ПАРАЛЛЕЛЬНЫЕ АЛГОРИТМЫ ЦЕЛОЧИСЛЕННОЙ ОПТИМИЗАЦИИ

В. И. ХОХЛЮК

27 июня 2007 г.

ПРЕДИСЛОВИЕ

Задачи целочисленной оптимизации возникают в различных приложениях и теоретических исследованиях. Настоящая монография посвящена изучению численных методов целочисленной оптимизации с точки зрения их распараллеливания. Интерес к параллельным вычислениям обусловлен следующими тремя основными причинами:

параллельные алгоритмы необходимы параллельным вычислительным системам, уже созданным и проектируемым;

переход от привычных последовательных вычислений к параллельным открывает новые возможности для построения алгоритмов;

распараллеливание вычислений позволяет во много раз увеличить скорость счёта.

В первых двух главах книги даётся общая характеристика задач и методов, которые составляют предмет целочисленной оптимизации, и приводятся математические модели часто встречающихся прикладных задач. Эти модели представляют собой задачи оптимизации, в которых ограничения, как правило, линейные, а все переменные или их часть целочисленные.

Важнейшие преобразования задач целочисленной оптимизации содержатся в главе 3. Практическое значение преобразований заключается в том, что они дают возможность свести огромное количество разнообразных задач к небольшому числу стандартных задач, для решения которых уже имеются программные средства.

Далее рассматриваются основные принципы построения параллельных алгоритмов (гл. 4) , параллельные алгоритмы метода ветвей и границ для решения задач комбинаторной и целочисленной оптимизации (гл. 5) и параллельные алгоритмы метода секущих плоскостей. Для нахождения оптимального решения целочисленной и смешанной целочисленной линейных задач (гл. 6).

В последних главах анализируются и сравниваются различные вычислительные схемы для приведения целочисленной матрицы к специальному виду (гл. 7) и приводятся краткие сведения о десяти вспомогательных АЛГОЛ процедурах для решения семи математических задач, встречающихся в качестве подзадач в методах целочисленной оптимизации.

Чтобы облегчить чтение монографии и активное усвоение излагаемого материала (по возможности без обращения к другим источникам), в книгу включены некоторые методы преобразования и решения задач целочисленной оптимизации (в достаточно общей форме), ряд последовательных алгоритмов и разнообразные примеры, формулируемые в виде упражнений для самостоятельной работы.

На вопрос, является ли распараллеливание алгоритмов магистральным путём целочисленной оптимизации, автор ответил бы следующим образом. В настоящее время, по-видимому, нет. Однако в ближайшем будущем параллельные вычисления, безусловно, станут преобладать в численных методах целочисленной оптимизации и её приложениях, как и в других разделах вычислительной математики. Для того чтобы это произошло, предстоит выполнить огромную работу в областях математики и программирования, высокопроизводительной вычислительной техники и электроники. Высказанное мнение автора разделяют и другие исследователи.

Ограниченный объём книги не позволил рассмотреть в ней вопросы параллелизма в таких разделах оптимизации, как преобразование задач комбинаторной оптимизации, анализ вычислительной сложности задач целочисленной и комбинаторной оптимизации, теория секущих плоскостей и др.

В заключение автор выражает глубокую благодарность рецензентам книги профессорам Б. А. Головкину и Д. Б. Юдину за содержательные критические замечания, а также всем, кто помогал ему в работе над книгой.

Глава 1

ЗАДАЧИ И МЕТОДЫ ЦЕЛОЧИСЛЕННОЙ ОПТИМИЗАЦИИ

В этой главе в сжатой форме рассматриваются ключевые вопросы теории, численных методов и приложений целочисленной оптимизации.

1.1 ПОСТАНОВКА ЗАДАЧ

Задача оптимизации. Пусть X — заданное множество элементов x произвольной природы, $f : X \to R$ — заданное отображение множества X в множество R всех действительных чисел. Тогда задача минимизации абстрактно может быть сформулирована следующим образом: либо найти элемент $x^*, x^* \in X$, который минимизирует функцию $f(x)$, либо установить его отсутствие. Если существует такой элемент x^*, то $f(x^*) \leq f(x)$ для всех $x \in X$.

Аналогично ставится задача максимизации. Каждая из этих задач также называется задачей оптимизации. Задача оптимизации, возникающая при принятии решений, представляет собой математическую модель реального процесса принятия решения, в которой среди возможных решений x, составляющих совокупность X, ищется решение $x*$, наилучшее в смысле заданной функции $f(x)$.

Задачу оптимизации будем записывать одним из следующих способов:

найти элемент x^*, который минимизирует $f(x)$ при условии $x \in X$;

минимизировать $f(x)$ при условии $x \in X$;

$$\min_{x \in X} f(x);\ \min\{f(x) \mid x \in X\}.$$

Если из контекста ясно, о чем идет речь, то слова "минимизация", "максимизация", "оптимизация" будут опускаться, а также будут опускаться прилагательные перед словом "задача" (например, линейная, нелинейная, целочисленная, смешанная, смешанная целочисленная). В приложении "Основные обозначения и сокращения" приведена таблица названий задач оптимизации.

Значение $f(x^*)$, элемент x^* из X, элемент x из X, множество X, отображение f называются соответственно значением (оптимальным значением), решением (оптимальным решением), допустимым решением, множеством допустимых решений (допустимым множеством), целевой функцией задачи оптимизации.

Природа элементов x множества X, способ задания множества X и вид отображения f определяют конкретный класс задач оптимизации. Такой класс задач для краткости будем называть задачей оптимизации, а элемент из этого класса — конкретной задачей оптимизации. Если не возникает недоразумений, то слово "конкретная" опускается. Рассмотрим комбинаторную, целочисленную и смешанную целочисленную задачи оптимизации.

Под комбинаторной задачей оптимизации понимаем задачу оптимизации, в которой множество X состоит из конечного числа элементов, а природа самих элементов зависит от задачи. Оптимизационные задачи на перестановках, различные оптимизационные задачи на конечном графе могут служить примерами комбинаторных задач. Например, комбинаторная формулировка задачи о назначениях имеет вид

$$\max\Big\{f(x)=\sum_{i=1}^{n} c_{i,j_i} \mid x\in X\Big\},$$

где X — множество всех перестановок $x=(j_1,j_2,\dots,j_n)$ чисел $1,2,\dots,n$, $C=(c_{i,j})$ — $n\times n$-матрица с неотрицательными элементами $C_{i,j}$ ($i=\overline{1,n};\ j=\overline{1,n}$). Конкретное значение величины n и конкретные значения элементов матрицы C определяют конкретную задачу о назначениях.

Пусть X — некоторое конечное или счётное множество точек $x=(x_1,x_2,\dots,x_r)$ со всеми целочисленными координатами x_j, $j=\overline{1,r}$. Тогда получаем целочисленную задачу оптимизации. Можно задать множество X как совокупность всех целочисленных решений некоторой системы линейных (нелинейных) неравенств. В этом случае получаем целочисленную задачу оптимизации с линейными (нелинейными) ограничениями.

Теперь рассмотрим множество X, состоящее из точек $(x,y)=(x_1,x_2,\dots,x_r,y_1,y_2,\dots,y_s)$, где x_j принимает только целочисленные значения ($j=\overline{1,r}$), а y_j изменяется в интервале $a_j\le y_j\le b_j$, $a_j<b_j$ ($j=\overline{1,s}$). Тогда получаем смешанную целочисленную задачу оптимизации. В ней присутствуют как целочисленные переменные x_j, так и непрерывные переменные y_j. Аналогично характер ограничений, которым удовлетворяют точки (x,y), определяет смешанную целочисленную задачу оптимизации с линейными (нелинейными) ограничениями.

Комбинаторная, целочисленная и смешанная целочисленная задачи оптимизации тесно связаны друг с другом, имеют общую методологию и практически необозримую область приложений. Основное внимание будет уделено целочисленной линейной и смешанной целочисленной линейной задачам оптимизации (см. (1.1) и (1.2), хотя многие результаты переносятся как на нелинейный случай, так и на комбинаторные задачи.

Условия целочисленности и дискретности. Условие целочисленности естественным образом возникает в тех случаях, когда нецелочисленное значение переменной не имеет смысла.

Сформулируем целочисленную линейную и смешанную целочисленную линейную задачи оптимизации

$$\min\{cx \mid Ax\ge b,\ x\ge 0,\ x\equiv 0\}, \tag{1.1}$$

$$\min\{cx+dy \mid Ax+By\ge b,\ x\ge 0,\ x\equiv 0,\ y\ge 0\}. \tag{1.2}$$

Здесь $m\times r$-матрица A, m-столбец b, $m\times s$-матрица B, r-строка c, s-строка d — неуправляемые переменные параметры задачи, а $x=(x_1,x_2,\dots,x_r)$,$y=(y_1,y_2,\dots,y_s)$ — столбцы управляемых переменных ($r\ge 1, s\ge 1$; либо $n=r$, либо $n=r+s$). Выражение $x\equiv 0$ читается "x сравнимо с 0" и означает, что все компоненты $x_1,x_2,\dots,x_r$ вектора x принимают только целочисленные значения. Задача (1.1) более кратко называется либо Ц-задачей (задачей IP), либо ЦЛ-задачей (задачей ILP), а задача (1.2) — либо СЦ-задачей (задачей MIP), либо СЦЛ-задачей (задачей MILP).

Опуская условие целочисленности $x\equiv 0$ в задачах (1.1) и (1.2), получаем линейные релаксации этих задач, т. е. линейные задачи оптимизации

$$\min\{cx \mid Ax\ge b,\ x\ge 0\},$$

$$\min\{cx+dy \mid Ax+By\ge b,\ x\ge 0,\ y\ge 0\}.$$

Линейная задача оптимизации, в частности и линейная релаксация, более кратко называется Л-задачей (задачей L или задачей LP).

Если в задачах (1.1) и (1.2) заменим условия $x_j\ge 0, x_j\equiv 0$ на условия дискретности $x_j\in D_j$, где D_j — дискретное множество действительных чисел ($j=\overline{1,r}$), то получим соответственно дискретную линейную и смешанную дискретную линейную задачи.

Условия бинарности и дизъюнктивности. Бинарные переменные $x_j, x_j=0,1$, позволяют записать самые разнообразные факты, например: дискретность переменной, кусочно-линейную аппроксимацию и линейное представление нелинейной функции, логические условия, бинарную формулировку оптимизационных комбинаторных задач (см., в частности, § 2.1, 2.4, 2.11 и 3.11).

Сформулируем бинарную линейную и смешанную бинарную линейную задачи оптимизации

$$\min\{cx \mid Ax \geq b,\ x_j = 0,1,\ j = \overline{1,r}\}, \tag{1.3}$$

$$\min\{cx + dy \mid Ax + By \geq b,\ x_j = 0,1,\ j = \overline{1,r},\ y \geq 0\} \tag{1.4}$$

Задача (1.3) более кратко называется либо Б-задачей (задачей BP), либо БЛ-задачей (задачей BLP), а задача (1.4) — либо СБ-задачей (задачей MBP), либо СБЛ-задачей (задачей MBLP).

Условие бинарности $x_j = 0,1$ равносильно отношениям $0 \leq x_j \leq j, x_j \equiv 0$. Это позволяет представить задачи (1.3) и (1.4) соответственно в виде задач (1.1) и (1.2).

Линейными релаксациями задач (1.3) и (1.4) являются соответственно задачи

$$\min\{cx \mid Ax \geq b,\ 0 \leq x \leq e\},$$

$$\min\{cx + dy \mid Ax + By \geq b,\ 0 \leq x \leq e,\ y \geq 0\},$$

где $e = (1, 1, \ldots, 1) \in R_r$.

Рассмотрим задачу оптимизации, допустимое множество X которой задаётся при помощи логического выражения. В этом логическом выражении в качестве элементарных высказываний выступают линейные неравенства, а сами высказывания связаны логическими операциями, такими как дизъюнкция, конъюнкция, импликация и др. Известно, что всякое логическое выражение можно привести к дизъюнктивной нормальной форме. Проделав это, получим дизъюнктивную задачу

$$\min\Big\{cx \mid \bigvee_{h \in H} (A^h x \geq b^h,\ x \geq 0)\Big\}.$$

Эта задача более кратко называется либо задачей D, либо задачей DP.

При формулировке дизъюнктивной задачи оптимизации было использовано соглашение о том, что всякое высказывание, выступающее в качестве условия, всегда является истинным (см. Приложение "Основные обозначения и сокращения").

Очевидно, что высказывание $(A^h x \geq b^h,\ x \geq 0)$ эквивалентно высказыванию $(\bigwedge_{i=1}^{m_h}(a_i^h x \geq b_i^h)) \bigwedge (\bigwedge_{j=1}^{r}(x_j \geq 0))$ (здесь a_i^h — i-я строка матрицы A^h). При этом высказывание $(A^h x \geq b^h, x \geq 0)$ истинно для всех x, удовлетворяющих системе неравенств $A^h x \geq b^h, x \geq 0$, и ложно для всех x, не удовлетворяющих этой системе. Присутствие дизъюнкции в задаче DP может привести к невыпуклому допустимому множеству

$$X = \Big\{x \mid \bigvee_{h \in H} (A^h x \geq b^h,\ x \geq 0)\Big\} = \bigcup_{h \in H} \{x \mid A^h x \geq b^h,\ x \geq 0\}.$$

Задачи BLP, MBLP и DP тесно связаны друг с другом.

Групповые задачи. Групповые задачи оптимизации возникают в целочисленной оптимизации как групповая релаксация ЦЛ- и СЦЛ-задач

$$\min\{cx \mid Ax = b,\ x \geq 0,\ x \equiv 0\},$$

$$\min\{cx + dy \mid Ax + By = b,\ x \geq 0,\ x \equiv 0,\ y \geq 0\},$$

где A, B, b, c, d целочисленные. Эта релаксация состоит в исключении условий неотрицательности на часть переменных. Любое справедливое неравенство для множества допустимых решений релаксации также является справедливым и для допустимого множества исходной задачи.

Групповая релаксация ЦЛ-задачи, называемая целочисленной групповой задачей (задачей IGP), формулируется в виде

$$\min\Big\{\sum_{j=1}^{r} c'_j x_j \mid \sum_{j=1}^{r} h_j x_j = h_0,\ x_j \in Z^+,\ j = \overline{1,r}\Big\}. \tag{1.5}$$

Здесь $H = \varphi(G)$ — гомоморфный образ группы G, порождённой целочисленными столбцами $a_1, a_2, \ldots, a_r$ матрицы $A, h_0 = \varphi(b)$, $h_j = \varphi(a_j)$, $h_j \in H$, φ — заданный гомоморфизм ($j = \overline{1,r}$). Кратное $x_j h_j =$

$h_j x_j$ и сумма определяются групповой операцией в H. Группа H может быть как конечной, так и бесконечной.

Групповая релаксация СЦЛ-задачи, называемая смешанной целочисленной групповой задачей (задачей MIGP), имеет вид

$$\min\Big\{\sum_{j=1}^{r} c'_j x_j + \sum_{j=1}^{s} d'_j y_j \Big| \sum_{j=1}^{r} \varphi(a_j)x_j + \varphi\Big(\sum_{j=1}^{s} b_j y_j\Big) = \varphi(b),\ x_j \in Z^+, j = \overline{1,r},\ y_j \geq 0,\ j = \overline{1,s}\Big\}. \quad (1.6)$$

Здесь φ — заданный гомоморфизм аддитивной группы R_m всех действительных m-векторов (групповая операция — обычное сложение векторов) на единичный гиперкуб U, где $U = \{u \mid 0 \leq u_i < 1,\ i = \overline{1,t}\}$, а групповая операция состоит в покомпонентном сложении по модулю 1.

Лагранжева релаксация. Для рассмотрения этой релаксации запишем задачу (1.2) в несколько изменённом виде

$$\min\{cx \mid Ax \geq b,\ x \geq 0,\ x_j \equiv 0,\ j \in N_1 \subseteq N\}, \quad (1.7)$$

где $N = \{1, 2, \ldots, n = r + s\}$.

Разбиваем систему ограничений $Ax \geq b$ задачи (1.7) на две подсистемы $A^1x \geq b^1, A^2x \geq b^2$, исключаем ограничения $A^2x \geq b^2$ и переносим их с множителями u в целевую функцию. В результате получаем лагранжеву релаксацию

$$l(u) = \min\{(c - uA^2)x + ub^2 \mid A^1x \geq b^1,\ x \geq 0,\ x_j \equiv 0,\ j \in N_1 \subseteq N\}. \quad (1.8)$$

Для любого $u, u \geq 0$, величина $l(u)$ является нижней границей для значений целевой функции задачи (1.7). Задачу максимизации $l(u)$ при условии $u \geq 0$ иногда называют лагранжевой двойственной задачей для (1.7). Существует несколько методов максимизации $l(u), u \geq 0$, один из них — субградиентная оптимизация. Если вектор $\overline{u}$ максимизирует $l(u)$, а вектор $\overline{x}$ — соответствующее оптимальное решение задачи (1.8), то $\overline{x}$ является оптимальным решением задачи (1.7), если $A^2\overline{x} \geq b^2$ и $\overline{u}(A^2\overline{x} - b^2) = 0$. Однако это обычно не имеет места, так как значение $l(\overline{u})$ и оптимальное значение задачи (1.7) имеют тенденцию разделяться так называемым разрывом двойственности. Тем не менее, поскольку вычисление значения $1(u)$ для фиксированного и может быть намного легче, чем решение задачи (1.7), это вычисление часто оказывается удобным способом нахождения хороших нижних границ.

Если $A^2x \geq b^2$ частично или полностью состоит из сечений, то лагранжева релаксация даёт способ использования последних без значительного возрастания числа неравенств, явно добавляемых к системе ограничений.

Детали по использованию лагранжевой техники можно найти в работах, ссылки на которые находятся, например, в [57].

Преобразование задачи оптимизации. Всякая задача обладает как формой, так и содержанием. Содержание задачи определяется природой и конкретными значениями управляемых и неуправляемых переменных величин, в то время как под формой понимается структура задачи, заданная перечнем ее управляемых и неуправляемых переменных величин и взаимосвязью этих величин, т. е. математическая модель.

Так, например, задача о назначениях может быть представлена, по крайней мере, в четырёх формах: как оптимизационная комбинаторная задача на множестве всех перестановок, как бинарная линейная задача оптимизации; как линейная задача оптимизации, как задача о максимальном взвешенном паросочетании на двудольном неориентированном графе.

Под преобразованием задачи оптимизации понимаем какую-либо операцию над ее элементами (переменными, ограничениями, целевой функцией, множеством допустимых решений), в результате выполнения которой получается новая задача оптимизации. После выполнения преобразования задачи могут измениться число переменных, число и вид ограничений, целевая функция, множество допустимых решений, условия целочисленности и т. д. Наконец, преобразованная задача может быть сформулирована в терминах, отличных от терминов исходной задачи. Переход от начальной задачи к конечной может состоять из нескольких последовательно выполняемых простейших преобразований. Например, переход от задачи оптимизации к её релаксации является преобразованием задачи.

1.2 ЧИСЛЕННЫЕ МЕТОДЫ

Решение задач целочисленной оптимизации. Целочисленная задача (1.1) иногда может быть решена как линейная: решаем ее линейную релаксацию LP и получаем оптимальное решение этой линейной задачи, удовлетворяющее условию целочисленности. Это можно сделать тогда, когда все базисные решения линейной задачи LP целочисленные. Хорошо известно, что для произвольного целочисленного вектора b система ограничений $Ax \geq b$, $x \geq 0$ имеет только целочисленные базисные решения, если и только если матрица A является абсолютно унимодулярной (т. е. все невырожденные подматрицы матрицы A имеют определитель либо -1, либо $+1$) [20].

Самым известным примером абсолютно унимодулярной матрицы является вершинно-рёберная матрица инциденций либо ориентированного графа, либо двудольного неориентированного графа. Как следствие, задача о потоке минимальной стоимости и её частные случаи (например, транспортная задача, задача о максимальном потоке, задача о кратчайшем пути), задачи о рёберной упаковке (покрытии) и о вершинной упаковке (покрытии) на двудольном неориентированном графе, а также другие целочисленные задачи, матрицей ограничений которых служит абсолютно унимодулярная матрица, по существу являются линейными задачами [1, 58].

Не говоря о классе абсолютно унимодулярных задач, важном, но очень специальном, подчеркнём здесь трудность численного решения задач целочисленной оптимизации, которая состоит в том, что множество допустимых решений этих задач является невыпуклым. Именно невыпуклость не позволяет установить глобальный оптимум из локальных условий. Указанная трудность преодолевается двумя различными способами — методом ветвей и границ и методом секущих плоскостей.

Следует также упомянуть о двух процедурах разбиения, которые не принадлежат ни одному из этих методов, а скорее дополняют их. Обе процедуры расчленяют смешанную целочисленную задачу (1.2): первая — с помощью разбиения переменных, вторая же — с помощью разбиения ограничений. Используя первую процедуру, принадлежащую Бендерсу [58], избавляемся от непрерывных переменных путём проектирования допустимого множества X задачи в подпространство целочисленных переменных и переходим к эквивалентной задаче (см. § 3.9). Применяя вторую процедуру, исключаем часть основных ограничений задачи (1.2), переносим их с множителями в целевую функцию и получаем задачу, называемую лагранжевой релаксацией (смотри § 1.1).

Ниже кратко охарактеризуем метод ветвей и границ. Метод секущих плоскостей, а также алгоритмы, полиномиальные и неполиномиальные, точные и приближённые, последовательные и параллельные.

Метод ветвей и границ. Этот метод представляет собой перебор и является стандартным способом решения задач на конечном множестве. Укажем основные составные части этого метода: последовательно дробим допустимое множество задачи на все меньшие подмножества; на каждом подмножестве вычисляем нижнюю и верхнюю границы для неизвестного оптимального значения целевой функции; используя вычисленные границы, из дальнейшего рассмотрения исключаем определённые подмножества. Процесс заканчивается, когда-либо для каждого подмножества уже произведено его наилучшее решение, либо каждое подмножество не содержит ни одного лучшего решения, чем уже имеющиеся решения. Наилучшее решение, найденное в процессе перебора, и является глобальным оптимумом. Для иллюстрации метода ветвей и границ используется дерево, называемое деревом перебора.

Опишем метод ветвей и границ для решения конкретных целочисленных задач (1.1), (1.2), обозначив их через P. Пусть $v(P)$ — оптимальное значение задачи P.

