компоненту вектора η. Из формул (7.3) видно, что при задании эллипсоидов в памяти ЭВМ в указанном выше виде для вычисления эллипсоида E' хватает порядка n^2 операций.

Для того чтобы описать нахождение ε-приближенного решения задачи линейного программирования методом эллипсоидов, рассмотрим теперь все множество X_ε^* таких решений в шаре

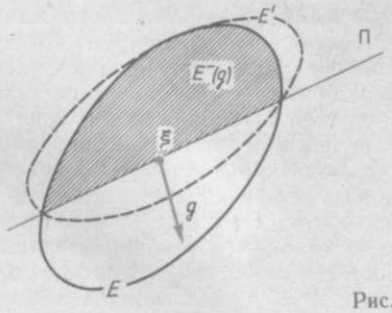


Рис. 6

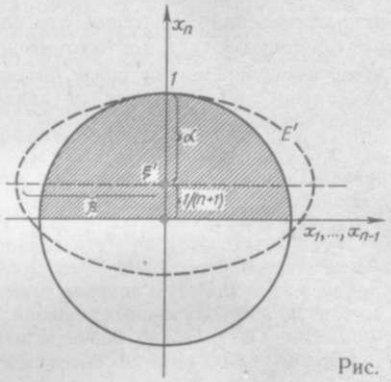


Рис. 7

$$X_\varepsilon^*=\{x\,|\,\|x\|\leqslant R,\quad cx\geqslant d^*-\varepsilon,\ a_ix\leqslant b_i+\varepsilon,\ i\in M\} \quad (7.4)$$

и предположим, что оно содержится в некотором известном нам эллипсоиде $X_\varepsilon^*\subseteq E$ с центром в точке ξ. Покажем, что если ξ не является ε-решением задачи, то по формулам (7.3) можно вычислить новый эллипсоид $X_\varepsilon^*\subseteq E'$ меньшего объема, также содержащий множество ε-решений. Действительно, если для некоторого ограничения $i\in\mathbf{M}$ окажется $a_i\xi>b_i+\varepsilon$, в чем можно убедиться подстановкой ξ в систему неравенств задачи, то для любого ε-решения x будем иметь $a_ix\leqslant b_i+\varepsilon<a_i\xi$, и поэтому

$$X_\varepsilon^*\subseteq E\cap\{x\,|\,a_i(x-\xi)\leqslant 0\}=E^-(a_i).$$

Значит, если какое-то i-е ограничение задачи нарушится в точке ξ более чем на ε, множество X_ε^* можно будет заключить в полуэллипсоид $E^-(a_i)$, а затем вычислить по формулам (7.3) при $g=a_i$ новый эллипсоид меньшего объема, также содержащий X_ε^*. Предположим поэтому, что при подстановке ξ в систему ограничений все они выполняются с требуемой точностью: $a_i\xi\leqslant b_i+\varepsilon$, $i\in\mathbf{M}$. Поскольку предполагается, что ξ не является ε-решением задачи, то $c\xi<d^*-\varepsilon$, и тогда по определению

$$X_\varepsilon^*\subseteq E\cap\{x\,|\,c(x-\xi)\geqslant 0\}=E^-(-c).$$

Следовательно, проведя вычисления (7.3) с вектором $g=-c$, опять получим эллипсоид E' меньшего объема, содержащий X_ε^*.

* Можно показать, что этот выбор E' оптимален, т. е. среди всех описанных вокруг полушара $E^-(g)$ эллипсоидов E' имеет минимальный объем.

Решение задачи линейного программирования методом эллипсоидов состоит в индуктивном применении указанной конструкции. На первой итерации рассматривается эллипсоид E^1, представляющий собой шар радиуса R с центром в начале координат, который содержит множество (7.4) искомых приближенных решений. Далее строится последовательность $E^1, E^2, \ldots, E^\kappa$ эллипсоидов убывающего объема (κ — номер итерации), обладающая тем свойством, что либо А) один из центров $\xi^1, \xi^2, \ldots, \xi^\kappa$ эллипсоидов является ε-решением задачи, либо Б) любой из указанных эллипсоидов содержит все множество X_ε^* приближенных решений. Покажем, что случай Б не может осуществляться достаточно долго.

Для этого вычислим оценку снизу объема X_ε^*. Пусть x^* — точное решение задачи линейного программирования в шаре $\|x\| \leqslant R$

$$cx^* = d^*,\ a_i x^* \leqslant b_i,\ i \in \mathbf{M}$$

и $\dot{x}$ — произвольная точка, расстояние от которой до x^* не превышает r. Поскольку модули чисел a_{ij}, c_j не превосходят h, при $\|x^* - \dot{x}\| \leqslant r$ будем иметь $|cx^* - c\dot{x}| \leqslant \|c\| \|x^* - \dot{x}\| \leqslant hn^{1/2} r$ и аналогично $|a_i x^* - a_i \dot{x}| \leqslant hn^{1/2} r$ для всех $i \in \mathbf{M}$. Это означает, что шар радиуса $r = \varepsilon/(hn^{1/2})$ с центром в точке $\dot{x}$ целиком состоит из ε-приближенных решений, и потому в множество (7.4) можно вписать шар радиуса $r/2$. Следовательно,

$$\operatorname{vol} X_\varepsilon^* / \operatorname{vol} E^1 \geqslant \left(\frac{r}{2R}\right)^n = \left(\frac{\varepsilon}{2Rhn^{1/2}}\right)^n.$$

С другой стороны, при выполнении Б из неравенства (7.2) имеем

$$\operatorname{vol} X_\varepsilon^* / \operatorname{vol} E^1 \leqslant \operatorname{vol} E^k / \operatorname{vol} E^1 < e^{-k/(2n+2)},$$

откуда сразу получается оценка

$$k < (2n+2)n \ln (2Rhn^{1/2}/\varepsilon) < 2n^2 \log (Rnh/\varepsilon)$$

числа итераций, в течение которых обязан выполниться случай А.

Таким образом, по крайней мере на одной из $2n^2 \log (Rnh/\varepsilon)$ итераций центр текущего эллипсоида окажется ε-приближенным решением задачи линейного программирования, что и завершает описание метода.

До сих пор мы игнорировали эффекты конечноразрядной компьютерной арифметики и неизбежные ошибки округлений. Описание метода эллипсоидов с учетом этих эффектов основывается на некотором «раздутии» описанного эллипсоида E' с целью поглощения ошибок округлений, т. е. на соответствующей коррекции формул (7.3), и здесь не приводится (см. [15]), а также [17], где содержатся точные выражения необходимого числа разрядов, пригодные для более общего случая*). В результате оказывается, что для двоичной записи чисел, над которыми выполняются арифметические операции в методе эллипсоидов, достаточно порядка $\log (Rnh/\varepsilon)$ разрядов. Что же касается количества арифметических операций, то с учетом стоимости $n^2 + nm$ каждой из итераций оно по порядку составляет величину $n^3(n+m) \log (Rnh/\varepsilon)$.