Заносим задачу P в список подзадач. Полагаем $\bar{v}(P) = +\infty$, где $\bar{v}(P)$ — верхняя граница для $v(P)$.

Шаг 1. Если список подзадач пуст, то останавливаемся: если не было найдено никакого решения, то допустимое множество задачи P пусто, иначе текущее наилучшее решение является оптимальным. По некоторому правилу, заданному стратегией поиска, из списка выбираем подзадачу P_k и удаляем её из этого списка.

Шаг 2. Если подзадача P_k имеет ограничения, содержащие только бинарные переменные, то при помощи логических тестов исследуем импликации этих ограничений, чтобы выделить по возможности больше новых ограничений вида либо $x_j = 0$, либо $x_j = 1$ (либо более сложного вида). Если допустимое множество подзадачи P_k оказывается пустым, то отбрасываем её и переходим к шагу 1.

Шаг 3. Вычисляем нижнюю границу $\underline{v}(P_k)$ для $v(P_k)$, решая некоторую релаксацию подзадачи P_k

(либо линейную релаксацию, либо лагранжеву релаксацию, либо одну из этих релаксаций, улучшенную сечениями). Если $\underline{v}(P_k) \geq \overline{v}(P_k)$, то отбрасываем подзадачу P_k и переходим к шагу 1. Если же $\underline{v}(P_k) < \overline{v}(P_k)$ и оптимальное решение релаксации удовлетворяет условию целочисленности, то обновляем верхнюю границу $\overline{v}(P)$ и текущее допустимое решение, удаляем из списка все подзадачи P_s с $\underline{v}(P_s) \geq \overline{v}(P)$ и переходим к шагу 1.

Шаг 4. Используя некоторую эвристику, пытаемся вычислить улучшенную верхнюю границу для $v(P)$ и найти улучшенное допустимое решение. Если это удаётся, то обновляем верхнюю границу $\overline{v}(P)$ и текущее допустимое решение, удаляем из списка все подзадачи P_s с $\underline{v}(P_s) \geq \overline{v}(P)$ и переходим к шагу 1.

Шаг 5. Разветвляем задачу P_k, на две или больше подзадач, разбивая множество её допустимых решений по некоторому заданному правилу. Добавляем новые подзадачи к списку и переходим к шагу 1.

Стратегии поиска, используемые на шаге 1, находятся между двумя крайностями, известными как "только вглубь" (всегда выбираем одну из новых подзадач, только что полученных (одноветвевая схема перебора)), и как "только вширь" (всегда выбираем подзадачу с наибольшим $\underline{v}(P_k)$ (многоветвевая схема перебора)). Одноветвевая схема перебора требует мало памяти и используется в большинстве работающих программ, реализующих последовательные алгоритмы. Многоветвевая схема перебора в каждый момент хранит информацию о многих ветвях и является очень эффективной в параллельных алгоритмах. По-видимому, именно промежуточные стратегии поиска дают наилучшие результаты.

Убедительно было показано, что логические тесты и присоединённые неравенства шага 2 значительно ускоряют процесс решения.

Самыми важными составными частями любого алгоритма ветвей и границ являются способы вычисления границ на шагах 3 и 4. Важность используемой релаксации была продемонстрирована при решении такой задачи со специальной структурой, как задача о коммивояжёре. Знание глубоких секущих плоскостей (обычно содержащих $(n-1)$-мерные грани выпуклой оболочки множества допустимых решений) позволило заменить линейную релаксацию более сильной релаксацией, полученной либо добавлением к линейной релаксации неравенств, соответствующих этим глубоким плоскостям, либо включением их в целевую функцию лагранжевой релаксации.

Аналогично использование эффективной эвристики для нахождения верхней границы и допустимого решения на шаге 4 резко сокращает время счёта.

Правило разветвления (разбиения) на шаге 5 обычно представляет собой дихотомию вида $x_j \leq \lfloor \bar{x_j} \rfloor \vee x_j \geq \lceil \bar{x_j} \rceil$, где значение $\bar{x_j}$ целочисленной переменной x_j в оптимальном решении релаксации задачи P_k не является целочисленным. Выбор переменной важен, однако для этого неизвестно ни одного надежного правила. В работающих программах обычно выбор переменной опирается на штрафы. Штраф представляет собой нижнюю границу для изменения целевой функции и может быть вычислен различными способами. Выбирается переменная с наибольшим штрафом.

В задачах со специальной структурой возможны более эффективные правила разветвления. Например, при наличии ограничения $\sum\limits_{j \in Q} x_j = 1, x_j = 0, 1, j \in Q$, можно разветвлять по дихотомии либо $x_j = 0, j \in Q_1$, либо $\sum\limits_{j \in Q_1} x_j = 1, x_j = 0, 1, j \in Q \backslash Q_1$ для некоторого подмножества $Q_1 \subset Q$, сразу фиксируя несколько переменных. Другие более утончённые правила разветвления были использованы в задаче о покрытии (разбиении) множества и в задаче о коммивояжёре.

Методу ветвей и границ посвящены, например, работы [54, 57, 67, 69].

Метод секущих плоскостей. Этот метод используется для решения конкретных целочисленных линейных задач ((1.1) и (1.2), обозначаемых P.

Выпуклая оболочка $\mathrm{conv}X$ допустимого множества X целочисленной задачи P аппроксимируется пересечением конечного числа полупространств, задаваемых такими линейными неравенствами, которые отсекают части многогранника линейной релаксации L без удаления каких-либо точек из X (отсюда название "метод секущих плоскостей"). Когда построено достаточное количество неравенств, чтобы отсечь все допустимые решения линейной задачи L, лучшие по сравнению с оптимумом целочисленной задачи, последний находится как оптимальное решение линейной задачи L, полученной из L путём добавления к ней построенных неравенств.

Один из способов решения целочисленной задачи P состоит в следующем. Начинаем с оптимального решения её линейной релаксации L и последовательно расширяем систему ограничений задачи

L добавлением новых неравенств (сечений) до тех пор, пока не будет отсечена вся область между оптимумами линейной и целочисленной задач. Количество вычислений, которое потребуется для этого, зависит от силы (глубины) сечений, а также от размера отсекаемой области, т. е. от величины разрыва между $v(L)$ и $v(P)$ ($v(L)$ и $v(P)$ — оптимальные значения задач L и P). Этот разрыв на самом деле может быть очень большим. Известны случаи, когда разрыв оценивается величиной $v(P)/v(L) = O(n)$, где n — число переменных в задачах L и P.

Эта часть целочисленной оптимизации богата теоретическими результатами, связанными с другими разделами математики.

Методу секущих плоскостей посвящены, например, работы [57, 58, 69].

Точные и приближённые алгоритмы. Построение оптимального решения задачи осуществляется при помощи алгоритма. Под алгоритмом будем понимать конечную совокупность действий (инструкций), выполнение которых преобразует исходные данные задачи в искомый результат.

Алгоритмы подразделяют на точные и приближённые, полиномиальные и неполиномиальные. Для характеристики точного алгоритма, как правило, достаточно знать только два параметра: трудоёмкость (сложность) t_A алгоритма A — число элементарных операций, выполняемых при реализации алгоритма A; память r_A алгоритма A — количество единиц памяти ЭВМ, необходимых для работы алгоритма A. Под элементарной операцией понимаем либо одну из арифметических операций (сложение, вычитание, умножение, деление двух чисел), либо сравнение двух чисел. Обычно рассматривают зависимость параметров алгоритма от длины записи входных данных.

Будем называть алгоритм полиномиальным, если для его трудоёмкости существует верхняя граница, имеющая полиномиальную (степенную) зависимость от длины записи входных данных. В противном случае будем считать алгоритм неполиномиальным.

Например, существуют полиномиальные алгоритмы для решения задачи о максимальном потоке, задач о рёберной упаковке (покрытии) и о вершинной упаковке (покрытии) на двудольном неориентированном графе.

Результаты теории вычислительной сложности, применённые к задачам дискретной оптимизации, открыли широкий класс задач, называемых NP-полными, относительно которых известно, что-либо все задачи этого класса разрешимы полиномиальным алгоритмом, либо ни одна задача из этого класса не разрешима полиномиальным алгоритмом. По-видимому, вторая альтернатива более правдоподобна, так как многие трудные задачи дискретной оптимизации, как показал их анализ, являются NP-полными. В частности, классу NP-полных задач принадлежат бинарная линейная задача (1.3), задача о коммивояжёре, задачи о покрытии и разбиении множества, задача о вершинной упаковке на произвольном n-вершинном неориентированном графе и многие другие задачи. Список NP-полных задач довольно большой, и их число продолжает расти [2, 16, 36].

Как показывают исследования, попытки построения точных алгоритмов для решения NP-полных задач оптимизации приводят к неполиномиальным алгоритмам. Поэтому построение точных полиномиальных алгоритмов для таких задач кажется бесперспективным. В последнее время большое внимание уделяется разработке приближённых алгоритмов. Жертвуя оптимальностью, пытаемся получить малотрудоёмкий алгоритм.

Для приближённого алгоритма, кроме трудоёмкости и памяти, вводят такие параметры, как относительная погрешность ε_A алгоритма A и вероятность несрабатывания δ_A алгоритма A.

1.3 ВЫЧИСЛИТЕЛЬНАЯ СЛОЖНОСТЬ ЗАДАЧ ЦЕЛОЧИСЛЕННОЙ ОПТИМИЗАЦИИ

Говоря об алгоритмах решения задач целочисленной оптимизации, нельзя не касаться вопроса о качестве этих алгоритмов. К числу важнейших характеристик алгоритма могут быть отнесены такие величины, как количество элементарных операций и объём памяти, необходимые для реализации этого алгоритма.

Время выполнения и простота реализации также являются важными характеристиками алгоритма, особенно при решении практических задач.

Экспериментальный способ является очевидным путём, который как-то позволяет оценить вычислительную трудоёмкость задач целочисленной оптимизации. Существующими алгоритмами можно

решить некоторое число конкретных задач и сравнить эти алгоритмы эмпирически. После статистической обработки результатов эксперимента можно получить информацию о времени счёта как функцию от структуры и размеров задачи (например, числа переменных и ограничений). При изложении многих алгоритмов приводятся результаты машинного счёта. Систематизация и анализ вычислительного опыта по решению задач целочисленной оптимизации дают достаточно полную картину о возможностях численных методов с учётом имеющихся вычислительных средств.

В этом параграфе кратко рассмотрим два теоретических способа оценки вычислительной сложности задач оптимизации.

Пусть K — класс конкретных задач оптимизации, H — класс конечных алгоритмов, каждый из которых находит оптимальное решение любой задачи из класса K. Через $t(x,y)$ обозначим число элементарных операций, необходимых для решения задачи $x, x \in K$, алгоритмом $y, y \in H$. Поскольку функция $t(x,y)$, вообще говоря, является неизвестной, находим для данного алгоритма y верхнюю границу $\overline{z}, \overline{z}(y) = \sup\limits_{x \in K} t(x,y)$, и нижнюю границу $\underline{z}, \underline{z} \leq \inf\limits_{y \in H} \sup\limits_{x \in K} t(x,y)$. Если $z(y^*) = z$, то алгоритм y^* называется оптимальным для класса K.

Даже в случае, когда $\overline{z}(y)$ — точная верхняя граница, эта величина может оказаться очень пессимистической оценкой сложности алгоритма y, так как она может отразить результаты решения "патологических", а не типичных задач из класса K Например, при решении практических линейных задач оптимизации симплекс-методом было установлено экспериментальным путём, что число итераций симплекс-метода является линейной функцией от числа основных ограничений задачи. Однако известны теоретические результаты, которые говорят о том, что число итераций может возрастать и экспоненциально от числа ограничений.

Если в задачах, рассматриваемых на n-вершинном графе, каждое ребро (дуга) является кандидатом на включение в оптимальное решение, то каждое ребро должно быть проверено, по крайней мере, один раз. Таким образом, нижняя граница сложности задаётся полиномом второй степени от n, т. е. имеет вид $O(n^2)$. Алгоритм Дийкстра для нахождения кратчайших путей между выделенной вершиной и всеми остальными вершинами в графе, ребрам (дугам) которого приписаны положительные длины, имеет трудоёмкость $O(n^2)$ и, следовательно, является оптимальным относительно порядка сложности. Существуют также алгоритмы сложности $O(n^2)$ для нахождения кратчайших путей в ациклическом ориентированном графе. Различные алгоритмы для более общей задачи о кратчайшем пути имеют порядок $O(n^3)$, но они необязательно оптимальны.

В настоящее время очень мало известно о точных нижних границах и об оптимальных алгоритмах.

Рассмотрим другой способ оценки качества алгоритма.

Класс P-задач. Будем называть алгоритм полиномиальным для некоторого класса задач, если верхняя граница его сложности есть полином от длины кодировки входных данных каждой задачи из этого класса. Эта идея лежит в основе теории вычислительной сложности, которая позволяет классифицировать задачи относительно точно определённой меры сложности.

Стандартная кодировка входных данных задачи представляет собой конечную последовательность нулей и единиц. Длина кодировки l равна числу нулей и единиц в этой последовательности. Задача принадлежит классу P (Polynomial) относительно своей кодировки, если для её решения существует полиномиальный алгоритм.

Способ кодировки является существенным для получения правильного результата. Например, рассмотрим бинарную линейную задачу

$$\min\{cx \mid Ax \geq b,\ x_j = 0,1,\ j = \overline{1,n}\}.$$

Предположим, что кодировка включает данные A, b, c, а также 2^n бинарных векторов. Рассмотрим алгоритм, который для каждого бинарного вектора по очереди определяет его допустимость, вычисляет значение целевой функции и сохраняет наилучшее допустимое решение. Очевидно, что относительно такой кодировки рассматриваемый алгоритм является полиномиальным. Однако вряд ли такой алгоритм можно считать эффективным для решения бинарной линейной задачи, поскольку для этой задачи существует более естественная и гораздо меньшая кодировка. Следовательно, решение таким наивным алгоритмом недостаточно для включения бинарной линейной задачи в класс P.

Граф может быть закодирован при помощи своей матрицы инциденций (или матрицы смежности), целое число — его бинарным представлением, а целочисленный вектор или целочисленная матрица — как список целых чисел.

Важно различать бинарную кодировку целого числа m, для которой $l = \lfloor \log_2 m \rfloor + 1$, и его унарную кодировку (т. е. с помощью одного символа), для которой $l = m$. Чтобы увидеть это, рассмотрим задачу о ранце

$$\max\Big\{\sum_{j=1}^{n} c_j x_j \Big| \sum_{j=1}^{n} a_j x_j = b,\ x_j \in Z^+,\ j = \overline{1,n}\Big\},$$

где a_j, b, c_j — положительные целые числа ($j = \overline{1,n}$). Известно, что эта задача может быть представлена как задача о кратчайшем пути в ациклическом ориентированном графе с $b+1$ вершинами (см., например, § 3.6). Если задача о ранце кодируется как задача о кратчайшем пути, то можно заключить, что существует алгоритм сложности $O((b+1)^2)$. Однако если входные данные задачи, состоящие из $2n+1$ целых чисел, кодируются с помощью бинарного представления, то длина кодировки растет, скажем, уже как $\log_2 b$ и сложность этого алгоритма не является полиномом от l. Следовательно, преобразование задачи о ранце в задачу о кратчайшем пути не даёт основания включить задачу о ранце в класс P.

В классе P находятся задача о кратчайшем пути, задача о назначениях, задача о максимальном потоке и другие задачи.

Класс NP-задач. Результаты С. А. Кука и Р. М. Карпа вызывают сомнение по поводу существования полиномиального алгоритма для решения бинарной линейной и целочисленной линейной задач. Поскольку для точного изложения этих результатов необходимы понятия теории вычислительной сложности, кратко опишем лишь основную идею.

Предположим, что решается бинарная линейная задача перечислением 2^n бинарных n-векторов на дереве перебора. Рассмотрим процесс, который, будучи поставленным перед выбором одной из двух альтернатив (либо $x_j = 0$, либо $x_j = 1$) на стадии j (для других задач число конечных альтернатив может быть больше), может создать две копии самого себя и проследить последовательность вычислений по обоим способам действия. Если бы сложность каждой стадии была полиномом от n, то сложность полного перебора была бы полиномом от n, так как глубина дерева перебора (число уровней) равна n.

Множество задач, которые могут решаться полиномиальным алгоритмом при помощи такого процесса, составляет класс NP (Nondeterministic Polynomial). Задача находится в классе NP, если и только если она может быть решена алгоритмом перебора на дереве, глубина которого есть полином от n. Конечно, $P \subseteq NP$. Класс NP является весьма обширным и содержит все целочисленные линейные задачи оптимизации с конечным числом допустимых решений, а также многие оптимизационные комбинаторные задачи, включая задачу о коммивояжёре. Заметим, что класс NP-полных задач содержится в классе NP.

Теории вычислительной сложности посвящены, например, работы [2, 16, 36, 44, 57].

1.4 ЭЛЕМЕНТЫ ТЕОРИИ СЕКУЩИХ ПЛОСКОСТЕЙ

Центральным вопросом теории секущих плоскостей является характеризация выпуклой оболочки множества X допустимых решений целочисленной и смешанной целочисленной задач

$$\min\{cx \mid Ax = b,\ x \geq 0,\ x \equiv 0\},$$

$$\min\{cx + dy \mid Ax + By = b,\ x \geq 0,\ x \equiv 0,\ y \geq 0\}.$$

Обозначим обе задачи через P. Под выпуклой оболочкой множества X понимаем наименьшее выпуклое множество, содержащее X. Обозначим её через $\mathrm{conv}X$ (convex — выпуклый). Из теории выпуклых множеств известно, что для рациональных данных выпуклая оболочка $\mathrm{conv}X$ есть пересечение конечного числа полупространств. Другими словами, целочисленная задача P эквивалентна линейной задаче. К сожалению, однако, система ограничений этой линейной задачи в общем случае трудно определима. К настоящему моменту только для малого числа оптимизационных комбинаторных задач, имеющих очень специальную структуру, все-таки известна линейная характеризация множества $\mathrm{conv}X$, т. е. точное представление множества $\mathrm{conv}X$ системой линейных неравенств. В методе секущих плоскостей используются несколько процедур для построения последовательности неравенств, которые дают некоторое приближение к такому представлению.

Линейное неравенство $\pi x \geq \pi_0$ называется справедливым для множества X, если оно выполняется для всех точек x из X. В целочисленной оптимизации справедливое неравенство также называют сечением, а соответствующую ему плоскость — секущей плоскостью. Что касается силы (глубины) различных секущих плоскостей, то полезно иметь в виду следующее.

Пусть $X \subset R_n, d \in R_n, d_0 \in R$. Множество $\{x \mid x \in \mathrm{conv}X,\ dx = d_0\}$ называется $(n-1)$-мерной гранью множества $\mathrm{conv}X$, если $dx \geq d_0$ для всех $x \in X$ и $dx = d_0$ для n аффинно независимых точек x из X. Грани выпуклой оболочки $\mathrm{conv}X$ важны, поскольку среди её многих возможных представлений именно неравенства, определяющие грани множества $\mathrm{conv}X$, дают минимальное представление и самые сильные возможные секущие плоскости.

Субаддитивные сечения. Рассмотрим целочисленную задачу $\min\{cx \mid Ax = b,\ x \geq 0,\ x \equiv 0\}$, обозначив её через P. Решая симплекс-методом линейную релаксацию L задачи P, получаем оптимальное базисное представление

$$x_i = y_{i,0} - \sum_{j \in J} y_{i,j} x_j,\ i \in I,$$

где I и J — индексные множества соответственно базисных и небазисных переменных.

Если $y_{i,0}$ не является целым числом, то можно показать, что целочисленность переменной x_i, неотрицательность и целочисленность переменных x_j, $j \in J$, i-е уравнение оптимального представления влекут неравенство $\sum\limits_{j \in J} f_{i,j} x_j \geq f_{i,0}$ которое имеет место для всех допустимых решений целочисленной задачи P, но нарушается, например, для оптимального базисного решения линейной задачи $L(f_{i,j} = y_{i,j} - \lfloor y_{i,j} \rfloor, j \in (0 \cup J))$. Следовательно, это неравенство является сечением. Оно было основой для метода целых форм, первого конечно сходящегося алгоритма секущих плоскостей для решения целочисленной линейной задачи [58].

Аналогичное сечение даёт, конечно, сходящийся алгоритм для решения смешанной целочисленной линейной задачи (при условии, что целевая функция принимает целочисленные значения).

Целочисленная задача над оптимальным конусом, ограничения которой задаются условиями $x_j \equiv 0$, $j \in (I \cup J)$, $x_j \geq 0$, $j \in J$ (заметим, что опущены условия $x_j \geq 0$, $j \in I$), эквивалентна целочисленной групповой задаче минимизации (1.5) над конечной абелевой группой H. Последняя может быть решена как задача о кратчайшем пути [58]. Если точка $\overline{x}$, соответствующая найденному оптимальному решению групповой задачи, удовлетворяет условиям $x_j \geq 0$, $j \in I$, то она является оптимальным решением задачи P. Если же это не имеет места, то точка $\overline{x}$ даёт нижнюю границу $c\overline{x}$ для $v(P)$.

Основное понятие в характеризации углового многогранника, то есть выпуклой оболочки целочисленных точек конуса, есть субаддитивность. Это впоследствии привело к субаддитивной характеризации выпуклой оболочки множества X допустимых решений задачи P.

Функция $f(v)$, определённая на моноиде V (коммутативной полугруппе по сложению, содержащей нуль полугруппы 0), называется субаддитивной, если $f(v+w) \leq f(v) + f(w)$ для всех v, w из V. Пусть A — $m \times n$-матрица с целочисленными элементами, a_j — j-й столбец матрицы A и $X = \{x \mid Ax = b,\ x \geq 0,\ x \equiv 0\} \neq \varnothing$. Тогда для любой субаддитивной функции $f(v)$, определённой на моноиде $V = \{v \mid v = Az,\ \exists z \in Z_n^+\}$ и удовлетворяющей условию $f(0) = 0$, каждая точка x из X удовлетворяет неравенству $\sum\limits_{j=1}^{n} f(a_j) x_j \geq f(b)$ (субаддитивное сечение).

Обратно если $\pi x \geq \pi_0$ является справедливым неравенством для множества X, то существует такая субаддитивная функция $f(u)$, определённая на моноиде V и удовлетворяющая условию $f(0) = 0$, что $f(a_j) \leq \pi_j$, $j = \overline{1,n}$, $f(b) \geq \pi_0$.