Прокомментируем смысл полученных оценок трудоемкости, начав с того замечания, что количество разрядов $\log (Rnh/\varepsilon)$ является необходимым для записи ε-приближенного решения в шаре $\|x\| \leqslant R$. Действительно, если какая-то координата приближенного решения окажется по модулю равной R, что вполне может случиться, для ее записи потребуется $\log R$ разрядов перед запятой. Далее, если отвести на запись каждой компоненты ν знаков после запятой, то общая ошибка, которая набежит в сумме $a_{i1}x_1 + \ldots + a_{in}x_n$ при округлении в последнем знаке, может составить при коэффициентах $|a_{ij}| = h$ величину $nh2^{-\nu}$, которая не должна превосходить ε. Следовательно, для записи ε-приближенного решения требуется по крайней мере $\log (nh/\varepsilon)$ знаков после запятой. На практике при приближенном решении задач линейного программирования величина $\log (Rnh/\varepsilon)$ числа знаков в ответе не превосходит нескольких десятков, и, как показывают расчеты, методу эллипсоидов во всех промежуточных расчетах заведомо хватает примерно учетверенного по сравнению с $\log (Rnh/\varepsilon)$ числа разрядов.

Обратимся теперь к оценке $\operatorname{const} n^3(n+m) \log (Rnh/\varepsilon)$ требующихся методу операций. Поскольку число операций симплекс-метода в большинстве случаев кубично по размерности, при решении массовых задач симплекс-метод оказывается на практике предпочтительнее метода эллипсоидов в линейном программировании*. Однако не следует забывать о гарантированном характере оценок трудоемкости метода эллипсоидов.

Комбинируя метод эллипсоидов с алгоритмом округления, получаем с учетом описанной в § 2 возможности перехода от общей задачи линейного программирования к задаче, в которой число переменных не превосходит числа ограничений, следующий важный теоретический результат.

Т е о р е м а. Задачи линейного программирования с целыми коэффициентами разрешимы за полиномиальное по их битовой размерности время. Для точного решения общей задачи линейного программирования с n неизвестными и m ограничениями методом эллипсоидов достаточно выполнить не более $\operatorname{const} \max(n, m) \min^3(nm) L$ арифметических операций над числами, имеющими в двоичной записи не более $\operatorname{const} L$ разрядов. Здесь под L можно понимать любую из следующих трех величин:

$\log [\min(n, m)\Delta]$, где Δ — константа, мажорирующая модули определителей матрицы коэффициентов задачи;

$\min(n, m) \log [\min(n, m) h]$, где h —

* Для задачи не только линейного, но и, например, квадратичного программирования с квадратичными ограничениями.

* В отличие, скажем, от выпуклого квадратичного программирования с квадратичными ограничениями, где симплекс-метод не работает и метод эллипсоидов оказывается практичным.

максимум модулей коэффициентов задачи;

двоичную длину входа задачи, т. е. число цифр в ее записи.

После доказательства полиномиальной разрешимости линейного программирования с помощью метода эллипсоидов появилось несколько других приближенных методов $M(\varepsilon, R)$, с помощью которых можно вывести тот же результат*. В следующем параграфе будут описаны идеи полиномиального метода Н. Кармаркара [39], который, судя по имеющимся сообщениям, оказывается конкурентоспособным с симплекс-методом на практике и в ряде случаев на задачах большой размерности его превосходит.

§ 8 Полиномиальный метод Кармаркара

В конце § 5 было показано, что решение общей задачи линейного программирования может быть сведено к решению системы линейных уравнений в неотрицательных переменных, число которых будет удобно обозначать через $N-1$:

$$\widehat{P}y=\widehat{Q}; \qquad (8.1)$$

$y=(y_1, ..., y_{N-1})\geqslant 0$ — вектор неизвестных;

$\widehat{P}$ — заданная целая матрица размера $M\times(N-1)$;

Q — заданный целый вектор размера M.

Добавим к системе уравнений неравенство

$$\mathbf{1}\,y=y_1+...+y_{N-1}\leqslant NR,$$

где R — известный либо из априорных соображений, либо по теореме о границах решений $R=\Delta([\widehat{P}, \widehat{Q}])$ параметр, мажорирующий координаты некоторого решения, если таковые в задаче существуют. Используя это неравенство, перепишем систему уравнений в однородном виде. Для этого добавим в задачу еще одну неотрицательную переменную y_N и превратим неравенство в равенство

$$[\widehat{P}, 0]y=\widehat{Q}, \quad \mathbf{1}\,y=y_1+...+y_{N-1}+y_N=NR,$$

где $[\widehat{P}, 0]$ — матрица размера $M\times N$, полученная дописыванием к $\widehat{P}$ столбца из одних нулей. Изменив масштаб переменных, добьемся того, чтобы правая часть последнего равенства равнялась N:

$$R[\widehat{P}, 0]x=\widehat{Q}, \ \mathbf{1}x=x_1+...+x_N=N,$$

где $x=R^{-1}y\geqslant 0$ — вектор новых неизвестных. Умножив M первых уравнений на N слева и на $x_1+...+x_N$ справа, окончательно получим

$$Px=0, \ x=(x_1, ..., x_N)\geqslant 0, \ \mathbf{1}\,x=N, \qquad (8.2)$$

где $M\times N$ матрица P получается умножением матрицы $\widehat{P}$ исходной задачи на число NR, дописыванием столбца из нулей и вычитанием из всех столбцов вектора $\widehat{Q}$. Задача (8.2) удобна для первоначального изложения метода Кармаркара. Прежде, однако, отметим одно простое, но важное для дальнейшего обстоятельство: условие $\mathbf{1}x=N$ можно ослабить до строгого неравенства $\mathbf{1}\,x>0$. Действительно, если x — решение (8.2), то, очевидно, $\mathbf{1}x=N>0$. Наоборот, если будет найдено решение задачи

$$Px=0, \ x=(x_1, ..., x_N)\geqslant 0, \ \mathbf{1}x>0, \qquad (8.3)$$

то, умножив его на подходящим образом подобранный положительный скаляр $x\to tx$, можно будет сделать величину $\mathbf{1}x$ равной любому положительному числу, в частности N. Таким образом, вместо (8.2) можно рассматривать однородную («проективную») задачу (8.3), решениями которой являются лучи tx.

Заменим теперь систему линейных уравнений $p_ix=0$, $i=1, 2, ..., M$, (p_i — строки матрицы P) на одно квадратное уравнение $f(x)=0$,

$$f(x)=(p_1x)^2+...+(p_Mx)^2$$

и рассмотрим функцию Кармаркара

$$k(x)=[f(x)]^{N/2}/\Pi(x), \qquad (8.4)$$

где через $\Pi(x)=x_1x_2...x_N$ обозначено произведение всех координат вектора x. Покажем прежде всего, что для точного решения исходной задачи (8.1) достаточно найти вектор $x>0$ со строго положительными координатами, для которого

$$k(x)\leqslant 1/[3(\Delta(\widehat{P}))^N], \qquad (8.5)$$

где $\Delta(\widehat{P})$ — константа, мажорирующая модули всевозможных определителей матрицы $\widehat{P}$ исходной системы (8.1).