Субаддитивная характеризация имеет место для выпуклой оболочки допустимых решений линейной и смешанной целочисленной линейной задач (см., например, [57, 73]).

Дизъюнктивные сечения. Под дизъюнктивным сечением понимаем справедливое неравенство для множества X (или $\mathrm{conv}X$) допустимых решений дизъюнктивной задачи

$$\min\Big\{cx \mid \bigvee_{h \in H} A^h x \geq b^h,\ x \geq 0)\Big\}.$$

Для выпуклой оболочки $\mathrm{conv}X$ известны следующие два результата. Пусть H^* — множество тех индексов h из H, для которых система $A^h x \geq b^h, x \geq 0$ совместна, а $\pi \in R_n$, $\pi_0 \in R$. Тогда каждая точка

$x \in X$ удовлетворяет неравенству $\pi x \geq \pi_0$, если и только если существует такое множество векторов $\lambda^h \in R_{m_h}$, $\lambda^h \geq 0$, $h \in H^*$, что для всех $h \in H^*$ имеют место неравенств $\lambda^h A^h \leq \pi$ и $\lambda^h b^h \geq \pi_0$.

Предположим, что выпуклая оболочка conv$X$ является n-мерной, множество H — конечным, а $\pi_0 \neq 0$. Тогда неравенство $\pi x \geq \pi_0$ задаёт $(n-1)$-мерную грань множества conv$X$, если и только если $\pi, \pi \neq 0$, есть вершина многогранника

$$F' = \{\pi \mid \pi \geq \lambda^h A^h \ \exists \lambda^h \geq 0, \ h \in H^*\}.$$

Первый из этих результатов даёт способ построения справедливых неравенств, требующий мало вычислений. Второй же результат указывает на то, что любое сечение может быть усилено за счёт увеличения вычислений до момента, когда оно задаёт $(n-1)$-мерную грань множества conv$X$.

В теории секущих плоскостей одним из основных является следующий вопрос. Можно ли построить выпуклую оболочку допустимых решений задачи, последовательно учитывая условия целочисленности? Как показывают исследования дизъюнктивной задачи, ответ для бинарной задачи (1.3) положителен, а для целочисленной задачи (1.1) отрицателен.

Комбинаторные сечения. Для задачи о максимальном взвешенном паросочетании существует линейная характеризация выпуклой оболочки её допустимых решений (см., например, § 3.10, [26]).

К сожалению, многовершинник паросочетании есть исключение, а не правило. Для большинства комбинаторных задач не существует такой простой линейной характеризации выпуклой оболочки допустимых точек. Однако для различных задач известны определённые классы $(n-1)$-мерных граней выпуклой оболочки.

Для задачи о ранце и задачи о коммивояжёре были охарактеризованы различные классы $(n-1)$-мерных граней выпуклой оболочки допустимых точек.

Теории секущих плоскостей посвящены, например, работы [20, 57, 58, 65, 73].

1.5 ПАРАЛЛЕЛЬНЫЕ ВЫЧИСЛЕНИЯ

Необходимость обработки огромных массивов информации за реальное время поставила перед наукой и техникой ряд сложных задач. Эти задачи успешно решаются усилиями многих исследователей.

Средствами параллельной обработки информации являются параллельные вычислительные системы, параллельное программирование и параллельные вычислительные методы. При распараллеливании можно выделить три узловых этапа прохождения задачи: разработку метода, программирование и вычисление.

Параллельные вычислительные системы. Любая параллельная вычислительная система (вычислительная система параллельной обработки информации) имеет более одного устройства управления или более одного центрального обрабатывающего устройства, которые работают одновременно. Повышенный интерес к параллельным вычислительным системам и их значение определяются тем, что они обеспечивают высокие показатели производительности, надежности, готовности, а также гибкости и живучести.

В развитии архитектуры вычислительных систем и их программного обеспечения можно выделить две основные тенденции: совмещение разных процессов обработки информации в одних и тех же устройствах для повышения эффективности использования ресурсов системы (мультипрограммирование, разделение времени); распределение одного процесса обработки информации между разными устройствами для ускорения его выполнения (мультиобработка, распараллеливание).

В настоящее время все универсальные ЭВМ работают в режиме разделения времени, а также созданы многопроцессорные вычислительные системы, в которых реализован принцип распараллеливания обработки. Для высокопроизводительных универсальных ЭВМ требуется совмещение обоих принципов обработки, позволяющее обеспечить, с одной стороны, высокую скорость обработки отдельных программ, а с другой — высокую производительность системы в целом с эффективным использованием её ресурсов. Таким образом, мультипрограммирование и мультиобработка не исключают, а дополняют друг друга. При этом оба принципа, хотя и выглядят как противоположные, имеют много общего с точки зрения проблем, возникающих при их практической реализации.

Архитектура универсальных параллельных вычислительных систем (с мультипрограммированием и мультиобработкой) существенно более сложна, чем архитектура последовательных ЭВМ первых

поколений. В таких системах гораздо сложнее процессы обработки информации и организация параллельных программ. Традиционные методы проектирования ЭВМ и программирования либо неприемлемы для параллельных вычислительных систем, либо неэффективны. Поэтому уже в течение длительного времени ведутся интенсивные исследования в области архитектуры параллельных вычислительных систем и параллельного программирования, Этой области посвящена обширная литература.

В [14], например, рассматриваются многомашинные, многопроцессорные, магистральные (конвейерные), ассоциативные и матричные системы, системы с векторным потоком данных и ансамбли процессоров, а также системы с комбинированной и перестраиваемой структурой. В описании приводятся структурные схемы и основные характеристики нескольких десятков отечественных и зарубежных параллельных вычислительных систем.

Работа [37] главным образом посвящена вопросам построения, функционирования и программного обеспечения параллельных вычислительных систем с общим управлением.

Параллельное программирование. Под термином "параллельное программирование" понимаем написание параллельных программ на специальном параллельном языке программирования для параллельной вычислительной системы. В параллельном программировании можно выделить два основных подхода, взаимно дополняющих друг друга: непосредственное написание параллельных программ на основе параллельных алгоритмов и автоматизацию параллельного программирования (методы автоматического распараллеливания последовательных программных конструкций и синтез параллельных программ) [74].

Работа [62], например, посвящена организации параллельных вычислений на многопроцессорных вычислительных комплексах. В ней рассматриваются проблемы параллелизма в архитектуре, языках программирования и системном математическом обеспечении современных и перспективных многопроцессорных вычислительных комплексов, а также вопросы синтеза параллельных алгоритмов и программ, методы автоматического распараллеливания вычислений.

Параллельные вычислительные методы. Эти методы включают математическую постановку задачи, выбор или разработку параллельного метода её решения и составление параллельного алгоритма. При разработке вычислительных методов для параллельных вычислительных систем рассматриваются следующие вопросы:

способы преобразования последовательных алгоритмов в параллельные, организация вычислений при наличии разнотипных операций, оптимизация структур данных при последовательном выполнении цепочки алгоритмов, распределение данных между параллельно работающими процессорами, эффективность распараллеливания вычислений и т. д.

Вычисления на параллельных вычислительных системах особенно эффективны при реализации параллельных методов, предназначенных для решения задач линейной алгебры, математической физики, статистики, экономики и т. д. Например, разработаны параллельные алгоритмы для решения таких задач линейной алгебры, как умножение матриц, обращение матрицы, решение систем линейных уравнений методом Гаусса и методом Зейделя. Распараллеливанию вычислений в линейной алгебре посвящено много работ, например [48, 49].

Распараллеливание вычислений в целочисленной оптимизации. При решении задач целочисленной оптимизации очень часто возникают ситуации, допускающие распараллеливание вычислений. В этих ситуациях либо выполняются операции над векторами и матрицами (например, сложение двух векторов или матриц), либо вычислительный процесс представляется в виде нескольких ветвей дерева. В обоих случаях выполнение вычислений допускает естественное распараллеливание. В других же случаях при помощи различных приемов и построений можно ввести параллелизм даже в некоторые процессы, являющиеся по своей природе сугубо последовательными. При этом, конечно, должен сохраняться разумный баланс между параллелизмом и количеством вычислений. Считаем, что размеры задач, по крайней мере, на порядок больше числа процессоров параллельной вычислительной системы.

Перечислим некоторые основные разделы целочисленной оптимизации, в которых распараллеливание вычислений особенно эффективно [39, 54, 55]: 1) простейшие преобразования; 2) преобразования задач оптимизации; 3) алгоритмы метода ветвей и границ; 4) алгоритмы метода секущих плоскостей; 5) алгоритмы динамического программирования; 6) преобразования целочисленной матрицы.

Распараллеливанию вычислений посвящены, например, работы [4, 10, 13, 14, 17–19, 28, 37, 39, 40, 43, 48, 49, 54, 55, 60, 62, 79, 82].

1.6 МАШИННАЯ РЕАЛИЗАЦИЯ

Все имеющиеся коммерческие программы для решения задач целочисленной оптимизации основаны на методе ветвей и границ. Они, как правило, находят допустимое решение приемлемого качества для задач с сотнями целочисленных переменных и тысячами непрерывных переменных, иногда дают оптимальные решения для таких задач. Однако нельзя гарантировать, что за разумное время будет решена любая задача, количество целочисленных переменных в которой больше 30.

В настоящее время алгоритмы секущих плоскостей для решения целочисленной и смешанной целочисленной линейных задач являются неустойчивыми и медленными, чтобы конкурировать с методом ветвей и границ. Однако при решении задач со специальной структурой (таких, как задача о покрытии множества, задача о коммивояжёре), в которых легко получаются сильные сечения, использование секущих плоскостей оказалось очень успешным как в переборе, так и в лагранжевой технике.

Часть практических задач решается имеющимися программами. Однако решение большого количества важных задач находится за пределами возможностей программ, реализующих точные алгоритмы. Для решения таких задач разрабатываются приближённые методы.

Сведения о программах можно найти, например, в [67, 76]. Применение пакетов прикладных программ по экономико-математическим методам рассмотрено, например, в [38, 45]. Пакет прикладных программ ДИСПРО предназначен для решения дискретных задач [32], а пакет ПЛАНЕР — для решения специальных классов задач производственно-транспортного планирования, имеющих большую размерность [33].

Внедрение автоматизированных систем управления стимулирует прикладные и теоретические исследования по целочисленной оптимизации. Вопросы разработки математического обеспечения ЭВМ освещены, например, в работах [7, 8, 39, 50–52, 56].

Глава 2

ЦЕЛОЧИСЛЕННЫЕ ФОРМУЛИРОВКИ ПРИКЛАДНЫХ ЗАДАЧ

В этой главе для часто встречающихся прикладных задач, имеющих общепризнанное практическое значение, приводятся математические формулировки. Эти математические формулировки представляют собой задачи оптимизации, в которых ограничения, как правило, линейные, а все переменные или их часть целочисленные.

Следует отметить, что ситуации, порождающие одну и ту же задачу оптимизации, могут быть самыми разнообразными, а их количество практически необозримо. Здесь же рассматривается только одна из таких возможных ситуаций.

Потенциальная возможность построения параллельного вычислительного метода для решения какой-либо задачи часто определяется структурой этой задачи. Поэтому для более детального представления структуры задачи в данной главе, например, не используются матричные обозначения и сохраняются индексы, хотя в некоторых случаях это не упрощает изложения.

Круг задач, допускающих формулировку с использованием целочисленных переменных, чрезвычайно широк. Достаточно указать на то, что любая нелинейная задача оптимизации с ограниченными переменными может быть представлена как бинарная линейная (с какой угодно точностью). Переход к целочисленной формулировке зачастую достигается за счёт введения огромного числа новых ограничений и новых переменных.

При этом возникают следующие вопросы. Целесообразна ли целочисленная формулировка рассматриваемой задачи? Какие факторы определяют, является ли данная формулировка задачи естественной или искусственной? Каким образом для конкретной задачи выбрать самый подходящий метод её решения? Эти вопросы очень трудны и находятся в стадии изучения. Постепенно вырабатываются определённые рекомендации, однако при их использовании следует проявлять некоторую осторожность.

2.1 ЗАПИСЬ УСЛОВИЙ В БИНАРНЫХ ПЕРЕМЕННЫХ

Переменная x, принимающая только два значения — 0 и 1, называется бинарной или двоичной, а также $0,1$-переменной, булевой и бивалентной. Значение 0 бинарной переменной x, как правило, означает отсутствие какого-нибудь свойства, а значение 1 — наличие этого свойства.

Если каждый элемент матрицы (вектора) равен либо 0, либо 1, то такая матрица (вектор) называется бинарной. Например, векторы $(0,0,\ldots,0),(1,0,\ldots,0),(1,1,\ldots,1)$ и единичная $n \times n$-матрица I_n являются бинарными. Используя бинарные переменные, можно записать самые разнообразные условия.

Бинарное представление переменной. Пусть целочисленная переменная x_j, принимает значения $0,1,2,\ldots,d_j$ (d_j — положительное целое число, $d_j < +\infty$).

Тогда её можно представить тремя различными способами:

$$x_j = \sum_{k=0}^{t_j} 2^k y_{j,k} \text{ при условиях } \sum_{k=0}^{t_j} 2^k y_{j,k} \le d_j,\ y_{j,k} = 0,1,\ k = \overline{0, t_j},\ \text{где } 2^{t_j} \le d_j < 2^{t_j+1};$$

$$x_j = \sum_{k=1}^{d_j} y_{j,k} \text{ при условиях } y_{j,k} = 0,1,\ k = \overline{1, d_j};$$

$$x_j = \sum_{k=1}^{d_j} k y_{j,k} \text{ при условиях } \sum_{k=1}^{d_j} y_{j,k} \le 1,\ y_{j,k} = 0,1,\ k = \overline{1, d_j}.$$

Первый способ требует $\lfloor \log_2 d_j \rfloor + 1$ бинарных переменных, а второй и третий — d_j переменных.

Если дискретная переменная x_j принимает действительные значения $a_{j,1}, a_{j,2}, \ldots, a_{j,p_j}$, то её можно представить как

$$x_j = \sum_{k=1}^{p_j} a_{j,k} y_{j,k} \text{ при условиях } \sum_{k=1}^{p_j} y_{j,k} = 1,\ y_{jk} = 0,1,\ k = \overline{1, p_j}.$$

Непрерывную переменную $x_j, 0 \le x_j \le d_j$ (d_j — положительное действительное число, $d_j < +\infty$) можно заменить на дискретную переменную $y_j = \sum\limits_{k=-q_j}^{t_j} 2^k y_{j,k}$ при условиях $\sum\limits_{k=-q_j}^{t_j} 2^k y_{j,k} \le d_j$, $y_{j,k} = 0,1$, $k = -\overline{q_j, t_j}$. Выбирая достаточно большое положительное целое число q_j, добиваемся того, что любое значение непрерывной переменной x_j как угодно мало отличается от ближайшего к нему значения дискретной переменной y_j (рис. 2.1).

Логические условия. Рассмотрим множество X точек x из $S, S \subseteq R_n$, для каждой из которых выполняются, по крайней мере, q каких-нибудь неравенств из системы $g_k(x) \ge 0$, $k = \overline{1, r}$, где $1 \le q \le r - 1$.

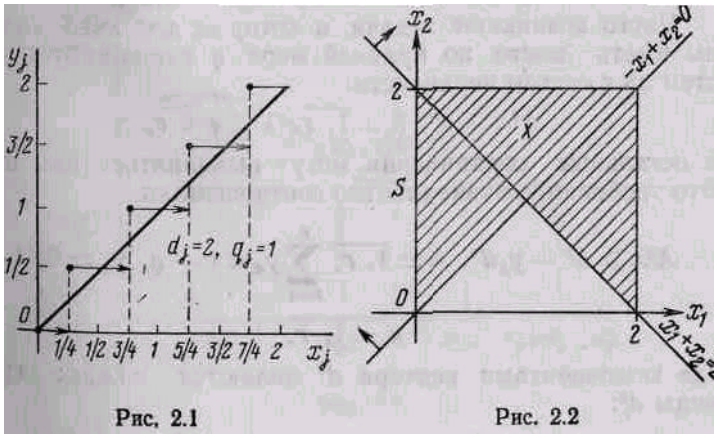


Рис. 2.1 Рис. 2.2

Можно задать множество X неравенствами $g_k(x) \ge \underline{g}_k y_k$, $k = \overline{1, r}$, $\sum\limits_{k=1}^{r} y_k \le r - q$ при выполнении условий $y_k = 0,1$, $\underline{g}_k \le g_k(x)$ для всех $x \in S, \underline{g}_k \ne -\infty$ ($k = \overline{1, r}$). Заметим, что условие $\underline{g}_k \ne -\infty$ является существенным. Результат не изменится, если заменить неравенство $\sum\limits_{k=1}^{r} y_k \le r - q$ на равенство $\sum\limits_{k=1}^{r} y_k = r - q$.

При $q = 1$ и $q = 2$ задача о многократных альтернативах превращается в задачу о дихотомии, т. е. получаем условие $g_1(x) \ge 0$ или $g_2(x) \ge 0$. Напомним, что союз "или" русского языка и логическая связка "или" математической логики (знак дизъюнкции $\bigvee$) употребляются здесь как неразделительный союз. Следовательно, условие $g_1(x) \ge 0$ или $g_2(x) \ge 0$ означает либо $g_1(x) \ge 0$, либо $g_2(x) \ge 0$, либо

$g_1(x) \geq 0$ и $g_2(x) \geq 0$. Легко проверить, что условие $g_1(x) \geq 0 \vee g_2(x) \geq 0$ эквивалентно соотношениям $g_1(x) \geq \underline{g}_1 y$, $g_2(x) \geq \underline{g}_2(1-y)$, $y = 0,1$ (рис. 2.2). Значит, для задания множества X в этом случае достаточно только одной бинарной переменной.

Известно, что высказывание $A \to B$ (если A, то B) эквивалентно высказыванию $\neg A \vee B$ (не A или B). Поэтому условие $g_1(x) < 0 \to g_2(x) \geq 0$ эквивалентно $g_1(x) \geq 0 \vee g_2(x) \geq 0$, что эквивалентно соотношениям $g_1(x) \geq \underline{g}_1 y$, $g_2(x) \geq \underline{g}_2(1-y)$, $y = 0,1$. Аналогично $g_1(x) > 0 \to g_2(x) \geq 0$ эквивалентно $g_1(x) \leq 0 \vee g_2(x) \geq 0$, что эквивалентно $g_1(x) \leq \overline{g}_1 y$, $g_2(x) \geq \underline{g}_2(1-y)$, $y = 0,1$.

Часто возникают задачи, в которых для $x \in S$ должны иметь место, по крайней мере, q каких-нибудь систем из r систем неравенств

$$A^k x \geq b^k, \quad k = \overline{1,r}, \quad 1 \leq q \leq r,$$

а остальные ограничения могут выполняться или нет. Это требование эквивалентно соотношениям

$$A^k x \geq b^k + y_k d,\ k = \overline{1,r},\ \sum_{k=1}^{r} y_k \leq r - q,\ y_k = 0,1,\ k = \overline{1,r},$$

где компонентами вектора d^k являются нижние границы d_i^k:

$$d_i^k \leq \sum_{j=1}^{n} a_{i,j}^k x_j - b_i^k \text{ для всех } x \in S,$$

$$d_i^k \neq -\infty\ (i = \overline{1,m_k},\ k = \overline{1,r}).$$

Кусочно-линейная аппроксимация нелинейной функции. Рассмотрим задачу об аппроксимации нелинейной функции $f_j(x_j)$, заданной на интервале $0 \leq x_j \leq d_j$ (d_j — положительное действительное число, $d_j < +\infty$). Кривая заменяется ломаной, состоящей из конечного числа отрезков (например, рис. 2.3, 2.4). Такое приближение называется кусочно-линейной аппроксимацией функции. Оставляя в стороне вопрос о построении самой ломаной, ниже рассмотрим три способа представления кусочно-линейной функции, задающей эту ломаную. Изложенные здесь способы линеаризации можно применить, например, для кусочно-линейной аппроксимации нелинейной сепарабельной функции

$$f(x_1, x_2, \ldots, x_n) = \sum_{j=1}^{n} f_j(x_j).$$

На интервале $[0, d_j]$ выбираем точки $0 = a_{j,0} < a_{j,1} < a_{j,2} < \ldots < a_{j,p_j-1} < a_{j,p_j} = d_j$ (рис. 2.3). Функция $l_j(x_j)$, задающая ломаную, может быть представлена следующим образом. Пусть

$$x_j = \sum_{k=0}^{p_j} a_{j,k}\lambda_{j,k},\ \sum_{k=0}^{p_j} \lambda_{j,k} = 1,\ \lambda_{j,k} \geq 0,\ k = \overline{0,p_j},$$

$$l_j(x_j) = \sum_{k=0}^{p_j} f_j(a_{j,k})\lambda_{j,k}.$$

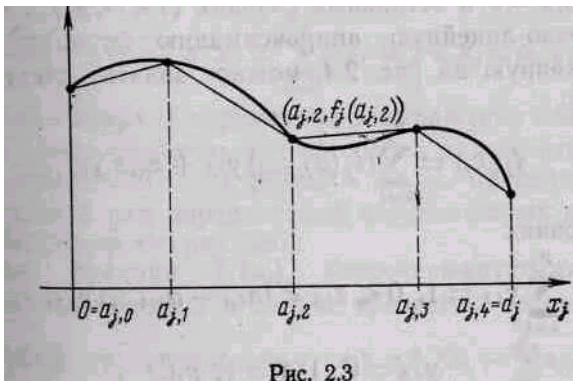


Рис. 2.3

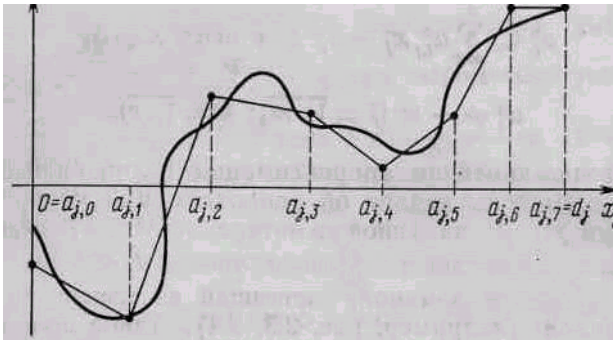


Рис. 2.4

Для фиксированного значения x'_j точка $(x'_j, l_j(x'_j))$ лежит на ломаной, если положительные не более двух $\lambda_{j,k}$, вида $\lambda_{j,t-1}, \lambda_{j,t}$.