Действительно, пусть такой положительный вектор x найден. Если умножить x на положительный скаляр t, функция Кармаркара не изменится, эта функция постоянна на лучах: $k(tx)=k(x)$. Подбором скаляра t, следовательно, можно вдобавок к (8.5) добиться выполнения условия $\mathbf{1}x=N$ на положительном векторе x. Из неравенства «среднее геометрическое $\leqslant$ среднее арифметическое»

$$(x_1x_2...x_N)^{1/N}\leqslant(x_1+...+x_N)/N=\mathbf{1}x/N=1$$

видно, что знаменатель функции Кармаркара не превосходит единицы. Поскольку числитель этой функции есть сумма неотрицательных квадратичных слагаемых в степени $N/2$, то

$$|p_ix|=|NR\hat{p}_ix-\hat{q}_i\mathbf{1}x|\leqslant 1/[3^{1/N}\Delta(\widehat{P})]$$

для всех i, или с учетом $\mathbf{1}x=N$

$$|R\hat{p}_ix-\hat{q}_i|\leqslant 1/[3^{1/N}N\Delta(\widehat{P})],$$

где $\hat{p}_i$ — строки матрицы $\widehat{P}$, а $\hat{q}_i$ — компоненты вектора $\widehat{Q}$. Возвращаясь к исходным переменным $y=Rx$, получим приближенное решение

* В момент написания брошюры автору их известно шесть.

системы линейных неравенств (8.1) $|\hat{p}_i y-\hat{q}_i|\leqslant\varepsilon_3$, $y_1>0, \ldots, y_N>0$ с точностью $\varepsilon_3=1/[3^{1/N}N\Delta(P)]$, которая, согласно теореме §6 о мере несовместности систем линейных неравенств, достаточна для применения алгоритма «округления». Действительно, если записать систему (8.1) в общем виде $Ay\leqslant b$, то для матрицы ее ограничений будем иметь $\Delta(A)=\Delta(P)$. Поскольку, кроме того, $3^{1/N}N>e^{1/N}N>N+1$, то $\varepsilon_3<\varepsilon_1$ с величиной ε_1, определенной формулой (6.3) в теореме о мере несовместности. Напомним, что с учетом естественного условия $M<N$ (число уравнений можно считать не превышающим числа неизвестных) трудоемкость алгоритма округления не превышает по порядку NM^2 арифметических операций.

Итак, точное решение задачи (8.1) свелось к приближенной минимизации функции $k(x)$ в положительном ортанте $x>0$ с погрешностью (8.5)*.

Зафиксируем произвольную точку $a=(a_1, \ldots, a_N)>0$ в положительном ортанте и рассмотрим принадлежащий неотрицательному ортанту эллипсоид $E(a)$ с центром в точке a и параллельными координатным осям полуосями $a_1, \ldots, a_N$ (левая половина рис. 8, на которой $E(a)$ заштрихован). Эллипсоид $E(a)$, составленный из точек $x=(x_1, \ldots, x_N)$, удовлетворяющих неравенству

$$(x_1-a_1)^2/a_1^2+\ldots+(x_N-a_N)^2/a_N^2\leqslant 1,$$

назовем 1-окрестностью точки a. Если заменить 1 в правой части неравенства на α^2, где α — скаляр из интервала (0,1), то получившееся неравенство будет описывать эллипсоид $\alpha E(a)$ с тем же центром и в α раз меньшими полуосями, который будем называть α-окрестностью точки a. Например, α-окрестностью точки $\mathbf{1}$ с единичными координатами является шар радиуса α с центром $\mathbf{1}$ (правая часть рис. 8, на которой заштрихована 1-окрестность $\mathbf{1}$). При $0<\alpha<1$ любая α-окрестность $\alpha E(a)$ положительной точки a целиком принадлежит положительному ортанту.

Предположив, что задача (8.3) имеет решение, возьмем некоторое α из интервала (0,1) и опишем процедуру $\varnothing$ нахождения в α-окрестности любой положительной точки a, не являющейся решением задачи $k(a)>0$, новой точки $b=\varnothing(a)$, для которой

$$k(b)/k(a)<e^{-2\alpha}/(1-\alpha). \qquad (8.6)$$

В частности, при $\alpha=0{,}5$ правая часть (8.6) принимает значение $2/e=0{,}73\ldots$ и строго меньше единицы. Метод Кармаркара будет состоять в итеративном применении процедуры, и при этом функция Кармаркара будет убывать от итерации к итерации не медленнее геометрической прогрессии со знаменателем $2/e$.

* На практике работают с логарифмом $k(x)$, который Н. Кармаркар называл потенциальной функцией.

Итак, требуется изучить поведение $k(x)$ в α-окрестности положительной точки a. Рассмотрим преобразование подобия («скейлинг») $x'_j=x_j/a_j$, переводящее растяжением координатных осей точку $x=a$ в единичную точку $x'=\mathbf{1}$ (рис. 8). Если обозначить через $\hat{a}=\operatorname{diag}(a_1, a_2, \ldots, a_N)$ матрицу, по диагонали которой записаны числа a_j, а вне диагонали — нули, то в матричном виде преобразование подобия и обратное к нему будут иметь вид $x'=\hat{a}^{-1}x$ и $x=\hat{a}x'$. Преобразование подобия переводит α-окрестность точки a в шар радиуса α с центром в $\mathbf{1}$. Запишем функцию $k(x)$ в новых координатах x':

$$k(x)=k_P(x)=[\Sigma(p_ix)^2]^{N/2}/\Pi(x)=$$

совершаем подстановку $x=\hat{a}x'$

$$=\Sigma(p_i\hat{a}x')^2]^{N/2}/\Pi(\hat{a}x')=$$

учитываем тождество $\Pi(\hat{a}_{x'})=\Pi(a)\Pi(x')$

$$=\{[\Sigma(p_i\hat{a}x')^2]^{N/2}/\Pi(x')\}/\Pi(a)=$$

по определению 8.4 $=k_{P\hat{a}}(x')/\Pi(a)$.

Следовательно, для любых двух точек x и z в α-окрестности a имеем

$$k(x)/k(z)=k_{P\hat{a}}(x')/k_{P\hat{a}}(z'),$$

где x' и z' — соответствующие им точки в α-окрестности $\mathbf{1}$, а $k_{P\hat{a}}$ — функция Кармаркара для задачи (8.3) с матрицей $P\hat{a}$, про которую можно сказать, что она a-подобна P. Далее, поскольку по предположению исходная задача (8.3) с матрицей P имела некоторое решение-луч в неотрицательном ортанте, задача (8.3) с матрицей $P\hat{a}$ также обладает этим свойством. Поэтому можно искать для исходной задачи удовлетворяющую (8.6) точку b в окрестности $\mathbf{1}$, помня, что для нахождения такой точки в окрестности a достаточно заменить P на $P\hat{a}$, найти требуемую точку в окрестности $\mathbf{1}$ и вернуться назад: $b=\varnothing_P(a)=\hat{a}\varnothing_{P\hat{a}}(\mathbf{1})$. Итак, $a=\mathbf{1}$, и нужно изучить поведение функции Кармаркара $k(x)$ в шаре радиуса α с центром в $\mathbf{1}$.

Поскольку $k(x)$ постоянная на лучах $k(tx)\equiv k(x)$, при ее рассмотрении можно ограничиться лишь такими точками $x>0$ в положительном ортанте, для которых $\mathbf{1}x=N$. Действительно, на множестве S — оно называется симплексом — таких точек функция $k(x)$ принимает все те же значения, что и во всем положительном ортанте. Поскольку $\mathbf{1}\in S$, а шар $\alpha E(\mathbf{1})$ целиком лежит в положительном ортанте, пересечение $\alpha E(\mathbf{1})$ с симплексом совпадает с пересечением этого шара с плоскостью $\mathbf{1}x=N$. Таким образом, это пересечение $\alpha E(\mathbf{1})\cap S$ — обозначим его через B — задается двумя условиями:

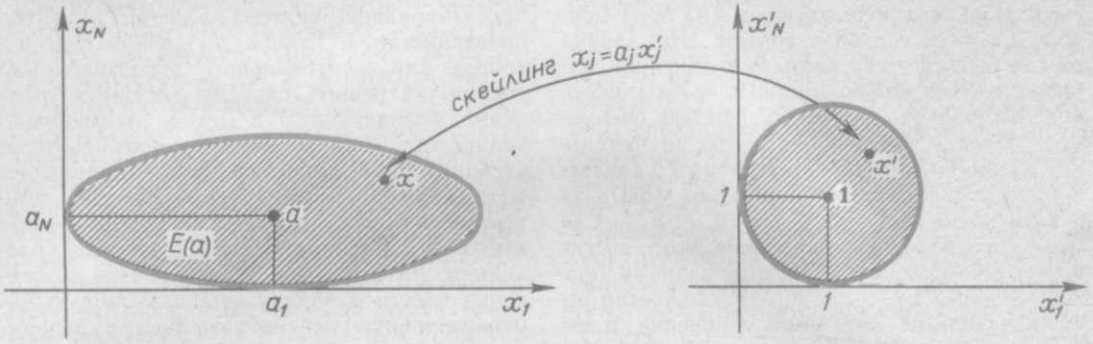


Рис. 8.

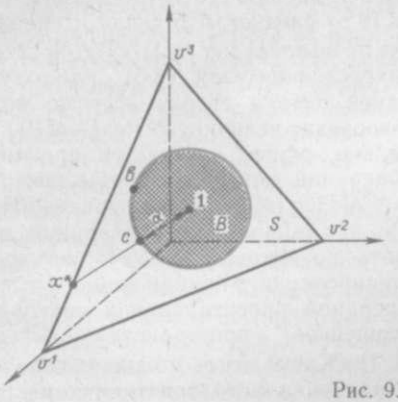


Рис. 9.

$$(x_1-1)^2+\ldots+(x_N-1)^2\leqslant\alpha^2,$$
$$x_1+\ldots+x_N=N \qquad (8.7)$$

и, получаясь в результате сечения N-мерного шара проходящей через его центр плоскостью, само представляет собой $N-1$-мерный шар в плоскости $\mathbf{1}x=N$. На рис. 9 изображен случай $N=3$, когда симплекс S является правильным треугольником с вершинами $(3,0,0)$, $(0,3,0)$, $(0,0,3)$, а B — заштрихованным кругом. Для того чтобы найти в B удовлетворяющую (8.6) точку b, воспользуемся приемом «игнорирования знаменателя» функции Кармаркара на B. Именно возьмем в качестве b решение задачи минимизации стоящей в числителе функции Кармаркара квадратичной формы при ограничениях (8.7)* $x\in B$,

$$f(x)=(p_1x)^2+\ldots+(p_Mx)^2\to\min, \qquad (8.8)$$

Покажем прежде всего, что

$$f(b)\leqslant f(\mathbf{1})\,(1-\alpha/\sqrt{N(N-1)})^2. \qquad (8.9)$$

Действительно, пусть x^* — точное решение задачи (8.3)* в симплексе и

$$c=\mathbf{1}(1-\alpha/\|\mathbf{1}-x^*\|)+x^*\alpha/\|\mathbf{1}-x^*\|$$

— точка из B, лежащая на отрезке $[\mathbf{1},x^*]$ на расстоянии α от $\mathbf{1}$ (рис. 9). Поскольку $f(x^*)=0$ и $c\in B$, то

$$f(b)\leqslant f(c)=f(\mathbf{1})(1-\alpha/\|\mathbf{1}-x^*\|)^2.$$

Для доказательства (8.9) осталось лишь заметить, что для любой точки x симплекса и, в частности, для x^* верно $\|\mathbf{1}-x\|\leqslant\sqrt{N(N-1)}$, поскольку это неравенство выполнено даже для самых далеких от $\mathbf{1}$ точек симплекса — его вершин $(0,\ldots,N,\ldots,0)$.

Займемся теперь знаменателем функции Кармаркара. Оказывается, что на B он не может сильно уменьшаться,

$$\Pi(b)\geqslant\min_{x\in B}\Pi(x)=\left(1-\alpha\sqrt{\frac{N-1}{N}}\right)\times$$
$$\times(1+\alpha/\sqrt{N(N-1)})^{N-1}. \qquad (8.10)$$

Пусть $(\bar{x}_1,\bar{x}_2,\ldots,\bar{x}_N)$ — точка достижения минимума произведения координат $\Pi(x)$ на B. Без ограничения общности можно считать первую координату $\bar{x}_1$ минимальной среди остальных. Поскольку $\mathbf{1}\bar{x}=N$, имеем $\bar{x}=1-\beta$, где β — положительное число. Можно показать, что в точке минимума произведения все оставшиеся координаты должны совпадать между собой: $\bar{x}_2=\ldots=\bar{x}_N=1+\beta/(N-1)$, а сама точка $\bar{x}$ должна находиться на расстоянии α от $\mathbf{1}$, что дает $\beta^2+\beta^2/(N-1)=\alpha^2$ и совпадает с (8.10).

Рассматривая (8.9) и (8.10) вместе, получаем

$$\frac{k(b)}{k(\mathbf{1})}\leqslant\frac{(1-\alpha/\sqrt{N(N-1)})^N}{\left(1-\alpha\sqrt{\frac{N-1}{N}}\right)(1+\alpha/\sqrt{N(N-1)})^{N-1}}<$$
$$<e^{-2\alpha}/(1-\alpha),$$

т. е. для находимой в процедуре «игнорирования знаменателя» точки b действительно выполняется (8.6). Грубо говоря, числитель $[f(x)]^{N/2}$ функции Кармаркара убывает при переходе от $\mathbf{1}$ к b как $(1-\alpha/\sqrt{N(N-1)})^N<$ $<e^{1-\alpha}\approx 1-\alpha$, а знаменатель, достигая в $\mathbf{1}$ максимума на плоскости $\mathbf{1}x=N$, отличается от 1 на B лишь квадратичным по α членом и не может сильно испортить убывание числителя.