Это условие может быть достигнуто при помощи ограничений

$$\lambda_{j,0} \le y_{j,1},\ \lambda_{j,k} \le y_{j,k} + y_{j,k+1},\ k = \overline{1, p_j - 1},$$

$$\lambda_{j,p_j} \le y_{j,p_j},\ \sum_{k=1}^{p_j} y_{j,k} = 1,\ y_{j,k} = 0, 1,\ k = \overline{1, p_j}.$$

Действительно, из этих ограничений следует, что $y_{j,t} = 1$ влечёт $y_{j,k} = 0, k \ne t$, а, следовательно, $\lambda_{j,t-1} \le 1, \lambda_{j,t} \le 1$ и $\lambda_{j,k} = 0$ в остальных случаях $(1 \le t \le p_j)$.

Кусочно-линейную аппроксимацию функции $f_j(x_j)$, изображённую на рис. 2.4, можно задать следующим образом:

$$l_j(x_j) = \sum_{k=1}^{p_j} (l_j(a_{j,k-1}) y_{j,k} + \alpha_{j,k} z_{j,k})$$

при условиях

$$\sum_{k=1}^{p_j} y_{j,k} = 1,\ 0 \le z_{j,k} \le (a_{j,k} - a_{j,k-1}) y_{j,k},\ y_{j,k} = 0, 1 \quad (k = \overline{1, p_j}).$$

Здесь $\alpha_{j,k}$ — тангенс угла наклона на интервале $[a_{j,k-1} - a_{j,k}]$, а $y_{j,k} = 1$ означает, что x_j, из интервала $[a_{j,k-1} - a_{j,k}]$ $(k = \overline{1, p_j})$.

Если функция $l_j(x_j)$ не является непрерывной, то её можно представить таким же образом с небольшими изменениями. Для этого нужно выбрать точки разбиения $a_{j,k}$ так, чтобы функция $l_j(x_j)$ была непрерывна в полуоткрытом интервале $[a_{j,k-1} - a_{j,k}]$. При этом переменные $z_{j,k}$ должны удовлетворять соотношениям $0 \le z_{j,k} \le (a_{j,k} - a_{j,k-1} - \varepsilon_{j,k}) y_{j,k}$, где $\varepsilon_{j,k}$ — достаточно малое положительное число.

Второй способ позволяет представить функцию $l_j(x_j)$, определённую на несмежных интервалах или в точках $a_{j,k}$.

Для непрерывной функции $l_j(x_j)$ можно указать еще один способ представления:

$$l_j(x_j) = l_j(0) + \sum_{k=1}^{p_j} \alpha_{j,k} z_{j,k}$$

при выполнении условий

$$(a_{j,k} - a_{j,k-1}) y_{j,k} \le z_{j,k} \le (a_{j,k} - a_{j,k-1}) y_{j,k-1},\ y_{j,k} = 0, 1,$$

$$y_{j,0} = 1\ (k = \overline{1, p_j}).$$

Из этих условий следует, что $1 \ge y_{j,1} \ge y_{j,2} \ge \ldots \ge y_{j,p_j}$.

Рассмотрим такое q, что $y_{j,q} = 1$, а $y_{j,q+1} = 0$ $(0 \le q \le p_{j-1})$. Тогда

$$y_{j,k} = 1,\ k = \overline{0, q}, y_{j,k} = 0,\ k = \overline{q+1, p_j},\ z_{j,k} = a_{j,k} - a_{j,k-1},$$

$$k = \overline{1,q},\ 0 \le z_{j,q+1} \le a_{j,q+1} - a_{j,q},\ z_{j,k} = 0,$$

$$k = \overline{q+2, p_j},\ \text{а}\ x_j = \sum_{k=1}^{p_j} z_{j,k},\ a_{j,q} \le x_j \le a_{j,q+1}.$$

Введя бинарные переменные в выражение для $l_j(x_j)$, можно слегка изменить третий способ и получить линейное представление для функции $l_j(x_j)$, не являющейся непрерывной или определённой на несмежных интервалах (например, см. рис. 2.6).

Если функция $l_j(x_j)$, аппроксимирующая $f_j(x_j)$ является выпуклой и в задаче требуется минимизировать $f_j(x_j)$, то выражение $l_j(x_j) = l_j(0) + \sum\limits_{k=1}^{p_j} \alpha_{j,k} z_{j,k}$ при выполнении условий $0 \le z_{j,k} \le a_{j,k} - a_{j,k-1}, k = \overline{1,p_j}$, даёт линейное представление функции $f_j(x_j)$ (в условиях уже не используются бинарные переменные). Это следует из условия $a_{j,1} < a_{j,2} < \ldots < a_{j,p_j}$.

Линейное представление полинома от бинарных переменных. Предположим, что степень полинома $p_i(x_1, x_2, \ldots, x_n)$ больше 1. Заменяем все степени x_j^k, $k \ge 2$, на x_j, так как $x_j = 0, 1$. Опустив коэффициент, рассмотрим произвольный член $\prod\limits_{j \in Q} x_j$ полинома $p_i(x_1, x_2, \ldots, x_n)$, где $Q = \{j_1, j_2, \ldots, j_{|Q|}\}$ — множество индексов бинарных переменных, входящих в этот член ($|Q| \ge 2$). Рассматриваемый член равен 1, если $x_j = 1$ для всех $j \in Q$, и равен 0, если это не так.

Введем бинарную переменную z_Q и два неравенства

$$\sum_{j \in Q} x_j - z_Q \le |Q| - 1, \quad -\sum_{j \in Q} x_j + |Q| \cdot z_Q \le 0.$$

Легко проверить соотношения

$$\prod_{j \in Q} x_j = 0 \longleftrightarrow z_Q = 0, \quad \prod_{j \in Q} x_j = 1 \longleftrightarrow z_Q = 1.$$

Заменяя каждое нелинейное слагаемое $a_Q \prod\limits_{j \in Q} x_j$ на $a_Q z_Q$, получаем линейное выражение относительно переменных x_j и z_Q, представляющее полином $p_i(x_1, x_2, \ldots, x_n)$ при выполнении неравенств указанного вида.

Существуют различные приемы для уменьшения числа переменных, появляющихся в процессе линеаризации. Они особенно важны для практических задач.

Бинарное представление подмножеств заданного множества. Пусть $S = \{s_1, s_2, \ldots, s_m\}$ — заданное конечное множество элементов произвольной природы, а T — его подмножество, $T \subseteq S$. Множеству T поставим в соответствие бинарный вектор $a = (a_1, a_2, \ldots, a_m)$, где

$$a_i = \begin{cases} 0, & s_i \notin T, \\ 1, & s_i \in T. \end{cases} \quad i = \overline{1,m}.$$

Такое представление множества позволяет отвлечься от природы его элементов и заменить операции надмножествами операциями над бинарными векторами. Этот прием используется, например, при бинарных формулировках задач о покрытии, разбиении и упаковке множества (см. § 2.4).

Роли бинарных переменных в целочисленной оптимизации посвящены, например [69, 77].

2.2 ВЫБОР ОПТИМАЛЬНОГО ВАРИАНТА

Задача о ранце. Пусть имеется достаточно много предметов n видов. Для предмета j-го вида известны его масса a_j и стоимость c_j, $a_j > 0$, $c_j > 0$ ($j = \overline{1,n}$). Требуется указать такой набор предметов, масса которых не превышает заданной массы b, а их стоимость максимальна. Эта задача интерпретируется как задача о загрузке походного ранца. Поэтому ниже сформулированные задачи оптимизации также называются задачами о ранце.

Введем целочисленные переменные x_j, значение которых — количество предметов j-го вида, помещённых в ранец ($j = \overline{1,n}$). Тогда можно сформулировать следующую ЦЛ-задачу.

Найти вектор x^*, который максимизирует

$$\sum_{j=1}^{n} c_j x_j$$

при условиях

$$\sum_{j=1}^{n} a_j x_j \le b,\ x_j \ge 0,\ j = \overline{1,n},\ x_j \equiv 0,\ j = \overline{1,n}.$$

Оптимальное решение этой задачи даёт наилучший вариант загрузки походного ранца.

Укажем несколько вариантов этой задачи. Если $x_j \le d_j < +\infty$, $j = \overline{1,n}$, то получаем задачу о ранце с ограниченными переменными. При $d_j = 1$ переменная x_j принимает значения 0 и 1 (предмет j-го вида либо не помещается в ранец, либо помещается). Если все переменные x_j, являются бинарными, то задача называется бинарной задачей о ранце. Многомерная задача о ранце формулируется как

$$\max\{cx \mid Ax \le b,\ 0 \le x \le d,\ x \equiv 0\},$$

где данные A, b, c, d — неотрицательные (т. е. все элементы $m \times n$-матрицы A неотрицательные), а компонента d_j вектора $d = (d_1, d_2, \ldots, d_n)$ либо является положительным целым числом, либо равна $+\infty$ ($j = \overline{1,n}$).

Несмотря на свою кажущуюся простоту, задача о ранце играет важную роль в целочисленной оптимизации (например [69, 77]).

Задача о выборе вариантов. Рассмотрим отрасль, в которой работают m предприятий. Пусть предприятие i имеет n_i взаимно исключающих друг друга вариантов развития ($i = \overline{1,m}$). Реализация варианта r на предприятии i требует затрат $c_{i,r}$ и характеризуется вектором $(a_{i,r,1}, a_{i,r,2}, \ldots, a_{i,r,p})$, компоненты которого определяют объёмы производства и потребления продуктов ($i = \overline{1,m};\ r = \overline{1,n_i}$). Нужно указать варианты развития, реализация которых требует наименьших затрат и даёт не меньше заданного объёма b_k по каждому продукту k ($k = \overline{1,p}$).

Введем бинарные переменные $x_{i,r}$: $x_{i,r} = 0 \longleftrightarrow$ вариант r не реализуется на предприятии i, $x_{i,r} = 1 \longleftrightarrow$ вариант r реализуется на предприятии i. Тогда можно сформулировать следующую БЛ-задачу.

Найти вектор x^*, который минимизирует

$$\sum_{i=1}^{m} \sum_{r=1}^{n_i} c_{i,r} x_{i,r}$$

при условиях

$$\sum_{i=1}^{m} \sum_{r=1}^{n_i} a_{i,r,k} x_{i,r} \ge b_k,\ k = \overline{1,p}, \quad \sum_{r=1}^{n_i} x_{i,r} = 1,\ i = \overline{1,m},$$

$$x_{i,r} = 0, 1,\ i = \overline{1,m},\ r = \overline{1,n_i}.$$

Нулевой вариант, характеризующийся нулевыми затратами и нулевым вектором, можно учесть либо введением бинарной переменной $x_{i,0}$, либо с помощью неравенства $\sum\limits_{r=1}^{n_i} x_{i,r} \le 1$.

2.3 НЕДЕЛИМЫЕ ВЕЛИЧИНЫ

Целочисленная распределительная задача. Эта задача возникает, например, при распределении самолетов n типов по m авиалиниям.

Предположим, что на определённый плановый период заданы следующие величины: a_i — число пассажиров, которых необходимо перевезти по i-й авиалинии; $a_{i,j}$ — число пассажиров, перевозимых одним самолетом j-го типа по i-й авиалинии; b_j — имеющееся число самолетов j-го типа; $c_{i,j}$ — стоимость эксплуатации одного самолета j-го типа на i-й авиалинии ($i = \overline{1,m},\ j = \overline{1,n}$). Требуется распределить имеющиеся самолеты по авиалиниям так, чтобы выполнить пассажирские перевозки с наименьшими затратами.

Введем целочисленные переменные $x_{i,j}$, значение которых — число самолетов j-го типа, направляемых на i-ю авиалинию ($i = \overline{1,m}, j = \overline{1,n}$).Тогда можно сформулировать следующую ЦЛ-задачу.

Найти вектор x^*, который минимизирует

$$\sum_{i=1}^{m}\sum_{j=1}^{n} c_{i,j}x_{i,j}$$

при условиях

$$\sum_{i=1}^{m} x_{i,j} = b_j,\ j = \overline{1,n},$$

$$\sum_{j=1}^{n} a_{i,j}x_{i,j} \geq a_i,\ i = \overline{1,m},\ x_{i,j} \geq 0,\ i = \overline{1,m},\ j = \overline{1,n},$$

$$x_{i,j} \equiv 0,\ i = \overline{1,m},\ j = \overline{1,n}.$$

Если из условий этой задачи исключим требование целочисленности переменных, то получим её непрерывный аналог.

Задача о выборе машин и распределении операций. Пусть имеется m различных типов машин, на каждой из которых могут выполняться n различных видов операций. На определённый плановый период задаются величины: a_i — фонд времени одной машины i-го типа; b_j — число операций j-го вида, которые нужно выполнить; $c_{i,j}$ — стоимость выполнения одной операции j-го вида на одной машине i-го типа; d_i — стоимость одной машины i-го типа; $t_{i,j}$ — время выполнения одной операции j-го вида на одной машине i-го типа ($i = \overline{1,m}, j = \overline{1,n}$). Требуется выбрать определённое число машин и осуществить распределение операций по выбранным машинам так, чтобы суммарная стоимость оказалась наименьшей.

Пусть y_t — число машин i-го типа, $x_{i,j}$ — число операций j-го вида, выполняемых на одной машине i-го типа в течение планового периода ($i = \overline{l,m}; j = \overline{l,n}$). Тогда можно сформулировать ЦЛ-задачу.

Найти векторы x^*, y^*, которые минимизируют

$$\sum_{i=1}^{m}\sum_{j=1}^{n} c_{i,j}x_{i,j} + \sum_{i=1}^{m} d_i y_i$$

при условиях

$$\sum_{i=1}^{m} x_{i,j} = b_j,\ j = \overline{1,n}, \quad \sum_{j=1}^{n} t_{i,j}x_{i,j} \leq a_i y_i,\ i = \overline{1,m},$$

$$x_{i,j} \geq 0, \quad i = \overline{1,m}, \quad j = \overline{1,n}, \quad y_i \geq 0, \quad i = \overline{1,m}, \quad x_{i,j} \equiv 0,$$
$$i = \overline{1,m}, \quad j = \overline{1,n}, \quad y_i \equiv 0, \quad i = \overline{1,m}$$

Задачи с неделимыми величинами рассматриваются. например, в [12].

2.4 ЗАДАЧИ О ПОКРЫТИИ, РАЗБИЕНИИ И УПАКОВКЕ МНОЖЕСТВА

Постановка задач. Рассмотрим множество $M = \{1, 2, \ldots, m\}$ и семейство P его подмножеств P_j, $P_j \subseteq M$ для всех $j \in N = \{1, 2, \ldots, n\}$. Подмножество J, $J \subseteq N$, определяет покрытие множества M, если $\bigcup\limits_{j\in J} P_j = M$. Если дополнительно $P_k \bigcap P_l = \varnothing$, $k \not\equiv l$ и k, l из J, то подмножество J определяет разбиение множества M. Подмножество J, $J \subseteq N$, определяет упаковку множества M, если каждый элемент из множества M либо отсутствует во всех подмножествах P_j, $j \in J$, либо входит только в одно из этих подмножеств. Покрытие, разбиение и упаковка множества представляют собой подсемейства заданного семейства подмножеств этого множества. Для удобства часто будем называть покрытием, разбиением или упаковкой само подмножество индексов, определяющее подсемейство.

Пусть с каждым подмножеством P_j связана стоимость (вес) $c_j, c_j \geq 0$ $(j \in N)$. Вес (стоимость) покрытия J есть $\sum\limits_{j \in J} c_j$. Задача оптимизации о взвешенном покрытии множества состоит в нахождении покрытия с минимальным весом. Теперь приведем бинарную формулировку этой задачи.

Найти вектор x^*, который минимизирует

$$\sum_{j=1}^{n} c_j x_j$$

при условиях

$$\sum_{j=1}^{n} a_{i,j} x_j \geq 1,\ i = \overline{1,m},\ x_j = 0,1,\ j = \overline{1,n}$$

Здесь

$$a_{i,j} = \begin{cases} 0, i \notin P_j, \\ 1, i \in P_j. \end{cases} \quad i = \overline{1,m}; j = \overline{1,n}.$$

Таким образом, столбец a_j матрицы A задаёт бинарное представление подмножества P_j, множества M $(j = \overline{1,n})$.

Связь между комбинаторной и бинарной задачами оптимизации устанавливается взаимно однозначным соответствием $J \longleftrightarrow x$, задаваемым соотношениями $x_j = 0 \longleftrightarrow j \notin J$, $x_j = 1 \longleftrightarrow j \in J$ $(j \in N)$. При $c_j = 1$, $j = \overline{1,n}$, получаем покрытие минимальной мощности.

Аналогично ставится задача оптимизации о взвешенном разбиении множества. Её бинарная формулировка отличается лишь условиями

$$\sum_{j=1}^{n} a_{i,j} x_j = 1,\ i = \overline{1,m}$$

При $c_j = 1$, $j = \overline{1,n}$, получаем разбиение минимальной мощности.

Задачу оптимизации о взвешенной упаковке множества удобно сформулировать в следующем виде. Найти вектор x^*, который максимизирует

$$\sum_{j=1}^{n} c_j x_j$$

при условиях

$$\sum_{j=1}^{n} a_{i,j} x_j \leq 1,\ i = \overline{1,m},\ x_j = 0,1,\ j = \overline{1,n}.$$

При $c_j = 1$, $j = \overline{1,n}$, получаем упаковку максимальной мощности.

Приведем обобщение рассмотренных задач. Бинарные формулировки задач о β-покрытии, β-разбиении и β-упаковке множества имеют соответственно вид:

$$\min\{cx \mid Ax \geq \beta,\ x \in B_n\}, \quad \min\{cx \mid Ax = \beta,\ x \in B_n\},$$

$$\max\{cx \mid Ax \leq \beta,\ x \in B_n\}.$$

Здесь β — m-вектор с положительными целочисленными компонентами. Раньше все компоненты вектора β были равны 1. Условие $\beta_j \geq 1$, например, в задаче о β-покрытии означает, что число подмножеств из покрытия J, содержащих элемент i из M, не меньше β_j.

Некоторые приложения. При рассмотрении прикладных задач либо используют бинарное представление множеств, либо сохраняют за элементом множества его конкретное содержание (последнее бывает иногда удобнее).

1. Задача о районировании. Пусть на некоторой географической территории находятся m населённых пунктов. Население пункта i составляет q_i человек $(i = \overline{1,m})$. Требуется разбить эту территорию на k непересекающихся районов так, чтобы эти районы были по возможности близкими по количеству населения.

Населённый пункт считается неделимым, т. е. его нельзя разбить на несколько частей и отнести эти части к различным районам. При формировании возможного района j могут учитываться такие факторы, как занимаемая площадь, смежность, компактность, экономические связи и др. Район j характеризуется вектором $(a_{1,j}, a_{2,j}, \ldots, a_{m,j})$, где $a_{i,j} = 1$, если населённый пункт i включается в район j, и $a_{i,j} = 0$, если нет $(i = \overline{1,m}, j = \overline{1,n})$. Идеальный случай, когда количество населения каждого района равно $\overline{q} = \frac{1}{k}\sum\limits_{i=1}^{m} q_i$, вообще говоря, недостижим из-за неделимости населённых пунктов. Мерой неравномерности для района j может служить величина $c_j = |p_j - \overline{q}|$, где p_j — количество населения этого района.

Введем бинарные переменные x_j:

$x_j = 0 \longleftrightarrow$ район j не выбирается;

$x_j = 1 \longleftrightarrow$ район j выбирается $(j = \overline{1,n})$.

Тогда можно сформулировать следующую бинарную задачу, нелинейную из-за целевой функции.

Найти вектор x^*, который минимизирует

$$\max_{j=1}^{n} c_j x_j$$

при условиях

$$\sum_{j=1}^{n} a_{i,j} x_j = 1,\ i = \overline{1,m}, \quad \sum_{j=1}^{n} x_j = k, x_j = 0,1,\ j = \overline{1,n}.$$

Эта задача может быть легко преобразована в ЦЛ-задачу (см. упражнение 2.1,б) и д)).

2. Задача о нахождении информации. Предположим, что получена заявка на m единиц информации, расположенной на n файлах. Файл с номером j имеет длину c_j, а единица информации с номером i находится на одном файле или дублируется на нескольких файлах, указанных при помощи $a_{i,j} = 1$ $(i = \overline{1,m}; j = \overline{1,n})$. Оптимальное покрытие даёт подмножество файлов, для которого минимальна суммарная длина файлов, просматриваемых для выполнения заявки.

3. Задача о разрыве путей в графе. Рассмотрим неориентированный граф $G = (V, E)$. Пусть I — некоторое множество путей в этом графе, а $N = \{1, 2, \ldots, |E|\}$. Равенство $a_{i,j} = 1$ означает, что ребро j входит в путь i, а c_j — стоимость удаления ребра j из графа G. Тогда оптимальное покрытие даёт минимальное по стоимости множество рёбер, которое разрывает все пути из I.

4. Задача о доставке. Рассмотрим организацию снабжения m пунктов товарами. Можно осуществлять выбор из n допустимых маршрутов. В этой задаче $a_{i,j} = 1$ означает, что пункт i находится на маршруте j $(i = \overline{1,m};\ j = \overline{1,n})$. Маршруту j приписана стоимость c_j (возможно, его длина). Тогда оптимальное разбиение даёт такой минимальный по стоимости набор маршрутов, что каждый пункт находится только на одном маршруте из этого набора.

5. Задача о расписании авиационных экипажей. Сеть воздушных линий состоит из m беспосадочных рейсов. Для обслуживания каждого такого рейса требуется один экипаж. Подмножество P_j находится в P, если оно представляет собой множество беспосадочных рейсов, которое по техническим условиям может быть обслужено одним экипажем, а c_j — стоимость такого обслуживания. В зависимости от того, разрешается ли членам экипажа быть пассажирами на некоторых рейсах или нет, минимальное покрытие или минимальное разбиение даёт оптимальное расписание.

6. Задача о раскраске. Требуется раскрасить заданную карту так, чтобы никакие две смежные области не имели одного и того же цвета. Пусть I — множество областей. Подмножество P_j находится в P, если никакие две области из P_j не имеют общей границы. Если все стоимости равны 1, то оптимальное разбиение указывает минимальное число требующихся цветов.