Для того чтобы завершить описание метода Кармаркара, осталось лишь рассмотреть способ решения задачи (8.8) по ми-

* Напомним, что при выполнении $\varnothing$ в окрестности a в (8.3) и (8.8) производится замена p_i на $p_i\hat{a}$.

нимизации квадратичной формы на $N-1$-мерном шаре B. Вообще говоря, эта задача может быть решена методами линейной алгебры, однако можно заменить ее на другую, для которой это станет полностью очевидным. Именно предположим, что на начальной итерации метода Кармаркара $\mathbf{1}$ удовлетворяет всем уравнениям системы (8.3), за исключением последнего $p_1\mathbf{1}=0, \ldots, p_{M-1}\mathbf{1}=0$, — как будет показано ниже, этого всегда можно легко добиться. Другими словами, обозначим через $L^{\perp}(p_1, \ldots, p_{M-1})$ множество решений системы линейных уравнений $p_1x=0, \ldots, p_{M-1}x=0$ и предположим, что $\mathbf{1}\in L^{\perp}(p_1, \ldots, p_{M-1})$. Выберем произвольное большое натуральное число F, скажем $F=10^{10}$, и представим себе, что квадратичная форма в определении функции Кармаркара с самого начала задается в виде

$$f(x)=F\cdot[(p_1x)^2+\ldots+(p_{M-1}x)^2]+(p_Mx)^2.$$

Просматривая описание метода начиная с формулы (8.4), можно убедиться в том, что в этом описании ничего не изменится. Ничего не изменится в описании метода и при $F=+\infty$, только теперь все последовательные приближения $\mathbf{1}, \varnothing(\mathbf{1}), \varnothing(\varnothing(\mathbf{1})), \ldots$ метода будут оставаться в линейном пространстве $L^{\perp}(p_1, \ldots, p_{M-1})$, поскольку функция Кармаркара примет вид

$$k(x)=\begin{cases} |p_Mx|^N/\Pi(x), & \text{если } x\in L^{\perp}(p_1, \ldots, p_{M-1}), \\ +\infty, & \text{если } x\notin L^{\perp}(p_1, \ldots, p_{M-1}). \end{cases}$$

При этом задача минимизации числителя функции Кармаркара на B будет представлять собой задачу минимизации $|p_Mx|$ на пересечении $B\cap L^{\perp}(p_1, \ldots, p_{M-1})$. Эта последняя решается ортогональным проектированием вектора p_M на множество решений системы линейных уравнений $p_1x=0, \ldots, p_{M-1}x=0, \mathbf{1}x=0$, т. е. нахождением (единственного) вектора $p\in L^{\perp}(p_1, \ldots, p_{M-1}, \mathbf{1})$, расстояние $\|p-p_M\|$ от которого до p_M минимально. Действительно, в этом случае минимизирующий функцию $|p_Mx|$ на $B\cap L^{\perp}(p_1, \ldots, p_{M-1})$ вектор b имеет вид $b=\mathbf{1}+sp/\|p\|$, где значение параметра $s\in[-\alpha, \alpha]$ при известной проекции p может быть легко вычислено из условия минимума $|p_Mb(s)|$. Как следует из (8.6), при $\alpha=0{,}5$ это обеспечит убывание функции Кармаркара $|p_Mx|^N/\Pi(x)$ на каждой итерации по крайней мере в $2/e$ раз. Если находить значение параметра s из условия минимума функции Кармаркара, или, что то же самое, ее логарифма $\log[k(b(s))]$ при $s\in(-\infty, \infty)$, то сходимость метода может только улучшиться. На практике это улучшение оказывается значительным, и в дальнейшем, говоря о методе Кармаркара, будем иметь в виду именно такой выбор параметра s.

Итак, если $\mathbf{1}\in L^{\perp}(p_1, \ldots, p_{M-1})$, то с учетом скейлинга выполнение итерации $a\leftarrow\varnothing(a)$ метода Кармаркара сводится к ортогональному проектированию вектора $p_M\hat{a}$ на линейное пространство $L^{\perp}(p_1\hat{a}, \ldots, p_{M-1}\hat{a}, \mathbf{1})$. Как известно, для ортогонального проектирования достаточно решить систему линейных уравнений, возникающую в методе наименьших квадратов*. Поэтому вместе с затратами на скейлинг для выполнения одной итерации метода Кармаркара достаточно порядка NM^2 арифметических операций. Наконец, чтобы вектор $\mathbf{1}$ удовлетворял всем линейным уравнениям, за исключением последнего, достаточно ввести в задачу еще одну новую неотрицательную переменную x_{N+1} и переписать систему уравнений $Px=0$ в виде $Px-[P\mathbf{1}]x_{N+1}=0$, $x_{N+1}=0$. Описание метода закончено. Оценим его трудоемкость.

Из (8.6) следует, что число итераций метода не превышает $\log(k(\mathbf{1})/\varepsilon_3)/\log(2/e)$, где ε_3 задается формулой (8.5), или с учетом очевидной оценки сверху $k(\mathbf{1})$ по порядку не превосходит величину $N\log[N\Delta(\widehat{P})]$. Следовательно, общее количество арифметических операций метода не превышает по порядку $N^2M^2\log[N\Delta(\widehat{P})]$. Как показал Н. Кармаркар, эту оценку числа операций можно улучшить по порядку в $N^{1/2}$ раз за счет модифицированного приближенного выполнения операции проектирования вектора $p_M\hat{a}$ на линейное пространство $L^{\perp}(p_1\hat{a}, \ldots, p_{M-1}\hat{a}, \mathbf{1})$. Кроме того, можно показать, что при реализации метода в конечноразрядной компьютерной арифметике для двоичной записи всех фигурирующих в нем чисел достаточно порядка $\log[N\Delta(\widehat{P})]$ разрядов. Вспоминая, что при решении общей задачи линейного программирования с n переменными и m ограничениями в системе (8.1), объединяющей прямую и двойственную задачи, обе величины N и M будут по порядку равны $n+m$, или, что то же самое, $\max(n, m)$, а также учитывая тот факт, что величины $\log[N\Delta(\widehat{P})]$ и $\log[N\Delta(D)]$ также совпадают по порядку, где D — матрица входных коэффициентов задачи линейного программирования, окончательно получаем следующий результат.

Метод Кармаркара является полиномиальным методом линейного программирования. Для точного решения этим методом общей задачи линейного программирования размерности (n, m) достаточно выполнить не более $\mathrm{const}\max^{3.5}(n, m)L$ арифметических операций над числами, имеющими $\mathrm{const}\,L$ разрядов. Здесь под L можно понимать любую из следующих величин:

$\log[\max(n, m)\Delta]$, где Δ — константа, ма-

* Или обратить для линейно-независимых уравнений $i=1, \ldots, M-1$ матрицу Грама $\Gamma=[p_i\hat{a}^2p_j^t]$ и, выкинув из P последнюю строку, вычислить проекцию по формуле $p=(I-\hat{a}P^t\Gamma^{-1}\mathbf{1}\times\times\mathbf{1}^+)\hat{a}p_m$.