7. Задача о взвешенной рёберной упаковке. Рассмотрим неориентированный граф $G = (V, E)$, где V — множество вершин, а E — множество рёбер. Подмножество попарно несмежных рёбер из E называется паросочетанием графа G и представляет собой упаковку множества вершин. Пусть A — вершинно-рёберная матрица инциденций графа G, w_j — вес ребра $e_j, e_j \in E$. Тогда задача нахождения паросочетания в G, вес которого максимален, представляет собой задачу оптимизации о взвешенной упаковке и может быть сформулирована как бинарная задача

$$\max\{wx \mid Ax \le e,\ x \in B_{|E|}\}.$$

Здесь вектор $e = (1, 1, \ldots, 1)$ имеет $|V|$ компонент.

8. Задача о взвешенной вершинной упаковке. Рассмотрим неориентированный граф $G = (V, E)$, вершинам v_i которого приписаны веса w_i ($i = \overline{1, |V|}$). Задача о взвешенной вершинной упаковке состоит в нахождении независимого подмножества вершин из V (т. е. подмножества попарно несмежных вершин), вес которого максимален. Пусть A — та же самая вершинно-рёберная матрица инциденций. Тогда рассматриваемая задача может быть сформулирована как бинарная задача

$$\max\{yw \mid yA \le e,\ x \in B_{|V|}\}.$$

Здесь вектор $e = (1, 1, \ldots, 1)$ имеет $|E|$ компонент.

Существует много других приложений задач о покрытии, разбиении и упаковке множества. К ним относятся, например, задачи о проектировании переключательных схем и задачи о балансировании сборочной линии. Задачи о покрытии, разбиении и упаковке множества рассматриваются, например, в [66, 69].

2.5 ДВЕ ЗАДАЧИ О НАЗНАЧЕНИЯХ

Задача о назначениях. Предположим, что имеются m работ и m исполнителей. Каждый исполнитель может быть назначен на любую из этих работ. При этом различные исполнители получают различные работы. При назначении исполнителя j на работу i значение полезности равно $c_{i,j}$ ($i = \overline{1, m}$; $j = \overline{1, m}$). Требуется осуществить назначение m исполнителей на m работ так, чтобы суммарная полезность оказалась наибольшей.

Пусть $x_{i,j} = 1$ означает, что исполнитель j назначается на работу i, а $x_{i,j} = 0$ — нет ($i = \overline{1, m}$; $j = \overline{1, m}$). Тогда можно сформулировать БЛ-задачу.

Найти вектор x^*, который максимизирует

$$\sum_{i=1}^{m}\sum_{j=1}^{m} c_{i,j}x_{i,j}$$

при условиях

$$\sum_{j=1}^{m} x_{i,j} = 1,\ i = \overline{1, m}, \quad \sum_{i=1}^{m} x_{i,j} = 1,\ j = \overline{1, m},\ x_{i,j} = 0, 1,\ i = \overline{1, m};\ j = \overline{1, m}.$$

Квадратичная задача о назначениях. Квадратичная задача о назначениях возникает, например, при размещении предприятий. Предположим, что имеются m пунктов, в которых могут быть построены n предприятий ($n \le m$). В каждом пункте может находиться не более чем одно предприятие. Каждое предприятие должно быть размещено только в одном пункте. Заданы величины: $c_{i,j}$ — стоимость перевозки единицы продукта из пункта i в пункт j ($i = \overline{1, m}$; $j = \overline{1, m}$); $d_{k,l}$ — количество продукта, которое требуется доставить с предприятия k на предприятие l ($k = \overline{1, n}$, $l = \overline{1, n}$). Если предприятие k размещено в пункте i, а предприятие l — в пункте j, то стоимость взаимных поставок есть $c_{i,j}d_{k,l} + c_{j,i}d_{l,k}$. Требуется осуществить назначение предприятий, для которого суммарная стоимость поставок является наименьшей.

Пусть $x_{i,k} = 1$ означает, что предприятие k размещено в пункте i, а $x_{i,k} = 0$ — нет ($i = \overline{1, m}$, $k = \overline{1, n}$). Тогда можно сформулировать следующую бинарную квадратичную задачу.

Найти вектор x^*, который минимизирует

$$\sum_{1 \le i \ne j \le m}\ \sum_{1 \le k \ne l \le n} c_{i,j}d_{k,l}x_{i,k}x_{j,l}$$

при условиях

$$\sum_{k=1}^{n} x_{i,k} \le 1,\ i = \overline{1, m}, \quad \sum_{i=1}^{m} x_{i,k} = 1,\ k = \overline{1, n},\ x_{i,k} = 0, 1,\ i = \overline{1, m},\ k = \overline{1, n}.$$

Задача о назначениях рассматривается, например, в [80].

2.6 ЗАДАЧА О РАСПИСАНИИ РАБОТ НА МАШИНАХ

Пусть имеется m работ, которые могут выполняться на n машинах. Для простоты предположим, что каждая работа состоит из n операций, выполняемых на различных машинах (см. упражнение 2.13). Операции одной работы должны выполняться в определённом порядке. Этот порядок задаётся величинами $r_{i,j,k}$, где $r_{i,j,k} = 1$, если операция j работы i выполняется на машине k, и $r_{i,j,k} = 0$, если нет ($i = \overline{1,m}$; $j = \overline{1,n}$; $k = \overline{1,n}$). Пусть $t_{i,k}$ — время выполнения работы i на машине $k, t_{i,k} > 0$. Введем переменные $x_{i,k}, x_{i,k} \geq 0$, значения которых выражают моменты начала выполнения работы i на машине k. Предполагается, что выполнение работ начинается в момент 0.

Чтобы выразить требование, что никакие две работы не выполняются на одной и той же машине в одно и то же время, используем логическую связку "или". Это требование достигается введением условий $x_{s,k} - x_{r,k} \geq t_{r,k}$ или $x_{r,k} - x_{s,k} \geq t_{s,k}$ для всех работ r, s и для всех машин k. Ясно, что совместное выполнение этих двух неравенств невозможно. Как известно, такое логическое условие "или" можно выразить через бинарную переменную (см. § 2.1). Пусть $y_{r,s,k} = 0$, если работа r предшествует (не обязательно непосредственно) работе s на машине k, и $y_{r,s,k} = 1$, если наоборот. Тогда логическое условие "или" может быть заменено соотношениями $x_{s,k} - x_{r,k} \geq t_{r,k} - dy_{r,s,k}, x_{r,k} - x_{s,k} \geq t_{s,k} - d(1 - y_{r,s,k})$, $y_{r,s,k} = 0, 1$, где d — достаточно большое положительное число. Имеются nC_m^2 пар ограничений такого вида и nC_m^2 бинарных переменных (nC_m^2 — число сочетаний из m элементов по 2).

Заметим, что момент начала выполнения операции j работы i есть $\sum\limits_{k=1}^{n} r_{i,j,k}x_{i,k}$. Сохранение заданного порядка выполнения операций обеспечивается ограничениями

$$\sum_{k=1}^{n} r_{i,j,k}(x_{i,k} + t_{i,k}) \leq \sum_{k=1}^{n} r_{i,j+1,k}x_{i,k}, \quad j = \overline{1,n-1}.$$

Имеется $m(n-1)$ ограничений такого вида.

В качестве целевой функции может выступать, например, переменная x_0, значение которой выражает момент окончания последней операции работы, выполняемой последней, т. е. время выполнения всех работ. Нужно ввести еще m ограничений вида

$$\sum_{k=1}^{n} r_{i,n,k}(x_{i,k} + t_{i,k}) \leq x_0 \quad (i = \overline{1,m}).$$

Таким образом, следует минимизировать x_0.

Задача минимизации при сформулированных условиях есть СБЛ-задача, имеющая $mn + 1$ непрерывных переменных, nC_m^2 бинарных переменных и $mn + 2nC_m^2$ ограничений. Ясно, что даже для малых значений m, n получается очень большая задача. Например, для $m = 10$, $m = 4$ задача имеет 41 непрерывную переменную, 180 бинарных переменных и 400 ограничений.

Если выбрать подходящую единицу времени (скажем, секунду, минуту, час и т. д.), то можно считать, что переменные $x_{i,k}$ принимают целочисленные значения.

Задача о расписании работ на машинах и многие другие задачи изучаются в теории расписаний. Теории расписаний посвящены, например, работы [31, 46, 47].

2.7 УЧЕТ ФИКСИРОВАННЫХ ДОПЛАТ

В некоторых задачах стоимость технологического способа (столбца матрицы ограничений) равна фиксированной доплате за применение этого способа вообще плюс слагаемое, пропорциональное интенсивности применения этого же способа. В этом случае стоимость является нелинейной функцией от интенсивности x_j вида

$$f_j(x_j) = \begin{cases} 0, & x_j = 0, \\ c_j x_j + d_j, & x_j > 0 \end{cases} \text{ (рис. 2.5).}$$

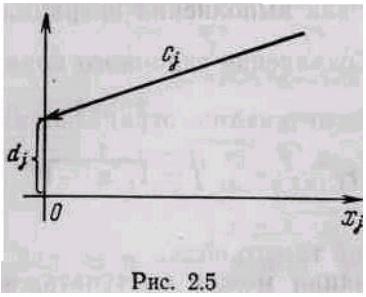


Рис. 2.5

Предположим, что $d_j > 0$ и функция $f_j(x_j)$ минимизируется. Тогда $f_j(x_j)$ может быть представлена как $g_j(x_j, y_j) = c_j x_j + d_j y_j$ при выполнении условий $x_j \geq 0$, $x_j < u_j y_j$, $y = 0, 1$, где u_j — известная конечная верхняя граница для значений переменной x_j.

Действительно, $x_j = 0 \to y_j = 0$ (в силу минимизации), $x_j > 0 \to y_j = 1$, $y_j = 0 \to x_j = 0$, $y_j = 1 \to x_j \leq u_j$(выполняется тривиально), а, следовательно, значения функций $f_j(x_j)$ и $g_j(x_j, y_j)$ совпадают для оптимальных решений.

Аналогичного результата можно достигнуть, если линейное неравенство $x_j \leq u_j y_j$ заменить на нелинейное ограничение $x_j(1 - y_j) = 0$ (теперь уже и не зная верхней границы u_j).

Переменную y_j, принимающую только два значения — 0 и 1, можно интерпретировать как "клапан", либо запрещающий использование технологического способа j ($y_j = 0$), либо позволяющий его использование ($y_j = 1$). Если $d_j > 0$ для всех j, то задача с фиксированными доплатами и линейными ограничениями может быть сформулирована как СБЛ-задача.

Производственная задача с фиксированными доплатами. Найти векторы x^*, y^*, которые минимизируют

$$\sum_{j=1}^{n}(c_j x_j + d_j y_j)$$

при условиях

$$\sum_{j=1}^{n} a_{i,j} x_j \leq b_j,\ i = \overline{1,m},\ x_j \leq u_j y_j,\ j = \overline{1,n},\ x_j \geq 0,$$

$$j = \overline{1,n},\ y_j = 0, 1,\ j = \overline{1,n}.$$

Эта задача возникает, например, при анализе производственной деятельности цеха, в котором могут работать n машин. Тогда c_j — стоимость единицы продукта, произведённого на машине j, d_j — стоимость установки (переналадки) машины j, x_j — количество продукта, производимого машиной j. Условия вида $\sum\limits_{j=1}^{n} a_{i,j} x_j \leq b_i$ выражают ограничения на производственное задание и имеющиеся ресурсы.

Если присутствуют ограничения на количество используемых машин, то их можно записать в виде $\sum\limits_{j=1}^{n} e_{i,j} y_j \leq h_i$ и присоединить к условиям задачи.

Транспортная задача с фиксированными доплатами. Если в предыдущей задаче условия $\sum\limits_{j=1}^{n} a_{i,j} x_j \leq b_i$ заменить условиями транспортной задачи, то получим транспортную задачу с фиксированными доплатами, часто встречающуюся в планировании перевозок.

Найти векторы x^*, y^*, которые минимизируют

$$\sum_{i=1}^{m}\sum_{j=1}^{n}(c_{i,j} x_{i,j} + d_{i,j} + y_{i,j})$$

при условиях

$$\sum_{j=1}^{n} x_{i,j} \leq a_i, \quad i = \overline{1,m},$$

$$\sum_{j=1}^{n} x_{i,j} \geq b_j, \quad j = \overline{1,n},$$

$$x_{i,j} \leq \min\{a_i, b_j\} \cdot y_{i,j}, \quad i = \overline{1,m}, \quad j = \overline{1,n},$$

$$x_{i,j} \geq 0, \quad i = \overline{1,m}, \quad j = \overline{1,n}, \quad y_{i,j} = 0,1, \quad i = \overline{1,m}, \quad j = \overline{1,n}.$$

Задача с фиксированными доплатами рассматривается, например, в [81].

2.8 ЗАДАЧИ О РАЗМЕЩЕНИИ ПРЕДПРИЯТИЙ

При размещении предприятий возникают задачи, в которых выбор вариантов производится с учётом как производственных, так и транспортных факторов. Эти задачи составляют класс производственно-транспортных задач с фиксированными доплатами.

Задача о размещении предприятий. Пусть имеется m возможных пунктов размещения предприятий. При размещении предприятия i в пункте A_i, требуются затраты d_i $(i = \overline{1,m})$. Предприятие i производит a_i единиц однородного продукта для n потребителей. Потребитель j из пункта B_i нуждается в b_i единицах этого продукта $(j = \overline{1,n})$. Известно, что $c_{i,j}$ — стоимость производства единицы продукта на предприятии i и доставки её потребителю j. Требуется выбрать пункты размещения таким образом, чтобы минимизировать суммарную стоимость удовлетворения потребностей.

Пусть $x_{i,j}$ — количество продукта, направляемого с предприятия i в пункте A_i потребителю j в пункте B_i, Значение $y_i = 1$ означает, что предприятие i строится в пункте A_i, а $y_i = 0$ — нет. Тогда можно сформулировать следующую СБЛ-задачу.

Найти векторы x^*, y^*, которые минимизируют

$$\sum_{i=1}^{m}\Big(\sum_{j=1}^{n} c_{i,j}x_{i,j} + d_i y_i\Big)$$

при условиях

$$\sum_{i=1}^{m} x_{i,j} = b_j,\ j = \overline{1,n}, \quad \sum_{j=1}^{n} x_{i,j} \leq a_i y_i,\ i = \overline{1,m},$$

$$x_{i,j} \geq 0,\ i = \overline{1,m},\ j = \overline{1,n},\ y_i = 0,1,\ i = \overline{1,m}.$$

Выбор значений $0,1$ для переменных y_i даёт транспортную задачу, которая совместна или нет. Таким образом, один из возможных способов решения этой задачи заключается в рассмотрении всех $2^m - 1$ возможных расположений предприятий, решении всех возникающих при этом транспортных задач и выборе наилучшего решения.

Простая задача о размещении. Если в рассматриваемой задаче считать величины a_i достаточно большими, то она эквивалентна простой задаче о размещении.

Найти векторы x^*, y^*, которые минимизируют

$$\sum_{i=1}^{m}\Big(\sum_{j=1}^{n} \overline{c}_{i,j}x_{i,j} + d_i y_i\Big)$$

при условиях

$$\sum_{i=1}^{m} x_{i,j} = 1,\ j = \overline{1,n},\ x_{i,j} \leq y_i,\ i = \overline{1,m},\ j = \overline{1,n},$$

$$x_{i,j} \geq 0, i = \overline{1,m},\ j = \overline{1,n},\ y_i = 0,1, i = \overline{1,m}.$$

Здесь $\overline{c}_{i,j} = c_{i,j}b_j$, а $x_{i,j}$ — часть заказа потребителя j, выполняемая предприятием i.

Задача об определении мощностей предприятий. Усложним задачу о размещении предприятий, предположив, что чем больше производится продукта, тем меньше стоимость единицы этого продукта. В рассматриваемой задаче наилучшим образом определяется объём производства для каждого предприятия. Будем считать, что стоимость продукта есть кусочно-линейная функция от

количества этого продукта (рис. 2.6). Более точно, если предприятие i выпустило больше чем $a_{i,k-1}$ единиц продукта, то берется доплата $d_{i,k}$, а стоимость единицы продукта равна $e_{i,k}$ до уровня $a_{i,k}$ $(0 = a_{i,0} < a_{i,1} < a_{i,2} < \ldots < a_{i,m_i-1} < a_{i,m_i} = a_i)$.

Пусть $z_{i,k}$ — количество продукта, производимого предприятием i со стоимостью $e_{i,k}$, а переменные $y_{i,k}$ принимают значения 0 и 1 $(i = \overline{1,m};\ k = \overline{1,m_i};\ y_{i,m_i+1} = 0)$. Тогда можно сформулировать следующую задачу.

Найти векторы x^*, y^*, z^*, которые минимизируют

$$\sum_{i=1}^{m}\Big(\sum_{j=1}^{n} c_{i,j}x_{i,j} + \sum_{k=1}^{m_i}(d_{i,k}y_{i,k} + e_{i,k}z_{i,k})\Big)$$

при условиях

$$\sum_{i=1}^{m} x_{i,j} = b_j,\ j = \overline{1,n},\ \sum_{j=1}^{n} x_{i,j} \le \sum_{k=1}^{m_j} z_{i,k},\ i = \overline{1,m},$$

$$(a_{i,k} - a_{i,k-1})y_{i,k+1} \le z_{i,k} \le (a_{i,k} - a_{i,k-1})y_{i,k},\ i = \overline{1,m},$$

$$k = \overline{1,m_i},\ x_{i,j} \ge 0,\ i = \overline{1,m},\ j = \overline{1,n},\ y_{i,k} = 0,1,$$

$$i = \overline{1,m},\ k = \overline{1,m_i},\ z_{i,k} \ge 0,\ i = \overline{1,m},\ k = \overline{1,m_i}.$$

Если $y_{i,k} = 1$ для $k \le q$ и $y_{i,k} = 0$ для $k > q$ (допустимый набор значений переменных $y_{i,k}$; $1 \le q \le m_i$), то предприятие i производит продукт в количестве от $a_{i,q-1}$ до $a_{i,q}$ со стоимостью $e_{i,q}$ за единицу. Если же для некоторого q имеет место $y_{i,q+1} = 0$, то из условий следует

$$z_{i,q+1} = z_{i,q+2} = \ldots = 0,\ y_{i,q+1} = y_{i,q+2} = \ldots = 0.$$

Можно сформулировать задачу об определении мощностей предприятий и для случая, когда стоимости задаются произвольными вогнутыми функциями.

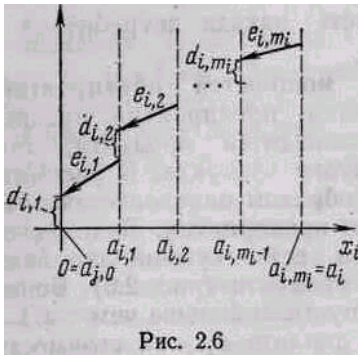


Рис. 2.6

Многопродуктовая задача о размещении предприятий. Является обобщением задачи о размещении предприятий, возникает, когда в пункте производства A_i есть несколько вариантов развития, по каждому варианту производится несколько продуктов.

Пусть в пункте производства A_i предприятие i в течение планового периода может развиваться только по одному из n_i возможных для него вариантов $(i = \overline{1,m})$. Вариант r для предприятия i характеризуется затратами $d_{i,r}$ и вектором $(a_{i,r,1}, a_{i,r,2}, \ldots, a_{i,r,p})$, компоненты которого определяют объёмы производства и потребления $(i = \overline{1,m};\ r = \overline{1,n_i})$. В течение планового периода потребность каждого пункта B_i, характеризуется вектором $(b_{j,1}, b_{j,2}, \ldots, b_{j,p})$ $(j = \overline{1,n})$. Известно, что $c_{i,j,k}$ — стоимость перевозки единицы продукта k из A_i в B_j $(i = \overline{1,m};\ j = \overline{1,n};\ k = \overline{1,p})$. Требуется выбрать варианты развития (значения переменных $y_{i,r}$) и объёмы перевозок (значения переменных $x_{i,j,k}$) таким образом, чтобы суммарная стоимость удовлетворения потребностей была наименьшей. В принятых обозначениях можно сформулировать следующую СБЛ-задачу.

Найти векторы x^*, y^*, которые минимизируют

$$\sum_{i=1}^{m}\Big(\sum_{j=1}^{n}\sum_{k=1}^{p} c_{i,j,k}x_{i,j,k} + \sum_{r=1}^{n_j} d_{i,r}y_{i,r}\Big)$$

при условиях

$$\sum_{i=1}^{m} x_{i,j,k} = b_{j,k},\ j = \overline{1,n},\ k = \overline{1,p},$$

$$\sum_{j=1}^{n} x_{i,j,k} \le \sum_{r=1}^{n_i} a_{i,r,k} y_{i,r},\ i = \overline{1,m},\ k = \overline{1,p},$$

$$\sum_{r=1}^{n_i} y_{i,r} = 1,\ i = \overline{1,m},$$

$$x_{i,j,k} \ge 0,\ i = \overline{1,m},\ j = \overline{1,n},\ k = \overline{1,p},$$

$$y_{i,r} = 0,1,\ i = \overline{1,m},\ r = \overline{1,n_i}.$$

Если заменить условие бинарности $y_{i,r} = 0,1$ на ограничение $y_{i,r} \ge 0$ для всех i, r, то получим непрерывный аналог этой задачи. Поскольку в непрерывном случае $0 \le y_{i,r} \le 1$, то значение переменной $y_{i,r}$ можно интерпретировать как часть планового периода, в течение которой предприятие i работает по варианту r ($i = \overline{1,m};\ r = \overline{1,n_i}$).

Многопродуктовая задача о размещении предприятий возникает, в частности, при размещении в нескольких пунктах нефтеперегонных заводов различных мощностей, производящих бензин, керосин и другие нефтепродукты с учётом потребностей конкретных потребителей.

Задачи о размещении предприятий рассматриваются, например, в [6, 21].

2.9 ЗАДАЧА О ПЕРЕВОЗКЕ ПРОДУКТА

Предположим, что в пунктах отправления (производства) i, $i \in V_1$, находятся по a_i; единиц некоторого однородного продукта, скажем, овощей или нефти, а в пунктах назначения (потребления) i, $i \in V_3$, существуют потребности в b_i; единицах этого продукта. Имеется транспортная сеть, связывающая пункты отправления и пункты назначения непосредственно или через промежуточные пункты i, $i \in V_2$. Известно, что $c_{i,j}$ — стоимость перевозки единицы продукта из пункта i в пункт j, $d_{i,j}$ — пропускная способность участка транспортной сети, соединяющего пункты i и j. Через эту транспортную сеть требуется осуществить перевозку продукта из пунктов отправления в пункты назначения с наименьшими затратами.

Для формулировки этой задачи удобно воспользоваться понятием ориентированного графа (см. упражнение 2.22), вложив в него конкретное содержание.

Пусть $E = \{(i,j)\}$ — множество участков транспортной сети (дуг), $V = V_1 \cup V_2 \cup V_3 = \{l, 2, \ldots, m\}$ — множество пунктов (вершин), $V(i) = \{j \mid (i,j) \in E\}, V'(i) = \{j \mid (i,j) \in E\}$, а $x_{i,j}$ — количество продукта, перевозимого из пункта i в пункт j. Тогда можно сформулировать следующую Л-задачу.

Задача о потоке минимальной стоимости. Найти вектор x^*, который минимизирует

$$\sum_{i \in V} \sum_{j \in V(i)} c_{i,j} x_{i,j}$$

при условиях

$$\sum_{j \in V(i)} x_{i,j} - \sum_{j \in V'(i)} x_{j,i} \begin{cases} \le a_i,\ i \in V_1, \\ = 0,\ i \in V_2, \\ \le -b_i,\ i \in V_3, \end{cases}$$

$$0 \le x_{i,j} \le d_{i,j},\ (i,j) \in E.$$

Допустимое решение $\{x_{i,j} \mid (i,j) \in E\}$ этой Л-задачи специальной структуры называется потоком. Если все величины $a_i, b_i, d_{i,j}$ целочисленные, то ЦЛ-задача, получающаяся из рассматриваемой задачи путём добавления ограничений целочисленности на все переменные, эквивалентна Л-задаче, так как каждое допустимое базисное (в частности, и оптимальное базисное) решение этой Л-задачи целочисленное.

Задачи о перевозке продукта рассматриваются, например, в [9, 15].

2.10 СЕТЕВОЙ ГРАФИК

Пусть задан ориентированный граф $G = (V, E)$ без петель и параллельных дуг. Каждой дуге (i, j) из E поставим в соответствие действительное число $c_{i,j}$, а каждой вершине i из V — действительное число b_i. При этом числа b_i удовлетворяют равенству $\sum\limits_{i=1}^{m} b_i$. Тогда сформулируем пару двойственных задач линейной оптимизации.

Прямая задача. Максимизировать

$$\sum_{i \in V} \sum_{j \in V(i)} c_{i,j} x_{i,j}$$

при условиях

$$\sum_{j \in V(i)} x_{i,j} - \sum_{j \in V'(i)} x_{j,i} = b_i,\ i \in V, x_{i,j} \geq 0,\ (i,j) \in E.$$

Двойственная задача. Минимизировать

$$\sum_{i \in V} b_i y_i$$

при условиях $y_i - y_j \geq c_{i,j}, (i,j) \in E$.

Если $c_{i,j} \leq 0, (i,j) \in E$, то вершины и дуги ориентированного графа G, величины $b_i, -c_{i,j}$ прямой задачи могут иметь ту же интерпретацию, что и в задаче о перевозке продукта, а саму прямую задачу часто называют транспортной задачей в сетевой постановке.

К частному случаю прямой и двойственной задач приводит анализ сетевого графика, что составляет основное содержание задач сетевого планирования и управления. Сетевой график, по существу являющийся сетью, используется для отражения взаимосвязей и последовательности выполнения комплекса работ. Укажем некоторые терминологические соответствия: сеть — сетевой график; вершина — событие; дуга (i, j) — работа $a_{i,j}$; начальная (конечная) вершина дуги — начальное (конечное) событие работы; число, поставленное в соответствие дуге, — продолжительность выполнения работы; источник — исходное событие; сток — завершающее событие и т. д.

Событие в сетевом графике есть факт выполнения одной или нескольких работ, входящих в это событие. К выполнению работы $a_{i,j}$, выходящей из события i, можно приступить лишь после выполнения всех работ, входящих в событие i. Ввиду этого правильно составленный сетевой график не может содержать контуров, поскольку ни одна работа из контура не может быть сделана из-за невыполнения предыдущей.

Путем в сетевом графике называется любая последовательность работ, соединяющая какие-либо два события. Если путь соединяет исходное и завершающее события, то он называется полным. Полный путь, имеющий наибольшую длину, называется критическим. При вычислении длины пути суммируются продолжительности выполнения работ, составляющих этот путь.

Предположим, что сетевой график имеет единственное исходное и единственное завершающее события (случай нескольких исходных или завершающих событий сводится к этому случаю). Будем считать, что выполнена правильная нумерация, т. е. все события сетевого графика занумерованы таким образом, что номер начального события любой работы меньше номера конечного события этой работы, исходное событие имеет номер 1 ($V'(1) = \varnothing$), а завершающее событие — номер m ($V(m) = \varnothing$) (рис. 2.7).

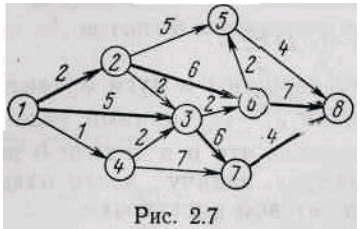


Рис. 2.7

Задача состоит в определении минимального срока t выполнения всего комплекса работ, а также в нахождении для каждого события i величины t_i, указывающей, насколько раньше завершающего события m должно выполняться событие i, чтобы весь комплекс работ был выполнен за время t.

Для решения этой задачи достаточно решить двойственную задачу, в которой $b_1 = l, b_2 = b_3 = \ldots = b_{m-1} = 0, b_m = -1$. Тогда $t = t_1 = y_1 - y_m, t_i = y_i - y_m, i = \overline{1,m}$. Ограничения двойственной задачи выражают, что $t_i \geq t_j + c_{i,j}$ для всех работ $a_{i,j}$, а целевая функция t_1 есть именно та величина, которую требуется минимизировать.

Для нахождения критического пути достаточно решить прямую задачу, в которой $b_1 = l, b_2 = b_3 = \ldots = b_{m-1} = 0, b_m = -1$. Длина критического пути даёт наименьшее время, в течение которого может быть выполнен весь комплекс работ.

Отметим, что по теореме двойственности оптимальные значения целевых функций прямой и двойственной задач равны.

Задачи сетевого планирования и управления рассматриваются, например, в [25].

2.11 ВЫБОР ОПТИМАЛЬНОГО МАРШРУТА

Задача о коммивояжёре. Пусть задано конечное множество пунктов, занумерованных числами $1, 2, \ldots, n$. Коммивояжёр желает с наименьшими затратами посетить все пункты по одному разу, начиная и заканчивая свое путешествие, например, в пункте 1. Такой объезд будем называть туром.

Формулировка задачи в терминах графа. Рассмотрим ориентированный граф $G = (V, E)$, где $V = \{1, 2, \ldots, n\}$ соответствует множеству пунктов, $(i, j) \in E$, если существует возможность прямого переезда из пункта i в пункт j. Тур t можно представить как последовательность вершин $t = (1, i_1, i_2, \ldots, i_{n-1}, 1)$, где $(i_1, i_2, \ldots, i_{n-1})$ — перестановка чисел $2, 3, \ldots, n$. Таким образом, существует не больше чем $(n-1)!$ туров. Тур также можно представить как последовательность дуг $t = ((1, i_1), (i_1, i_2), \ldots, (i_{n-1}, 1))$.

Пусть $c_{i,j}$ — стоимость переезда из пункта i в пункт j (длина дуги (i, j)); $E(t)$ — множество дуг тура t. Тогда стоимость тура t определяется как

$$z(t) = \sum_{(i,j) \in E(t)} c_{i,j}.$$

Предполагается, что граф G содержит, по крайней мере, один тур (гамильтонов контур). Если нет дуги, ведущей из вершины i в вершину j, то $c_{i,j} = d$, где d — достаточно большое положительное число. В частности, считаем $c_{i,i} = d$ $(i = \overline{1,n})$. Равенство $c_{i,j} = c_{j,i}$ может и не выполняться.

Задача о коммивояжёре может быть сформулирована как следующая комбинаторная задача.

Минимизировать

$$z(t) = \sum_{(i,j) \in E(t)} c_{i,j},$$

где t — тур.

Связь с задачей о назначениях. Будем считать, что $x_{i,j} = 0 \longleftrightarrow (i, j)$ не принадлежит $E(t)$. Из каждой вершины i выходит только одна дуга любого тура, что даёт $\sum\limits_{j=1}^{n} x_{i,j} = 1$ $(i = \overline{1,n})$. Аналогично в каждую вершину j входит только одна дуга (i, j) любого тура, что даёт $\sum\limits_{i=1}^{n} x_{i,j} = 1$ $(j = \overline{1,n})$. Стоимость тура есть $\sum\limits_{i=1}^{n} \sum\limits_{j=1}^{n} c_{i,j} x_{i,j}$. Таким образом, задача минимизации стоимости тура при сформулированных условиях представляет собой задачу о назначениях в бинарной формулировке.

К сожалению, задача о назначениях не эквивалентна задаче о коммивояжёре. Это иллюстрирует пример, изображённый на рис. 2.8, где представлено оптимальное решение задачи о назначениях, для которого $x_{1,5} = x_{5,7} = x_{7,2} = x_{2,1} = x_{3,6} = x_{6,4} = x_{4,3} = 1$ и $x_{i,j} = 0$ в остальных случаях. Однако это решение не является даже допустимым решением задачи о коммивояжёре, так как оно даёт два контура $(1, 5, 7, 2, 1)$ и $(3, 6, 4, 3)$, ни один из которых не гамильтонов. Такие контуры называют подтурами. Это означает, что к условиям задачи о назначениях нужно присоединить ограничения, исключающие появление подтуров.

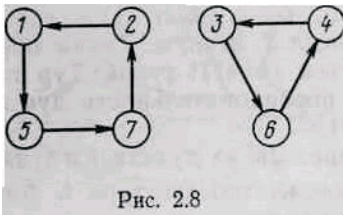


Рис. 2.8

Чтобы записать эти ограничения, заметим, что x' даёт подтур, если и только если $\sum\limits_{i\in Q}\sum\limits_{j\in\overline{Q}} x'_{i,j} = 0$ для некоторого непустого подмножества Q из V, где $\overline{Q} = V\backslash Q$. Таким образом, тур есть такое допустимое решение задачи о назначениях, которое удовлетворяет неравенствам $\sum\limits_{i\in Q}\sum\limits_{j\in\overline{Q}} x_{i,j} \geq 1$ для всех $Q \subset V$, $Q \neq \varnothing$. Итак, задача о коммивояжёре может быть сформулирована как следующая БЛ-задача.

Минимизировать

$$z = \sum_{i=1}^{n}\sum_{j=1}^{n} c_{i,j}x_{i,j}$$

при условиях

$$\sum_{j=1}^{n} x_{i,j} = 1,\ i = \overline{1,n},$$

$$\sum_{i=1}^{n} x_{i,j} = 1,\ j = \overline{1,n},$$

$$\sum_{i\in Q}\sum_{j\in\overline{Q}} x_{i,j} \geq 1\ \forall Q \subset V,\ Q \neq \varnothing,$$

$$x_{i,j} = 0,1,\ i = \overline{1,n},\ j = \overline{1,n}.$$

Существует $2^n - 2$ подмножеств Q, что даёт такое же число ограничений, исключающих подтуры. Поэтому непрактично решать эту задачу общим алгоритмом из-за большого числа ограничений. Задача о назначениях, которая является релаксацией задачи о коммивояжёре, даёт возможность построить алгоритм ветвей и границ для нахождения минимального тура.

Ниже приведем для задачи о коммивояжёре целочисленную формулировку, в которой исключение подтуров осуществляется более экономным образом (вместо 2^n-2 неравенств достаточно присоединить не более n^2 неравенств).

Задача о нескольких коммивояжёрах. Пусть имеется $n+1$ пунктов, занумерованных числами $0, 1, 2, \ldots, n$. Известно, что $c_{i,j}$ — стоимость переезда из пункта i в пункт j $(0 \leq i \neq j \leq n)$. Имеется несколько коммивояжёров. Каждый коммивояжёр по одному разу посещает не больше, чем p пунктов (не считая пункт 0), начиная и заканчивая свое путешествие в пункте 0. Такой объезд будем называть подтуром.

Требуется определить количество y коммивояжёров и маршрут для каждого коммивояжёра таким образом, чтобы суммарная стоимость всех подтуров оказалась наименьшей, а каждый пункт, исключая пункт 0, был посещён только один раз.

Если $y = 1$ и $p \geq n$, то эта задача превращается в задачу о коммивояжёре, рассмотренную выше.

Задача о нескольких коммивояжёрах может быть сформулирована как следующая ЦЛ-задача (см. упражнение 2.19).

Минимизировать

$$z = \sum_{0\leq i\neq j\leq n} c_{i,j}x_{i,j}$$

при условиях

$$\sum_{\substack{j=0\\ j\neq i}}^{n} x_{i,j} = 1,\ i = \overline{1,n},$$

$$\sum_{\substack{i=0\\ i\neq j}}^{n} x_{i,j} = 1,\ j = \overline{1,n},$$

$$\sum_{i=1}^{n} x_{i,0} = y,$$

$$z_i - z_j + p x_{i,j} \le p - 1,\ 1 \le i \ne j \le n,$$

$$\sum_{i=1}^{n} z_i \le np,$$

$$x_{i,j} \ge 0,\ 0 \le i \ne j \le n,\ y \ge 0,\ z_i \ge 0,\ i = \overline{1,n},$$

$$x_{i,j} \equiv 0,\ 0 \le i \ne j \le n,\ y \equiv 0,\ z_i \equiv 0, i = \overline{1,n}.$$

Сформулированная ЦЛ-задача имеет $n^2 + n + 2$ основных ограничений и $n^2 + n$ целочисленных переменных. Неравенства вида $z_i - z_j + p x_{i,j} \le p - 1$ служат для исключения недопустимых туров. Если число подтуров у не зафиксировано, то можно исключить равенство $\sum\limits_{i=1}^{n} x_{i,0} = y$. Как показывает небольшой вычислительный эксперимент, линейные ограничения рассматриваемой ЦЛ-задачи дают довольно хорошее приближение к выпуклой оболочке допустимых решений задачи о коммивояжёре.

Результаты счета (задача о коммивояжёре, алгоритм секущих плоскостей, система АЛЬФА-6) приведены в табл. 2.1, где используются обозначения: $n0$ — номер задачи, c_1 — число пунктов, c_2 — число основных ограничений, c_3 — число целочисленных переменных, c_4 — оптимальное значение целевой функции Л-задачи, c_5 — оптимальное значение целевой функции ЦЛ-задачи, c_6 — число симплекс-итераций для решения Л-задачи, c_7 — число симплекс-итераций для решения ЦЛ-задачи, c_8 число построенных сечений, c_9 — время решения ЦЛ-задачи в секундах на БЭСМ-6.

Задачи о выборе оптимального маршрута рассматриваются, например, в [41, 57].

Таблица 2.1

$n0$	c_1	c_2	c_3	c_4	c_5	c_6	c_7	c_8	c_9
1	3	8	4	16	16	5	5	0	0.26
2	4	14	9	6.3	7	6	8	1	0.7
3	5	22	16	14.5	15	18	22	1	4.4
4	4	14	9	29	29	13	13	0	1.2
5	5	22	16	30	32	27	30	1	5.7
6	6	32	25	14.4	18	24	46	2	16.5
7	6	32	25	56.2	63	35	80	4	32.4
8	10	92	81	140.8	146	84	155	3	376
9	10	92	81	24	–	56	100	3	258

УПРАЖНЕНИЯ

2.1. Доказать, что имеют место соотношения:

а) $\max\{x_1, x_2\} = \frac{x_1+x_2+|x_2-x_1|}{2}$;

б) $\max\limits_{j=1}^{n} a_j \le b_{\leftarrow} \to a_1 \le b,\ a_2 \le b, \ldots, a_n \le b$;

в) $\min\limits_{x\in X} f(x) = -\max\limits_{x\in X} -f(x)$;

г) $\max\limits_{x\in X} f(x) = -\min\limits_{x\in X} -f(x)$;

д) $\min\{f(x) \mid x \in X\} = \min = \{z \mid z \ge f(x),\ x \in X\}$;

е) $\max\{f(x) \mid x \in X\} = \max = \{z \mid z \le f(x),\ x \in X\}$.

Здесь предполагается существование соответствующих минимумов и максимумов.

2.2. Множество $X, X \subset R_n$, называется выпуклым, если из $x \in X$, $y \in X$ следует $(\lambda x + (1-\lambda)y) \in X$ для всех $\lambda \in [0, 1]$.

Показать, что множества $X_1 = \{x \mid Ax = b,\ x \geq 0\}$ и $X_2 = \{x \mid Ax \geq b,\ x \geq 0\}$ являются выпуклыми. Всегда ли множество вида $X = \{x \mid Ax = b,\ x \equiv 0\}$ не является выпуклым?

2.3. Сформулировать задачу $\max\{dy \mid A^1y \leq b^1,\ b^2 + A^2y \geq 0,\ b^2 + A^2y \equiv 0,\ y \geq 0\}$ как СЦЛ-задачу.

2.4. В пространстве $0x_1y_1y_2$ изобразить множество допустимых решений следующей СЦЛ-эадачи: максимизировать $x_1 - 3y_1 + 4y_2$ при условиях

$$-2x_1 - y_l + y_2 \leq -3,$$

$$-3x_l + y_1 - y_2 \leq 2,$$

$$x_1 \geq 0,\ x_1 \equiv 0,$$

$$y_1 \geq 0,\ y_2 \geq 0.$$

2.5. Записать числа $33, 63$ и 128 в двоичной, троичной и восьмеричной системах счисления.

2.6. Однозначная функция $f(x_1, x_2, \ldots, x_n)$ от n бинарных переменных, принимающая только значения 0 и 1 или одно из них называется двоичной функцией.

А. Показать, что существует 2^{2^n} различных двоичных функций.

Б. Перечислить все четыре двоичные функции от одной бинарной переменной.

В. Перечислить все шестнадцать двоичных функций от двух бинарных переменных, продолжив таблицу

x_1	x_2	f_0	f_1	f_2	f_3	$\ldots$
0	0	0	0	0	0	$\ldots$
0	1	0	0	0	0	$\ldots$
1	0	0	0	1	1	$\ldots$
1	1	0	1	0	1	$\ldots$

2.7. Используя бинарные переменные, записать логические условия и сформулировать соответствующие предпосылки:

а) $g_1(x) \geq 0$ или $g_2(x) \geq 0$, но не оба;

б) если $g_l(x) \geq 0$ или $g_2(x) \geq 0$, то $g_3(x) \geq 0$;

в) если $x > 0$, то $g(x) \leq a$ или $g(x) \geq b$.

2.8. Изменить формулировку задачи о загрузке ранца, дополнительно предположив, что предмет l помещается в ранец лишь в случае, когда уже выбран (не выбран) предмет k, где k и l заданы, а выбор осуществляется из n различных предметов.

2.9. Задача о садовнике. Садовнику на Кентукки необходимо закупить для сада 107 фунтов удобрений. Он может купить удобрения в мешках массой 35 фунтов по цене 14 дол. за мешок или в мешках массой 24 фунта по цене 12 дол. за мешок. Цель садовника - купить не менее 107 фунтов удобрений, затратив минимальную сумму денег.

Сформулировать соответствующую ЦЛ-задачу.

2.10. Задача о линейном раскрое. Вектор $a_j = (a_{1,j}, a_{2,j}, \ldots, a_{m,j})$ с неотрицательными целочисленными компонентами определяет способ раскроя полосы длинной l: из полосы вырезаются $a_{i,j}$ заготовок длиною l_i ($\sum_{i=1}^{m} a_{i,j}l_i \leq l$; $i = \overline{1,m}$; $j = \overline{1,n}$). Из достаточно большого числа одинаковых полос длиною l нужно вырезать не меньше b_i заготовок длиною l_i ($i = \overline{1,m}$). При этом требуется либо использовать наименьшее число полос, либо получить наименьший остаток материала.

Сформулировать две ЦЛ-задачи оптимизации. На практике, как правило, решают соответствующие им линейные задачи. В каких случаях этот подход оправдан?

2.11. Задачи о бесконфликтной и эффективной экспедициях.

А. Задача о бесконфликтной экспедиции. Группа исследователей намерена отправиться в экспедицию. При этом желательно, чтобы число участников экспедиции было максимальным. Однако некоторые из них не ладят друг с другом. Следовательно, если одно из лиц, входящих в конфликтную пару, отправляется в экспедицию, то другому лицу придется остаться дома.

Сформулировать задачу об упаковке множества и соответствующую ей бинарную задачу.

Б. Задача об эффективной экспедиции. Экспедиция, в которой не более n участников, должна выполнить m работ. Каждый вид работ выполняется хотя бы одним участником экспедиции. Цель состоит

в том, чтобы сформировать экспедицию с наименьшим числом участников так, чтобы выполнить все виды работ.

Сформулировать задачу о покрытии множества и соответствующую ей бинарную задачу.

2.12. Сформулировать задачу о назначениях, бинарная формулировка которой приведена в § 2.5, как задачу о взвешенной рёберной упаковке на двудольном неориентированном графе.

2.13. Изменить формулировку задачи о расписании работ на машинах, предположив, что для выполнения работы i требуется q_i операций, и рассмотрев два случая:

а) операции работы i выполняются на различных машинах;

б) одна машина может выполнять больше чем одну операцию работы i ($q_i \le n;\ i = \overline{1,m}$).

2.14. Показать, что функция $f_j(x_j)$ в задаче с фиксированными доплатами вогнута (см. § 2.7).

2.15. Сформулировать задачу об определении мощностей предприятий для случая, когда стоимости задаются вогнутыми функциями (см. § 2.8). Использовав кусочно-линейную аппроксимацию функций, дать линейное представление этой задачи.

2.16. Рассмотрим некоторый продукт, стоимость единицы которого изменяется с течением времени. Единица только что изготовленного продукта имеет стоимость δ, а единица продукта давности в k периодов имеет стоимость $c_k\delta$ ($k = \overline{l,n}$), где $1 = c_1 > c_2 > \ldots > c_n$. Фирма располагает a_k единицами продукта давности в k периодов ($k = \overline{0, n-m}$). В период $(n-m+i)$ фирма должна поставить b_i единиц продукта различной давности ($i = \overline{i,m}$). Требуется минимизировать суммарную стоимость поставленного продукта.