жорирующая модули определителей матрицы задачи;

$\min(n, m) \log [\min(n, m) h] + \log \max(n, m)$, где h — высота задачи;

длину входа задачи.

Аналогично методу эллипсоидов при приближенном решении задачи линейного программирования с разумным числом знаков величина L в указанных оценках трудоемкости не превосходит нескольких десятков. Таким образом, если n и m имеют примерно одинаковую величину, то метод Кармаркара по гарантированным оценкам числа операций оказывается по порядку в $n^{1/2}$ быстрее метода эллипсоидов. На практике, однако, даже без применения техники приближенного проектирования, за счет которой и появляется ускорение в корень из размерности раз в теоретических оценках, это превосходство оказывается гораздо существеннее. Дело в том, что, хотя установленная выше теоретическая оценка числа итераций метода Кармаркара растет линейно по размерности и числу требующихся знаков в приближенном решении задачи линейного программирования, имеются сообщения о практических вычислениях, в которых даже для решения больших задач с тысячами неизвестных и ограничений хватает лишь нескольких десятков итераций (хотя, как отмечается в [37], похоже, что так хорошо дело обстоит все-таки не всегда). В настоящее время не ясно, каким образом и насколько может быть улучшена оценка числа итераций метода Кармаркара. Однако, как уже говорилось в первой главе, в 1986 г. был получен следующий важный результат: используя метод Ньютона для приближенной минимизации логарифма некоторой «кармаркароподобной» барьерной функции, Дж. Ренегар [45] построил полиномиальный алгоритм линейного программирования, число итераций которого не превосходит по порядку $\max^{0,5}(n, m)L$, причем по-прежнему каждая итерация метода сводится к решению системы линейных уравнений и может быть выполнена за $\max(n, m) \min^2(n, m)$ операций*. Замечательным свойством оценки числа итераций Ренегара является ее слабая зависимость от размерности задачи: при изменении размерности от сотни до десяти тысяч корень из размерности меняется лишь в пределах от десятка до сотни. Сказанное выше рождает надежду, что в один прекрасный день на вопрос: как сложно решать системы линейных неравенств, можно будет ответить, что в разумном диапазоне размерностей и точностей делать это ненамного сложнее, чем решать системы линейных уравнений.

В 1969 г. Д. Гейл охарактеризовал проблему определения вычислительной сложности линейного программирования как задачу, «которая являлась вызовом исследователям вот уже двадцать лет и остается, по моему мнению, принципиально открытым вопросом теории линейных вычислений» [32]. С тех пор прошло еще почти двадцать лет, и до окончательных ответов по-прежнему далеко.

ЗАКЛЮЧИТЕЛЬНЫЕ ЗАМЕЧАНИЯ

Остановимся вкратце на некоторых других вопросах, так или иначе связанных с линейным программированием и не затронутых в брошюре.

1. Сильнополиномиальные алгоритмы. Можно ли построить алгоритм линейного программирования с полиномиальным по размерности (n, m) числом арифметических операций, которые выполнялись бы над числами полиномиальной по битовой размерности (n, m, l) цифровой длине? Другими словами, можно ли для некоторого алгоритма линейного программирования полиномиальной битовой сложности избавиться от вхождения цифровой длины входных коэффициентов в оценку числа операций? Ответ на этот вопрос о существовании сильнополиномиального алгоритма линейного программирования не известен. Как замечено в [49], с помощью любого из алгоритмов предыдущей главы и некоторого итеративного «округления» входных данных можно построить полиномиальный алгоритм решения задач линейного программирования $cx \to \max$, $Ax \leqslant b$, в оценку числа операций входит лишь цифровая длина коэффициентов матрицы A (но не b и c). Это дает сильнополиномиальные алгоритмы для многих задач линейного программирования, возникающих в дискретной комбинаторной оптимизации, поскольку в этих задачах коэффициенты матрицы A принимают фиксированный набор целочисленных значений, скажем 0, +1 и —1.

2. Нижние оценки. Вопрос о том, каковы могут быть лучшие по числу операций алгоритмы линейного программирования, в настоящее время полностью открыт. Например, неизвестно, можно ли для числа арифметических операций, необходимых для решения общей задачи линейного программирования размерности (n, m), получить нижнюю оценку вида $\text{const} \max(n, m) \min^{1+\delta}(n, m)$ с некоторым положительным δ. Для систем линейных уравнений подозревают, что получить такую нижнюю оценку нельзя. Дело в том, что, как показал в 1969 г. Ф. Штрассен, алгоритм Гаусса не является оптимальным, и общую систему линейных уравнений размерности (n, m) можно решить с затратами $\text{const} \max(n, m) \min^{1+\delta}(n, m)$ арифметических операций при $\delta = 0{,}808$ вместо $\delta = 1$ в алгоритме Гаусса [14, 48]. С тех пор показатель δ неоднократно уменьшался (с увеличением константного множителя), и к настоящему моменту для него достигнуто значение $\delta = 0{,}376$ [26]. Интересно отметить, что эти результаты позволяют уменьшить (см. по-

* О построении полиномиального алгоритма линейного программирования с помощью метода Ньютона см. также работу [33], где дана оценка числа операций $\text{const} \, n^2 m^2 L$. Отметим, что метод Кармаркара также допускает трактовку в виде метода Ньютона для минимизации барьерной функции [34].

следний параграф брошюры) степень размерности в полиномиальных оценках числа операций, необходимых для решения общей задачи линейного программирования, однако из-за одновременного увеличения константных множителей это уменьшение степени малосущественно в разумном диапазоне размерностей.

3. Распараллеливание вычислений. Как сильно можно ускорить решение задач линейного программирования, имея в распоряжении много процессоров? Оставляя в стороне вопросы распараллеливания конкретных методов линейного программирования, остановимся на одной важной открытой проблеме в этой области. Известно, что решение систем линейных уравнений можно распараллеливанием ускорить чрезвычайно сильно: при наличии полиномиального (менее чем кубического) по размерности числа процессоров общую систему линейных уравнений можно решить за время, полиномиальное (квадратичное) по *логарифму* размерности [24],— для невырожденных систем уравнений эта возможность была впервые отмечена Л. Ксанки [14]. Можно ли добиться такого «сверхраспараллеливания» вычислений для решения систем линейных неравенств? Предполагают, что ответ на этот вопрос должен быть отрицательным, поскольку известно, что решение на полиномиальном по L числе процессоров систем неравенств за полиномиальное по $\log L$ время означало бы возможность «сверхраспараллеливания» процесса вычислений любой функции, вычислимой за полиномиальное по длине входа L «однопроцессорное» время.

4. Целочисленное и булево линейное программирование. В огромном числе приложений возникают задачи линейного программирования, неизвестные $x_1, x_2, ..., x_n$ в которых являются целочисленными. Частным случаем таких задач служат задачи с ограничениями $0 \leqslant x_j \leqslant 1$, целочисленные неизвестные в которых оказываются булевыми, т. е. принимают значения 0 или 1. Вопрос о возможности построения алгоритма полиномиальной битовой сложности для решения задач линейного целочисленного, или эквивалентно булева программирования, совпадает с фундаментальной открытой проблемой современной теории сложности вычислений «$P=NP$?» (более подробно об этом см. [10]). Широко распространена гипотеза $P \neq NP$, согласно которой построить такой полиномиальный алгоритм невозможно. Как показал К. Ленстра [42], требуемый полиномиальный алгоритм линейного целочисленного программирования тем не менее можно построить для задач с фиксированным числом неизвестных или ограничений.

Наконец, опишем один широкий класс задач на линейных неравенствах, для которого доказана невозможность построения полиномиального разрешающего алгоритма.

5. Теория вещественных чисел со сложением и порядком. Задачи этой теории — обозначим ее $\langle R, +, \leqslant \rangle$ — являются весьма и весьма далеким обобщением задач, изучаемых в линейном программировании. Они состоят в проверке истинности высказываний вида

$$O_1 x_1 ... O_n x_n F(a_1 x \leqslant b_1, ..., a_m x \leqslant b_m), \qquad (*)$$

где $a_i x \leqslant b_i$ — обычные линейные неравенства с целыми или рациональными коэффициентами относительно вещественных переменных x_j, F — составленное из этих неравенств с помощью связок & (и), $\vee$ (или), $\neg$ (не) предложение, а каждое O_j может быть либо квантором $\exists$ (существует x_j), либо квантором $\forall$ (для любого x_j). Например, если взять в качестве F предложение $a_1 x \leqslant b_1 \& ... \& a_m x \leqslant b_m$, то проверка истинности высказывания $\exists x_1 ... \exists x_n F$ будет означать проверку разрешимости системы линейных неравенств (1.2) в вещественных переменных. Если, кроме того, обозначить через F' предложение $(x_1=0 \vee x_1=1) \& ... \& (x_n=0 \vee x_n=1)$, то для проверки истинности высказывания $\exists x_1 ... \exists x_n F \& F'$ придется проверять разрешимость системы (1.2) в булевых переменных. Проверка истинности высказывания $\forall x_1 ... \forall x_{10} \exists x_{11} ... \exists x_n F$ означает проверку разрешимости системы (1.2) в вещественных переменных $x_{11}, ..., x_n$ при любых фиксированных значениях первых десяти переменных и т. д. — можно легко увеличить число рассмотренных примеров.

Существует общий алгоритм проверки истинности высказываний вида $(*)$, восходящий к идее исключения неизвестных, предложенной еще Ж. Фурье. Суть идеи состоит в следующем. Разрешим каждое неравенство $a_i x \leqslant b_i$ относительно переменной x_n, т. е. перепишем его в виде $\varepsilon_i x_n \leqslant \hat{b}_i(x_1, ..., x_{n-1})$, где $\hat{b}_i(x_1, ..., x_{n-1}) = \alpha_{i1} x_1 + ... + \alpha_{in-1} x_{n-1} + \beta_i$ — линейная функция оставшихся переменных, а ε_i принимает одно из трех значений 0, +1, −1. Представим на мгновение, что $\hat{b}_i$ не зависят от оставшихся переменных, тогда ясно, что бесконечную область изменения переменной $-\infty < x_n < +\infty$ можно заменить на конечное множество точек вида $(\hat{b}_i + \hat{b}_{i'})/2$, $i, i' \in \{1, 2, ..., m\}$, составленное из всевозможных середин интервалов с краями $\hat{b}_i$ и $\hat{b}_{i'}$. Это можно сделать и тогда, когда $\hat{b}_i$ зависят от переменных, только теперь x_n будет пробегать некоторое конечное множество линейных функций от оставшихся переменных. С помощью этого приема можно переписать высказывание $(*)$ на эквивалентное по истинности, в которое, однако, уже не будет входить переменная x_n со своим квантором. Затем можно исключить следующую переменную x_{n-1} и т. д. Истинность получающегося в конце концов предложения без переменных может быть уже легко установлена.

Существенным недостатком описанного алгоритма является быстрое нарастание числа неравенств и битовой длины записи высказывания $(*)$ по мере исключения неизвестных. Этот эффект в процедуре исключения переменных является неустранимым уже для систем линейных неравенств, и оказывается, что даже при экономной реализации алгоритма [29] ему требуется экспоненциальный по длине входа L объем памяти. Разумеется, это означает неполиномиальность процедуры исключения неизвестных. Однако, как следует из одной общей теоремы М. Фишера и М. Рабина [30], любой разрешающий алгоритм для теории $\langle R, +, \leqslant \rangle$ имеет по крайней мере экспоненциальную сложность $2^{\text{const}\cdot L}$, и поэтому полиномиальные алгоритмы определения истинности высказываний $(*)$ не существуют. Справедливости ради следует отметить, что это верно и для теории $\langle R, + \rangle$, в которой высказывания вида $(*)$ строятся на линейных равенствах. Задачи вида $(*)$, следовательно, являются чересчур общими и будут оставаться сложными всегда.

ЛИТЕРАТУРА

1. Ашманов С. А. Линейное программирование.— М.: Наука, 1981.

2. Брейман Л. Задачи о правилах остановки. Прикладная комбинаторная математика / Сб. статей под ред. Э. Беккенбаха.— М.: Мир, 1968.

3. Вершик А. М., Спорышев П. В. Оценки среднего числа шагов симплекс-метода и задачи асимптотической интегральной геометрии: Доклады АН СССР. 1983.— 271.— № 5.

4. Воеводин В. В. Вычислительные основы линейной алгебры.— М.: Наука, 1977.

5. Воеводин В. В. Ошибки округления и устойчивость в прямых методах линейной алгебры.— М.: Изд-во МГУ, 1969.

6. Гантмахер Ф. Р. Теория матриц.— М.: Физматгиз, 1953.

7. Гасс С. Линейное программирование.— М.: Физматгиз, 1961.

8. Гольштейн Е. Г., Юдин Д. Б. Линейное программирование, теория, методы и приложения.— М.: Наука, 1969.

10. Гэри М., Джонсон Д. Вычислительные машины и труднорешаемые задачи.— М.: Мир, 1982.

11. Данциг Д. Линейное программирование, его обобщения и применение.— М.: Прогресс, 1966.

12. Канторович Л. В. Экономический расчет наилучшего использования ресурсов.— М.: Изд-во АН СССР, 1966.

13. Пападимитриу Х., Стайглиц К. Комбинаторная оптимизация. Алгоритмы и сложность.— М.: Мир, 1985.

14. Солодовников В. И. Верхние оценки сложности решения систем линейных уравнений. Теория сложности вычислений. I. Записки ученых семинаров ЛОМИ.— Ленинград.: Наука, 1982.— 1.