Сформулировать эту задачу как задачу о потоке минимальной стоимости.

2.17. Многопродуктовая задача о потоке минимальной стоимости. Рассмотрим задачу о перевозке нескольких продуктов. Пусть для продукта k заданы величины: V_1^k — множество источников; V_3^k-множество стоков; a_i^k, $i \in V_1^k$, — предложения; b_i^k, $i \in V_3^k$, — спросы; $c_{i,j}^k$ — стоимость перевозки единицы продукта k по дуге (i,j) ($k = \overline{1,p}$). Через $x_{i,j}^k$ обозначим величину дугового потока продукта k по дуге (i,j). Суммарный дуговой поток по дуге (i,j) не должен превышать пропускной способности этой дуги, т. е. $\sum\limits_{k=1}^{p} x_{i,j}^k \le d_{i,j}$, $(i,j) \in E$.

Сформулировать рассматриваемую задачу о перевозке как многопродуктовую задачу о потоке минимальной стоимости.

2.18. Дать интерпретацию всех понятий, введенных при рассмотрении сетевого графика (§ 2.10), на примере ориентированного графа изображенного на рис. 2.7.

2.19. Показать, что задача о нескольких коммивояжёрах эквивалентна ЦЛ-задаче, сформулированной в § 2.11.

2.20. Под неориентированным графом понимают математическую систему, состоящую из двух множеств V, E и отображения Γ множества E в $V \wedge V$. Элементы V и E называются соответственно вершинами и ребрами неориентированного графа, а Γ — отображением инцидентности. Если $\Gamma(e) = (v \wedge w)$, то вершины v и w называются граничными точками ребра e.

Вершинно-рёберная матрица инциденций A неориентированного графа $G = (V, E)$ представляет собой бинарную $|V| \times |E|$-матрицу (слово "вершинно-рёберная" часто опускается). Столбец a_j матрицы A, представляющий ребро графа $e_j = (v_k, v_l)$, состоит из элементов $a_{i,j}$, где

$$a_{i,j} = \begin{cases} 0, & i \ne k \wedge i \ne l, \\ 1, & i = k \vee i = l. \end{cases} \quad i = \overline{1,|V|};\ j = \overline{1,|E|}.$$

Пусть $G = (V, E)$ — неориентированный граф, являющийся простым циклом. Показать, что при подходящей нумерации вершин и рёбер вершинно-рёберную матрицу инциденций графа G можно записать в виде $|V| \times |V|$-матрицы A с элементами

$$a_{1,i} = a_{1,|V|} = 1,\ a_{1,j} = 0 \text{ в противном случае.}$$

$$a_{i,i-1} = a_{i,i} = 1,\ a_{i,j} = 0 \text{ в противном случае } (i \ge 2).$$

Доказать, что $|A| \ne\!\!\leftarrow\!\!\rightarrow |V|$ нечётно и что в этом случае $|A| = 2$.

2.21. Последовательность различных рёбер $(e_{j,1}, e_{j,2}, \ldots, e_{j,p})$, где ребра a_{j_t} и $e_{j_{t+1}}$ смежны ($t = \overline{1, p-1}$), называется путём. Для неориентированного графа $G = (V, E)$ и $i \in V$ определим $E^i = \{e_j \mid$

существует путь через вершину i, содержащий $e_j\}$, $V^i = \{t \mid$ существует $r \in V$ такое, что $(r,t) \in E^i\}$. Подграф $G^i = (V^i, E^i)$ называется компонентой графа G.

Найти различные компоненты графа G по его матрице инциденций

$$A = \begin{pmatrix} 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \end{pmatrix}.$$

2.22. Под ориентированным графом понимают математическую систему, состоящую из двух множеств V, A и отображения Δ множества A в $|V| \times |V|$. Элементы V и A называются соответственно вершинами и дугами ориентированного графа, а Δ — отображением ориентированной инцидентности. Если $\Delta(a) = (v, w)$, то v называется начальной вершиной дуги a, а w — конечной вершиной.

На плоскости изобразить ориентированные графы без параллельных дуг с тремя вершинами, имеющие соответственно $0, 1, 2, \ldots, 9$ дуг.

2.23. Вершинно-дуговая матрица инциденций $A = (A_{i,j})$ ориентированного графа $G = (V, E)$ без петель определяется как

$$a_{i,j} = \begin{cases} -1,\ i = l,\ e_j = (v_k, v_l),\ k \neq l, \\ 0,\ i \neq k \wedge i \neq l, \\ 1,\ i = k, e_j = (v_k, v_l),\ k \neq l, \end{cases} \qquad i = \overline{1, |V|};\ j = \overline{1, |E|}.$$

(слово "вершинно-дуговая"часто опускается).

Построить ориентированный граф, имеющий матрицу инциденций

$$A = \begin{pmatrix} 0 & 1 & 0 & 1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 0 & 0 & -1 & -1 & 0 \\ 0 & -1 & 1 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & -1 & -1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & -1 \\ 0 & 0 & 0 & 0 & 0 & -1 & -1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 1 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 \\ -1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

2.24. Для ориентированного графа, изображенного на рис. 2.7, построить матрицу инциденций.

Решение некоторых упражнений

2.7, в). Условие $x > 0 \to (g(x) \leq a \vee g(x) \geq b)$ можно записать как

$$x \leq dy_1,$$

$$g(x) \leq a + (u - a)(1 - y_1) + (u - a)y_2,$$

$$g(x) \geq b + (l - b)(1 - y_2) + (l - b)(1 - y_2),$$

$$x \geq 0, y_1 = 0, 1, y_2 = 0, 1,$$

где a, b, d, l, u — фиксированные действительные числа (отличные от $-\infty, +\infty$), $l \leq g(x) \leq u$ для $0 \leq x \leq d$, $u - a > 0$, $l - b < 0$.

Следует дважды найти нижнюю и верхнюю границы. Например, $g(x) - a \leq u - a$, $g(x) - a - (u - a)(1 - y_1) \leq u - a$.

2.19. Связь между комбинаторной и целочисленной задачами устанавливается взаимно однозначным соответствием: дуга (i, j) отсутствует во всех подтурах $\longleftrightarrow x_{i,j=0}$; дуга (i, j) находится в некотором подтуре $\longleftrightarrow x_{i,j} = 1$.

Рассмотрим допустимый набор из y подтуров. Эти подтуры определяют значения всех переменных $x_{i,j}$ $(0 \leq i \neq j \leq n)$. Если пункт i есть q-й в некотором подтуре $(1 \leq q \leq p)$, то полагаем $z_i = q$ $(i = \overline{1, n})$. Тогда неравенство $z_i - z_j + p_{i,j} \leq p - 1$ для $x_{i,j} = 1$ превращается в равенство, а для $x_{i,j} = 0$

оно справедливо в силу выбора значений z_i ($1 \le i \ne j \le n$). Легко проверить, что для так определённых значений переменных $x_{i,j}$, y, z_i, выполняются и все остальные ограничения ЦЛ-задачи.

Теперь покажем, что допустимое решение ЦЛ-задачи определяет допустимый набор подтуров.

Из условий задачи следует, что все переменные $x_{i,j}$ являются бинарными. Рассмотрим какую-нибудь переменную $x_{r_0,r_1}, x_{r_0,r_1} = 1, r_1 \ne 0$. Тогда существует такое единственное r_2, что $x_{r_1,r_2} = 1$. Покажем, что для некоторого l будет иметь место $r_l = 0$ (аналогично доказывается, что подтур начинается в пункте 0).

Если это не так, то для некоторых k и l будем иметь $r_k = r_l, 1 \le k, k+1 \le l$. Поскольку все r_i отличны от 0, то $z_{r_i} - z_{r_{i+1}} + px_{r_i,r_{i+1}} \le p-1 \to z_{r_i} - z_{r_{i+1}} \le -1$ ($i = \overline{k, l-1}$). Суммируя последние неравенства от $i = k$ до $i = l-1$, получаем $z_{r_k} - z_{r_l} \le (l-k) \times (-1) \to 0 \le k - l < -1$. Таким образом, получено противоречие.

Покажем, что каждый подтур содержит не более p пунктов. Предположим, что в подтур входят пункты $r_1, r_2, \ldots, r_{p+1}$. Суммируя, как и раньше, получаем $z_{r_1} - z_{r_{p+1}} \le -p \to z_{r_{p+1}} - z_{r_1} \ge p$, что противоречит неравенству $z_{r_{p+1}} - z_{r_1} + px_{r_{p+1},r_1} \le p-1$.

Глава 3

ПРЕОБРАЗОВАНИЯ ЗАДАЧ ЦЕЛОЧИСЛЕННОЙ ОПТИМИЗАЦИИ

В данной главе для задач целочисленной оптимизации рассматриваются важнейшие преобразования, позволяющие перейти от одной задачи оптимизации к другой, сохранив при этом связь между оптимальными решениями исходной и преобразованной задач. После выполнения преобразования могут измениться число ограничений, число переменных, условие целочисленности (дискретности) переменных, множество допустимых решений, множество оптимальных решений, целевая функция. Наконец, преобразованная задача может быть сформулирована в терминах, отличных от терминов исходной задачи. Всякое преобразование переводит один класс задач оптимизации в другой. Целесообразность того или иного преобразования определяется каждый раз конкретной ситуацией.

Рассматриваемые преобразования, как правило, естественным образом распараллеливаются. В качестве параметра, по которому производится распределение данных задачи между параллельно работающими процессорами, могут выступать число ограничений, число переменных, число строк или столбцов матрицы, длина однородного массива или какая-нибудь другая характеристика задачи. При этом очень часто получаются параллельные ветви, слабо связанные между собой или совсем несвязанные. В результате счёт ускоряется во столько раз, каково число процессоров. То или иное распределение данных задачи зависит от вычислений, которые следует выполнить. При распараллеливании вычислений считаем, что размеры задачи на порядок больше числа параллельно работающих процессоров.

3.1 ОБЩИЕ ЗАМЕЧАНИЯ

Преобразование задачи оптимизации. Под преобразованием задачи оптимизации понимаем какую-либо операцию, выполняемую над ее элементами: переменными, ограничениями, множеством допустимых решений, множеством оптимальных решений, целевой функцией. В результате выполнения преобразования получается новая задача оптимизации.

Цель преобразования состоит в нахождении такого вида задачи, который облегчает её анализ или её численное решение. Зачастую преобразование задачи позволяет установить свойства этой задачи, неочевидные в её первоначальной формулировке. Использование математического обеспечения ЭВМ при решении прикладных задач, как правило, требует преобразования исходной задачи в задачу, для которой уже готовы программные средства.

Переход от задачи оптимизации P_1, для которой заданы допустимое множество X_1 и целевая функция $f_1(x)$, $x \in X_1$, к задаче оптимизации P_2, для которой также известны допустимое множество X_2 и целевая функция $f_2(x)$, $x \in X_2$, можно осуществить при помощи следующих преобразований.

А. Сильно эквивалентное преобразование. Существует взаимно однозначное соответствие между допустимыми решениями задач P_1 и P_2, устанавливающее взаимно однозначное соответствие и между оптимальными решениями этих задач.

Б. Эквивалентное преобразование. Существует взаимно однозначное соответствие между оптимальными решениями задач P_1 и P_2.

В. Слабо эквивалентное преобразование. Хотя бы одно оптимальное решение задачи P_1 или его образ находится среди оптимальных решений задачи P_2 (условие $X_1^* \neq \varnothing$ влечёт $(T_1^* X_1^*) \cap X_2^* \neq \varnothing$).

Г. Расширение. Допустимое множество X_1 задачи P_1 или его образ расширяется до допустимого множества X_2 задачи $P_2((T_1 X_1) \subset X_2)$.

Например, Л-задача оптимизации является расширением (линейным расширением) как целочисленной, так и смешанной целочисленной линейных задач. Это преобразование состоит в исключении условий целочисленности на переменные.

Исключение каких-либо ограничений задачи приводят к новой задаче оптимизации с более широким множеством допустимых решений. Так, исключение условий неотрицательности на базисные переменные в ЦЛ-задаче

$$\max\{cx \mid Ax = b,\ x \geq 0,\ x \equiv 0\},$$

приводит к целочисленной линейной задаче над конусом (коническое расширение).

После перехода от комбинаторной задачи оптимизации к ее целочисленной формулировке часто используется выпуклое расширение образа $T_1 X_1$, т. е. множество X_2 представляет собой выпуклую оболочку точек, соответствующих допустимым решениям комбинаторной задачи.

Д. Сужение. Допустимое множество X_1 задачи P_1 сужается до допустимого множества X_2 задачи P_2 или до его образа $(X_1 \supset (T_2 X_2))$.

Если в примерах, приведённых в п. Г, поменять местами задачи P_1 и P_2, то получим соответствующие примеры сужений. В методе ветвей и границ, например, каждая текущая подзадача представляет собой результат сужения исходной задачи, состоящего из нескольких последовательно выполненных сужений.

Е. Изменение целевой функции. Целевая функция $f_2(x)$ задачи $P_2, x \in X_2$. по каким-либо правилам получается из целевой функции $f_1(x)$ задачи $P_1, x \in X_1$. Это преобразование может возникнуть в каждом из пяти вышерассмотренных случаев.

Изменяя целевую функцию, можно учитывать, например, различные ограничения задачи (см. § 3.2). От задачи P_1, в которой минимизируется функция $f_1(x)$ на множестве X, можно перейти к задаче P_2, в которой уже максимизируется функция $f_2(x) = -f_1(x)$ на X. Это преобразование является сильно эквивалентным.

Часто говорят, что задачи оптимизации P_1 и P_2 эквивалентны, не различая случаев А, Б, В. Как правило, из самого преобразования задачи ясно, какой из шести перечисленных случаев имеет место. Часто расширение множества допустимых решений достигается с помощью релаксации (ослабления) ограничений задачи. В этих случаях такое расширение называют релаксацией. Примерами релаксаций являются уже упоминавшиеся линейное и коническое расширения. Если оптимальное решение релаксированной задачи P_2, является допустимым решением задачи P_1, то оно оптимально и для P_1 (это легко проверить).

Простейшие преобразования. В качестве примеров, иллюстрирующих понятие преобразования задачи оптимизации, рассмотрим несколько простейших преобразований, часто используемых в теории и в приложениях. Переход от исходной задачи к преобразованной может состоять из конечного числа последовательно выполняемых простейших преобразований.

1. Переход к неотрицательным переменным. Переменную x_j, для которой $l_j \leq x_j \leq u_j$ (l_j, u_j — произвольно заданные действительные числа, $l_j \leq u_j$), можно заменить либо переменной x'_j изменяющейся в интервале $[0, u'_j]$, либо переменной x''_j со значениями из интервала $[0, 1]$. Формулами преобразования будут:

$$x_j = x'_j + l_j,\ 0 \leq x'_j \leq u'_j = u_j - l_j;$$

$$x_j = (u_j - l_j)x''_j + l_j,\ 0 \leq x''_j \leq 1.$$

Для $l_j = -\infty$ и $u_j = +\infty$ будем иметь соответственно

$$x_j = -x'_j + u_j,\ 0 \leq x'_j < +\infty,$$

$$x_j = x'_j + l_j,\ 0 \leq x'_j < +\infty.$$

При $l_j = -\infty, u_j = +\infty$ переменная x_j, произвольного знака заменяется двумя неотрицательными переменными, а именно:

$$x_j = x_j^+ - x_j^-,\ x_j^- \cdot x_j^+ = 0,\ 0 \le x_j^- < +\infty,$$

$$0 \le x_j^+ < +\infty,\ \text{где } x_j^- = -\min\{0, x_j\},\ x_j^+ = \max\{0, x_j\}.$$

Для каждого базисного решения Л-задачи оптимизации $\min\{cx \mid Ax = b, x \ge 0\}$ нелинейное условие вида $x_j^+ \cdot x_j^- = 0$ выполняется, так как переменные x_j^- и x_j^+ не могут быть одновременно базисными.

2. Переход к бинарным переменным. Целочисленную, дискретную и непрерывную переменные, изменяющиеся в конечном интервале, можно заменить системой бинарных переменных (см. § 2.1).

3. Преобразование ограничений. Уравнение $g_i(x) = 0$ эквивалентно двум неравенствам $g_i(x) \le 0$, $g_i(x) \ge 0$; введя слабые переменные $y_i, y_i \ge 0$, и $z_i, z_i \ge 0$, можно преобразовать неравенства $g_i(x) \le 0$ и $g_i(x) \ge 0$ соответственно в уравнения $g_i(x) + y_i = 0$ и $g_i(x) - z_i = 0$ $(x \in R_n)$.

4. Преобразование линейных ограничений. Система неравенств $Dx \le d$ эквивалентна системе неравенств $-Dx \ge -d$. Система уравнений $Dx = d$ эквивалентна системе неравенств $Dx \ge d, -eDx \ge -ed$. Здесь d — m-вектор; B — $m \times n$-матрица; $e = (1, 1, \ldots, 1) \in R_m$, $x \in R_n$. Таким образом, векторное неравенство $Ax \ge b$ можно рассматривать как общую форму задания линейных ограничений.

5. Агрегация ограничений. Под агрегацией двух уравнений, входящих в состав ограничений задачи оптимизации, понимаем замену этих двух уравнений одним уравнением, эквивалентным им (см., например, § 3.2). Агрегацию трёх уравнений можно осуществить, например, выполнив дважды агрегацию двух уравнений, и т. д.

6. Переход к неотрицательной матрице ограничений. Рассмотрим систему ограничений $Ax = b$, $0 \le x \le d$ $(0 < d_j < +\infty\ \forall j)$. Если в столбце a_j матрицы A для некоторого i существует отрицательный элемент $a_{i,j}$, то заменяем $a_j x_j$ на $a_{j,1}x_{j,1} + a_{j,2}x_{j,2}$, $0 \le x_{j,1} \le d_j$, $0 \le x_{j,2} \le d_j$, где $a_{i,j,1} = \min\{0, a_{i,j}\}\forall i$, $a_{i,j,2} = \max\{0, a_{i,j}\}\forall i$. Затем добавляем ограничение $x_{j,2} - x_{j,1} = 0$. Используя преобразование $y_{j,1} = d_j - x_{j,1}$, для всех столбцов рассматриваемого вида, получаем неотрицательную матрицу ограничений A'. Заметим, что если $b \ge 0$, то и $b' \ge 0$.

3.2 УМЕНЬШЕНИЕ РАЗМЕРОВ ЗАДАЧИ

Агрегация двух линейных уравнений. Рассмотрим систему из двух уравнений $a_1 x - b_1 = 0$, $a_2 x - b_2 = 0$, $x \in B_n$, где данные a_1, a_2, b_1, b_2 целочисленные $(a_1 \in Z_n, a_2 \in Z_n)$. Полагаем

$$a_{1,j}^- = \min\{0, a_{i,j}\}\forall j,\ a_{1,j}^+ = \max\{0, a_{i,j}\}\forall j,$$

$$\lambda^- = \sum_{j=1}^{n} a_{1,j}^- - b_1,\ \lambda^+ = \sum_{j=1}^{n} a_{1,j}^+ - b_1,\ \lambda = \max\{|\lambda^-|, |\lambda^+|\}.$$

Рассмотрим агрегированное уравнение $(a_1 + \alpha a_2)x - b_1 - \alpha b_2 = 0$, где α — произвольное целое число, $|\alpha| > \lambda$.

Очевидно, что любой бинарный вектор x', удовлетворяющий системе из двух уравнений, удовлетворяет и агрегированному уравнению. Если x' удовлетворяет агрегированному уравнению, то $a_1 x' - b_1 = -\alpha(a_2 x' - b_2)$. Правая часть последнего равенства представляет собой целое число, умноженное на α, и по модулю меньше $|\alpha|$ (в силу выбора α и целочисленности данных). Следовательно, правая часть равна нулю. Это означает, что $a_2 x' - b_2 = 0$ и $a_1 x' - b_1 = 0$, т. е. вектор x' удовлетворяет системе из двух уравнений.

Аналогично система из двух уравнений с целочисленными коэффициентами $a_1 x - b_1 = 0$, $a_2 x - b_2 = 0$, $0 \le x \le d$, $x \equiv 0$ преобразуется в эквивалентное ей уравнение $(a_1 + \alpha a_2)x - b_1 - \alpha b_2 = 0$, $0 \le x \le d$, $x \equiv 0$.

Многократное применение только что описанного преобразования уменьшает число ограничений, но, вообще говоря, вызывает сильный рост коэффициентов по модулю в агрегированном уравнении. Можно добиться некоторого уменьшения коэффициентов, использовав преобразование, в котором каждое из двух уравнений умножается на веса, отличные от 1.

Агрегация двух нелинейных уравнений. Теперь покажем, как можно заменить два нелинейных уравнения $r(x) = 0, s(x) = 0$ одним эквивалентным им уравнением $\alpha r(x) + \beta s(x) = 0 (x \in R_n)$.

Предположим, что функции $r(x)$ и $s(x)$ принимают целочисленные значения для всех $x \in S \subseteq R_n$, где S — произвольное множество. Определим следующие величины:

$$i_1 = \min\{0, \inf\{r(x) \mid \operatorname{sign}(\alpha)s(x) \le -|\alpha|,\ x \in S\}\},$$

$$i_2 = \min\{0, \inf\{r(x) \mid \operatorname{sign}(\alpha)s(x) \ge |\alpha|,\ x \in S\}\},$$

$$i_3 = \min\{0, \inf\{s(x) \mid \operatorname{sign}(\beta)r(x) \le -|\beta|,\ x \in S\}\},$$

$$i_4 = \min\{0, \inf\{s(x) \mid \operatorname{sign}(\beta)r(x) \ge |\beta|,\ x \in S\}\},$$

$$s_1 = \max\{0, \sup\{r(x) \mid \operatorname{sign}(\alpha)s(x) \le -|\alpha|,\ x \in S\}\},$$

$$s_2 = \max\{0, \sup\{r(x) \mid \operatorname{sign}(\alpha)s(x) \ge |\alpha|,\ x \in S\}\},$$

$$s_3 = \max\{0, \sup\{s(x) \mid \operatorname{sign}(\beta)r(x) \le -|\beta|,\ x \in S\}\},$$

$$s_4 = \max\{0, \sup\{s(x) \mid \operatorname{sign}(\beta)r(x) \ge |\beta|,\ x \in S\}\}.$$

При этом считаем, что для пустого множества S и для произвольной функции $f(x)$

$$\inf\{f(x) \mid x \in S = \varnothing\} = +\infty,$$

$$\sup\{f(x) \mid x \in S = \varnothing\} = -\infty,$$

$$\operatorname{sign}(\gamma) = \begin{cases} -1, & \gamma < 0, \\ 0, & \gamma = 0, \\ 1, & \gamma > 0. \end{cases}$$

Теперь сформулируем условия агрегации.