15. Хачиян Л. Г. Полиномиальные алгоритмы в линейном программировании // Журнал вычислит. математ. и мат. физики.— 1980.— 20.— № 1.

16. Хачиян Л. Г. О точном решении систем линейных неравенств и задач линейного программирования. Там же.— 1982.— 22.— № 4.

17. Хачиян Л. Г. Выпуклость и алгоритмическая сложность решения задач полиномиального программирования // Изв. АН СССР, Тех. киб.— 1982.— № 6.

18. Черников С. Н. Линейные неравенства.— М.: Наука, 1968.

19. Шор Н. З. Метод отсечения с растяжением пространства для решения задач выпуклого программирования // Кибернетика.— 1977.— № 1.

20. Юдин Д. Б., Немировский А. С. Информационная сложность и эффективные методы решения выпуклых экстремальных задач.— Экономика и мат. методы, XII, № 2, 1976.

21. Adler I., Megiddo N. A simplex algorithm whose average number of steps is bounded between two quadratic functions of the smaller dimention, in: Proceedings of the 16th Annual Symposium on Theory of Computing. — N. Y., 1984.

22. Adler I., Megiddo N., Todd M. J. New results on the average behavior of simplex algorithms, Bulletin of the American Math. Society, 11, No. 2. 1984.

23. Bland R. G. New finite pivoting rules, Math. Oper. Res., 3, 1978.

24. Borodin A., von zur Grathen J., Hopcroft J. E. Fast parallel matrix and GCD computations, Information and Control, 52, 1982.

25 Borgwardt K.—H., The average nember of steps reguired by the simplex method is polynomial, Zeitschrift fur Operations Research, Ser A—B, 26, 1982.

26. Coppersmith D., Winograd S. Matrix multiplication via arithmetic progression, Dept. of Math. Sci, IBM Thomas J. Watson Res. Centr., Preprint, Nov. 1986.

27. Dantzig G. B. Reminiscences about the origins of linear programming, in: „Mathematical Programming. The State of the Art", Bachem A., Grötschel M. and Korte B. eds, Springer Verlad, Berlin, 1983.

28. Dixon J. D. Exact solutions of linear equations using p-adic expansion, Numer. Math, 40, 1982.

29. Ferrante J., Rackoff. A decision procedure for the first order theory of real addition with order, SIAM J. Comput., 4, No 1, 1975.

30. Fisher M., Rabin M. O. Super exponential complexity of Presburger arithmetic, Project MAC Tech. Mem. 43, Mass Inst. of Tech, Cambridge 1974.

31. Fourier J. B. J. Analyse de travaux de l'[1]. Academie Rovale des Sciences pendant l'année 1823, Partie Mathématique, Histoire de l Academie Royale des Sciences de l'Institut de France, 6, XXIV-xli, 1826.

32. Gale D. How to solve linear inequalities, American Mathematical Monthly, 76, 1969.

33. De Gellinck G., Vial J.-P. A polynomial Newton method for linear programming, Algorithmica, 1, No 4, 1986.

34. Gill P. E., Murray W., Saunders M. A., Tomlin J. A., Wright M. H. On projected Newton barrier methods for linear programming and an equivalence to Karmarkar's projective method, Math. Programming, 36, 1986

35. Goldfarb D., Sit W. Y. Worst case behavior of the steepest edge simplex method, Discrete Applied Math., 1, 1979.

36. Haimovich M. The simplex algorithm is very good.— On the expected number of pivot steps and related propeties of random linear programs, manuscript, Columbia University, N. Y., 1983.

37. Iri M., Imai H. A multiplicative barrier function method for linear programming, Algorithmica, 1, No 4, 1986.

38. Jeroslow R. G. The simplex algorithm with the pivot rule of maximizing criterion improvement, Discrete Math., 4, 1973.

39. Karmarkar N. A new polynomical-time algorithm for linear programming, Combinatorica, 4, 1984.

40. Klee V., Minty G. J. How good is he simplex algorithm?, Inequalities, III, O. Shisha ed, Academic Press, N. Y, 1972.

41. Kotiah C. T., Steinberg D. I. On the possibility of cycling with the simplex method. Oper. Res, 26, 1976.

42. Lenstra H. W. Jr. Integer programming with a fixed number of variables, Math. Oper. Res., 8, 1983.

43. Megiddo N. Towards a genuinely polynomial algorithm for linear programming, SIAM J. Comput., 12, 1983.

44. Megiddo N. Linear programming in linear

time when the dimention is fixed, J. of the ACM, 31, 1984.

45. Renegar J. A polynomial-time algorithm, based on Newton's method for linear programming, Math. Sci. Res. Institute, Berkeley, Calif., preprint 07118—86, June 1986.

46. Smale S. The problem of the average speed of the simplex method, in: „Mathematical Programming. The State of the Art" Bachem A., Grötschel M. and Korte B. eds, Springer Verlag, Berlin, 1983.

47. Schrijver A. The new linear programming method of Karmarkar, CWI Newsletter, No. 8., September, 1985.

48. Strassen V. Gaussian elimination is not optimal, Numer. Math, 13, 1969.

49. Tardos E. A strongly polynomial algorithm to solve combinatorial linear problems, Report 84360 — OR, Institut fur Ökonometrie and Operations Research, University of Bonn, January 1985

50. Todd M. J. Polynomial expected behavior of a pivoting algorithm for linear complementarity and linear programming problems, Math. Programming, 35, 1986.

Замечание. Во время работы над корректурой брошюры автор получил от Ю. Е. Нестерова известие о том, что ему удалось построить полиномиальный алгоритм линейного программирования с кубическим по размерности (n, m) и линейным по L числом операций.

Научно-популярное издание

Леонид Генрихович ХАЧИЯН

СЛОЖНОСТЬ ЗАДАЧ
ЛИНЕЙНОГО ПРОГРАММИРОВАНИЯ

Главный отраслевой редактор Л. А. Ерлыкин
Редактор Г. Г. Карвовский
Мл. редактор М. А. Гаврилова
Художник Л. П. Ромасенко
Худож. редактор М. А. Бабичева
Техн. редактор И. Е. Жаворонкова
Корректор В. В. Каночкина

ИБ № 8672

Сдано в набор 18.08.87. Подписано к печати 02.10.87. Т-00787. Формат бумаги 70×100¹/₁₆. Бумага кн.-журнальная. Гарнитура «Литературная». Печать офсетная. Усл. печ. л. 2,60. Усл. кр.-отт. 5,52. Уч.-изд. л. 3,77. Тираж 36 216 экз. Заказ 2316. Цена 11 коп. Издательство «Знание». 101835, ГСП, Москва, Центр, проезд Серова, д. 4. Индекс заказа 874310.
Ордена Трудового Красного Знамени
Чеховский полиграфический комбинат ВО «Союзполиграфпром» Государственного комитета СССР по делам издательств, полиграфии и книжной торговли
142300, г. Чехов Московской области