Пусть функции $r(x)$ и $s(x)$ принимают целочисленные значения для всех $x \in S \subseteq R_n$, а α и β — взаимно простые целые числа, для которых истинно логическое выражение

$$(-\alpha < i_4 \vee -\alpha > s_4 \vee \beta < i_1 \vee \beta > s_1) \wedge$$

$$\wedge(\alpha < i_3 \vee \alpha > s_3 \vee -\beta < i_2 \vee -\beta > s_2).$$

Тогда

$$X = \{x \mid r(x) = 0, s(x) = 0, x \in S\} = \{x \mid \alpha r(x) + \beta s(x) = 0,$$

$$x \in S\} = X(\alpha, \beta).$$

Докажем это. Ясно, что $X \subseteq X(\alpha, \beta)$. Если $X(\alpha, \beta) = \varnothing$, то утверждение очевидно. Пусть x — произвольная точка, $x \in X(\alpha, \beta)$. Тогда $\alpha r(x) + \beta s(x) = 0, \alpha r(x) = p, \beta s(x) = -p$, где p — целое. Поскольку $r(x)$ и $s(x)$ — целые числа, то $\alpha \mid p$ и $\beta \mid p$. Так как числа α и β взаимно простые, то $p = \alpha\beta\theta$ для некоторого целого θ. Тогда $r(x) = \beta\theta, s(x) = -\alpha\theta$. Выполнение каждого неравенства из первой скобки логического выражения влечёт, что $\theta \le 0$. Выполнение же каждого неравенства из второй скобки логического выражения даёт, что $\theta \ge 0$. Следовательно, $\theta = 0$, $p = 0$, $r(x) = 0$ и $s(x) = 0$, $x \in X$. Поскольку x — произвольная точка из $X(\alpha, \beta)$, то получаем $X(\alpha, \beta) \subseteq X$.

Рассмотрим, например, случай $-\alpha < i_4$. Остальные семь случаев доказываются аналогично. Предположим, что $\theta > 0$, т. е. $\theta \ge 1$. Из $x \in X(\alpha, \beta)$ следует, что $r(x) = \beta\theta, s(x) = -\alpha\theta$. Для $\theta \ge 1$ получаем $r(x) = \beta, 2\beta, \ldots$. Это даёт $\beta < 0 \to r(x) \le \beta$, $\beta > 0 \to r(x) \ge \beta$, т. е. $\operatorname{sign}(\beta)r(x) \ge |\beta|$. Далее имеем $-\alpha\theta = s(x) \ge \inf\{s(z) \mid \operatorname{sign}(\beta)r(z) \ge |\beta|,\ z \in S\} \ge i_4 > -\alpha$, что даёт $-\alpha\theta > -\alpha \to \theta < 1$ (так как $-\alpha < 0$). Полученное противоречие завершает доказательство рассматриваемого случая.

Описанное преобразование может быть использовано для уменьшения числа ограничений целочисленной задачи $\max\{f(x) \mid g_i(x) = 0,\ i = \overline{1,m},\ x \in S \subseteq Z_n\}$, где $g_i(x) \equiv 0$, $i = \overline{1,m}$. При нахождении множителей α и β можно заменить инфимум и супремум соответственно на нижнюю и верхнюю границы, даваемые каким-либо образом релаксированной задачей. Коэффициенты α и β могут оказаться большими по модулю.

Задача об агрегации, рассматриваемая в целочисленной оптимизации, состоит в доказательстве существования или в поиске такого вектора $t, t \in Z_m$, что агрегированное уравнение $tAx = tb$ и заданная

система уравнений $Ax = b$ с целочисленными данными A, b имеют одно и то же множество неотрицательных целочисленных решений. Агрегация ограничений рассматривается, например, в [20, 53].

Учёт ограничений при помощи штрафов. Рассмотрим Ц-задачу

$$\min\{g(x) \mid g_i(x) \le 0,\ i = \overline{1, m'},\ g_i(x) = 0,$$
$$i = \overline{m'+1, m},\ a_j \le x_j \le b_j,\ j = \overline{1, n},\ x \in Z_n\}.$$

Преобразование этой задачи в задачу минимизации функции без ограничений основано на введении определённых функций, называемых функциями штрафа, и на включении их в целевую функцию. Применение такого преобразования к части ограничений даёт новую задачу с меньшим числом ограничений.

Для исключения ограничений $a_j \le x_j \le b_j$, $x_j \equiv 0$ может быть использована функция штрафа вида

$$\varphi_j(x_j) = \begin{cases} (x_j - a_j)^2,\ x_j < a_j, \\ \sin^2 \pi x_j,\ a_j \le x_j \le b_j, \\ (x_j - b_j)^2,\ x_j > b_j\ (j = \overline{1, n}). \end{cases}$$

Тогда получаем новую задачу

$$\min\Big\{g(x) + \sum_{j=1}^{n} \alpha_j \varphi_j(x_j) \mid g_i(x) \le 0,\ i = \overline{1, m'},\ g_i(x) = 0,\ i = \overline{m'+1, m},\ x \in R_n\Big\}.$$

Здесь α_j, $j = \overline{1, n}$, — большие положительные числа.

Удаляя оставшиеся ограничения, получаем задачу

$$\min\Big\{f(x) = g(x) + \sum_{j=1}^{n} \alpha_j \varphi_j(x_j) + \sum_{i=1}^{m'} \mu_i (\max\{0, g_i(x)\})^2 + \sum_{i=m'+1}^{m} \mu_i (g_i(x))^2 \mid x \in R_n\Big\},$$

где μ_i, $i = \overline{1, m}$, — достаточно большие положительные числа.

Пусть X — множество допустимых решений исходной задачи. Тогда $\min\limits_{x \in R_n} f(x)$ стремится к $\min\limits_{x \in X} g(x)$ когда α_j стремится к $+\infty$, $j = \overline{1, n}$, μ_i стремится к $+\infty$, $i = \overline{1, m}$.

Для рассматриваемых функций штрафа даже при достаточно больших α_j, μ_i множества оптимальных решений исходной и преобразованной задач могут не совпадать (например, $\min\{-x \mid 0 \le x \le 1,\ x \equiv 0\}$). Как показывает практический опыт, сходимость по α_j, μ_i является быстрой.

Применение метода штрафных функций в целочисленной оптимизации рассматривается, например, в [78].

Уменьшение размеров матрицы ограничений. Рассмотрим задачу о покрытии

$$\min\{cx \mid Ax \ge e, x_j = 0, 1, j = \overline{1, n}\}$$

и задачу о разбиении

$$\min\{cx \mid Ax = e,\ x_j = 0, 1,\ j = \overline{1, n}\}\ (e = (1, 1, \ldots, 1) \in R_m).$$

Будем называть их соответственно П-задачей и Р-задачей. Предположим, что в этих задачах $c_j > 0$, $j = \overline{1, n}$.

В П- и Р-задачах часто априори можно исключить из матрицы A некоторые строки или столбцы. Приведем несколько очевидных правил исключения. Будем обозначать строку i и столбец j матрицы A соответственно через r_i и a_j.

1. П- и Р-задачи. Если $r_i = 0$ для некоторого i, то не существует допустимых решений.

2. П- и Р-задачи. Если $r_i = e_k$ для некоторых i и k (e_k — k-й единичный вектор), то $x_k = 1$ для каждого допустимого решения и столбец a_k может быть исключен. Также может быть вычеркнута каждая строка r_t такая, что $t \in P_k = \{i \mid a_{i,k} = 1\}$.

2а. Р-задача. В дополнение к исключениям правила 2 должен быть вычеркнут каждый столбец a_q, для которого $q \ne k$ и $a_{t,q} = a_{t,k} = 1$ для $t \in P_k$.

3. П- и Р-задачи. Если $r_t \geq r_p$ для некоторых t и p, то строка r_t может быть вычеркнута.

3а. Р-задача. В дополнение к вычёркиванию строки r_t по правилу 3 должен быть исключен каждый столбец a_k такой, что $a_{t,k} = 1$ и $a_{p,k} = 0$.

4. П- и Р-задачи. Если для множества столбцов S и столбца a_k выполняются соотношения $\sum\limits_{j \in S} a_j = a_k$ и $\sum\limits_{j \in S} c_j \leq c_k$, то столбец a_k может быть исключен.

4а. П-задача. Если для множества столбцов S и столбца a_k выполняются неравенства $\sum\limits_{j \in S} a_j > a_k$ и $\sum\limits_{j \in S} c_j \leq c_k$, то столбец a_k может быть исключен.

Уменьшение размеров матрицы ограничений рассматривается, например, в [69].

3.3 ПРЕОБРАЗОВАНИЕ БИНАРНОЙ ЛИНЕЙНОЙ ЗАДАЧИ В ЗАДАЧИ О ПОКРЫТИИ, РАЗБИЕНИИ И УПАКОВКЕ

Преобразование бинарной линейной задачи в задачу о разбиении. В преобразованной задаче каждый столбец матрицы ограничений A имеет не больше трёх единиц.

Рассмотрим БЛ-задачу $\min\{cx \mid Ax = u, x_j = 0, 1, \forall j\}$, где $b \geq 0$. Сначала преобразуем её в задачу с неотрицательной матрицей ограничений. Если в столбце a_j для некоторого i существует отрицательный элемент $a_{i,j}$, то заменяем $a_j x_j$ на $a_{j,1}x_{j,1} + a_{j,2}x_{j,2}$, $x_{j,1} = 0, 1$, $x_{j,2} = 0, 1$, где $a_{i,j,1} = \min\{0, a_{i,j}\} \forall i$, $a_{i,j,2} = \max\{0, a_{i,j}\} \forall i$. Затем добавляем ограничение $x_{j,2} - x_{j,1} = 0$ и полагаем $c_{j,1} = c_{j,2} = c_j/2$. Используя замену переменных $x_{j,1} = 1 - y_{j,1}$, $y_{j,1} = 0, 1$, для всех столбцов a_j рассматриваемого вида, получаем неотрицательную матрицу ограничений.

Следующее преобразование даёт бинарную матрицу ограничений не более чем с тремя единицами в каждом столбце. Заменим столбец a_j на $m \times t_{m,j}$-матрицу $A'_j = (a'_{i,k})$, где

$$a'_{i,k} = \begin{cases} 0, \ 1 \leq k \leq t_{i-1,j} \vee t_{i,j} < k \leq t_{m,j}, \\ 1, \ t_{i-1,j} < k \leq t_{i,j}, \end{cases}$$

$$i = \overline{1, m}; \ k = \overline{1, t_{m,j}}.$$

Здесь $t_{0,j} = 0$, $t_{i,j} = \sum\limits_{r=1}^{i} a_{r,j}$, $i = \overline{1, m}$.

Таким образом, если $a_j = \begin{pmatrix} 2 \\ 0 \\ 3 \end{pmatrix}$, то $A'_j = \begin{pmatrix} 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 \end{pmatrix}$. Заметим, что единичный вектор не требует преобразования.

Свяжем переменную $x_{j,k}$ с k-м столбцом матрицы A'_j и положим $c_{j,k} = \frac{c_j}{t_{m,j}}$ $\forall k$. Новая задача эквивалентна первоначальной, если $x_j = x_{j,k} \forall k$. Это условие достигается добавлением ограничений $x_{j,k} + x'_{j,k} = 1$, $x_{j,k+1} + x'_{j,k} = 1$, $x'_{j,k} = 0, 1$, при этом $x'_{j,k}$ входит в целевую функцию с коэффициентом 0 ($k = \overline{1, t_{m,j} - 1}$).

Теперь можно считать, что система ограничений уже приведена к системе уравнений вида $Ax = b$, где x — бинарный вектор; b — неотрицательный целочисленный вектор; A — бинарная матрица и $\sum\limits_{i=1}^{m} a_{i,j} \leq 3; \forall j$. Удобно рассматривать эту систему уравнений как задачу на m-вершинном неориентированном графе. Столбец a_j изображается петлей, ребром или треугольником (циклом, состоящим из трёх рёбер), если он содержит соответственно одну, две или три единицы. Допустимое решение теперь есть такой набор петель, рёбер и треугольников, что вершина i имеет степень инцидентности b_i, с этим набором.

При этом матрица A построена так, что каждое ребро или каждый треугольник инцидентен не более чем одной вершине, имеющей $b_i \geq 2$. Если a_j — такой единичный вектор, что $a_{i,j} = 1$ и $b_i \geq 2$, то добавляем ограничение $x_j + x_s = 1$, $x_s = 0, 1$, где слабая переменная x_s входит в целевую функцию с коэффициентом 0. Равенство $x_j + x_s = 1$ даёт новую вершину s, так что столбец a_j изображается

теперь ребром, а столбец, соответствующий x_s, — петлей. Таким образом, можно считать, что каждая петля, каждое ребро и каждый треугольник инцидентны, по крайней мере, одной вершине, имеющей $b_i = 1$.

Чтобы удовлетворить условию $b_i = 1$ для всех i, заменяем каждую вершину t, для которой $b_t \geq 2$, её b_t копиями, а каждое ребро и каждый треугольник, если они инцидентны вершине t, — соответственно их b_t копиями (как это показано на рис. 3.1).

Для задачи о разбиении уже получена система ограничений $A'y = e$, где y — бинарный вектор. При этом в каждом столбце матрицы A' содержится не более чем три единицы. Если столбец a_j "инцидентен" вершине t с $b_t \geq 2$, то копии для переменной x_j обозначаются $y_{j,1}, y_{j,2}, \ldots, y_{j,b_t}$, а $c_{j,k} = c_j \forall k$. Так как каждый столбец a_j инцидентен по крайней мере одной вершине, имеющей $b_i = 1$, условие $\sum\limits_{k=1}^{b_t} y_{j,k} \leq 1$ заведомо выполняется. Если дано решение y, то решение x с тем же значением целевой функции получается при помощи соотношения $x_j = \sum\limits_{k=1}^{t} y_{j,k}$. Также ясно, что из x можно получить y.

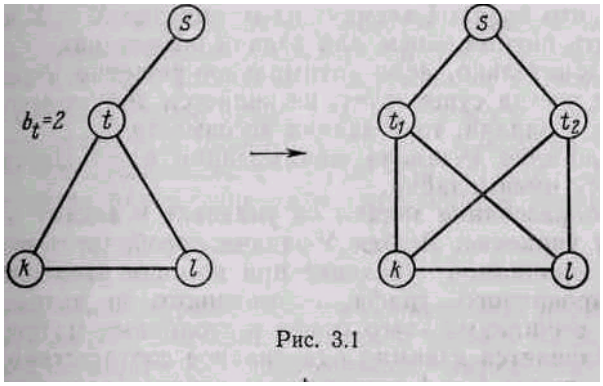


Рис. 3.1

Преобразование задачи о разбиении в задачи об упаковке и покрытии. Если задача о разбиении

$$\max\{z(x) = cx \mid Ax = e, x_j = 0, 1 \ \forall j\}$$

имеет допустимое решение, то множество её оптимальных решений совпадает с множеством оптимальных решений задачи об упаковке (У-задачи)

$$\max\{z'(x) = c'x \mid Ax \leq e, x_j = 0, 1 \ \forall j\},$$

где $c'_j = c_j + lt_j \forall j$, l и t_j — величины, определяемые ниже.

Пусть $\overline{z}$ — любая верхняя граница для значений z, получающихся для всех возможных значений бинарного вектора x, $\underline{z}$ — строгая нижняя граница значений z для всех допустимых решений Р-задачи, $l = \overline{z} - \underline{z}$. Определим $t_j = \sum\limits_{i=1}^{m} a_{i,j}; \forall j$ и $c'_j = c_j + lt_j \forall j$.

Пусть X и X' — множества допустимых решений соответственно Р- и У-задач. Заметим, что $X \subseteq X'$. Предположим, что $X' \backslash X \neq \varnothing$, иначе утверждение тривиально. Произвольно выберем два вектора x и $x', x \in X$ и $x' \in (X' \backslash X)$. Тогда получаем $c'x = cx + lm > \underline{z} + lm$, $c'x' \leq cx' + l(m-1) \leq \overline{z} + l(m-1) = \overline{z} + lm < c'x$. Это показывает, что никакой элемент из множества $X' \backslash X$ не может быть оптимальным для задачи об упаковке.

Следовательно, если оптимальное решение У-задачи, которое всегда существует, не является допустимым решением Р-задачи, то Р-задача несовместна.

Аналогично Р-задача минимизации преобразуется в П-задачу минимизации.

Преобразование задачи об упаковке в задачу о вершинной упаковке. Любая У-задача преобразуется в задачу о вершинной упаковке при помощи введения неориентированного графа, построенного по матрице A. Между вершинами итого графа и столбцами матрицы Л устанавливается взаимно однозначное соответствие, при этом вершины k и l соединяются ребром, если существует такое i, что $a_{i,k} = a_{i,l} = 1$. Вершине j приписывается вес c'_j $(j = \overline{1, n})$. Подмножество вершин является

независимым (никакие две вершины из подмножества не инцидентны одному и тому же ребру), если и только если среди соответствующих столбцов матрицы A не существует двух столбцов, имеющих 1 в одной и той же позиции. Последнее условие равносильно $Ax \le e$.

Преобразование бинарной линейной задачи в задачу о покрытии. Рассмотрим БЛ-задачу

$$\min\Big\{cx \mid \sum_{j=1}^{n} a_{i,j}x_j \ge b_i,\ i=\overline{1,m},\ x_j=0,1,\ j=\overline{1,n}\Big\}$$

и предположим, что все элементы $a_{i,j}$ матрицы A неотрицательные.

Каждое неравенство системы ограничений преобразуется по одной и той же схеме. Поэтому проиллюстрируем рассматриваемое преобразование на примере одного неравенства (исходного)

$$\sum_{j=1}^{n} a_j x_j \ge b,\ a_j \ge 0,\ j=\overline{1,n},\ b>0.$$

Рассмотрим все минимальные подмножества $J_k \subseteq J=\{1,2,\dots,n\}$, для которых $\sum\limits_{j\in J_k} a_j \ge b$ ($k=\overline{1,p}$) (множество J_k меньше множества J_l, если $J_k \subset J_l,\ k\ne l$). Тогда для выполнения исходного неравенства необходимо и достаточно, чтобы имело место соотношение

$$\sum_{k=1}^{p}\prod_{j\in J_k} x_j \ge 1.$$

Заменяя в последнем неравенстве сложение и умножение соответственно дизъюнкцией и конъюнкцией, получаем уже для логических переменных x_j логическое выражение, которое равно 1, если последнее неравенство выполняется, и равно 0, если нет. Многократно применяя преобразование $q\vee(r\wedge s)=(q\vee r)\wedge(q\vee s)$, можно привести рассматриваемое выражение к конъюнктивной нормальной форме $\bigwedge\limits_{i=1}^{v}\bigvee\limits_{j\in J'_i} x_j$.

Логическое выражение в конъюнктивной нормальной форме имеет значение 1 тогда и только тогда, когда каждая элементарная дизъюнкция имеет значение 1. Каждая элементарная дизъюнкция $\bigvee\limits_{j\in J_i} x_j = 1$ эквивалентна неравенству $\bigvee\limits_{j\in J'_i} x_j \ge 1$.

Следовательно, исходное неравенство $\sum\limits_{j=1}^{n} a_j x_j \ge b$ эквивалентно системе неравенств $\sum\limits_{j=1}^{n} a'_{i,j}x_j \ge 1$, $i=\overline{1,v}$. Здесь элемент $a'_{i,j}$ равен либо 0, либо 1 ($i=\overline{1,v}$, $j=\overline{1,n}$), v — число элементарных дизъюнкций в конъюнктивной нормальной форме.

Можно уменьшить размеры получившейся таким образом П-задачи (см. § 3.2).

Изложенные в этом параграфе преобразования рассматриваются, например, в [53, 69].

3.4 ЛИНЕЙНАЯ РЕЛАКСАЦИЯ ЗАДАЧ О ПОКРЫТИИ И РАЗБИЕНИИ

Линейные задачи $\min\{cx \mid Ax \ge e,\ x \ge 0\}$ и $\min\{cx \mid Ax = e, x \ge 0\}$, соответствующие П- и Р-задачам, будем называть ЛП- и ЛР-задачами. Предполагаем, что $c_j > 0$, $j \in N = \{1,2,\dots,n\}$ ($e=(1,1,\dots,1)$).

Столбец $j', j' \in J'$, называется избыточным относительно покрытия J', если множество $J'\backslash j'$ является также покрытием. Покрытие, содержащее один или несколько избыточных столбцов, называется избыточным. Покрытие, не являющееся избыточным, называется простым.

Столбец $j', j' \in J'$, будет избыточным относительно покрытия $J^{/}$ тогда и только тогда, когда $\sum\limits_{j\in J'} a_{i,j} \ge 2$, $i \in P_{j'}(P_{j'}=\{i \mid a_{i,j}=1,\ i=\overline{1,m}\}\forall j \in N)$.

Столбец $\hat{j}, \hat{j} \in J'$, не будет избыточным тогда и только тогда, когда $I(\hat{j}) = \{i \mid \sum\limits_{j\in J'} a_{i,j} = 1,\ i \in P_{\hat{j}}\} \ne \varnothing$.

Если x^* — оптимальное решение ЛП-задачи, то $0 \le x^* \le e$. Покажем это. Пусть x' — произвольное допустимое решение ЛП-задачи. Тогда x'', для которого $x''_j = \min\{1, x'_j\} \forall j$, также является допустимым решением ЛП-задачи, но уже с меньшим значением целевой функции, так как $c_j > 0$ для всех j.

Пусть вектор $\hat{x}$ такой, что $\hat{x} = \lceil x^*_j \rceil \, \forall j$. Вектор $\hat{x}$ определяет покрытие $\hat{J} = \{j \mid \hat{x}_j = 1, \ j \in N\}$. Покрытие $\hat{J}$ не обязательно должно быть оптимальным или даже простым.

Если покрытие $\widehat{J}$ избыточно, то за один раз исключаем какой-либо избыточный столбец. Это повторяем до тех пор, пока не получится простое покрытие. Получившееся простое покрытие не является единственным и зависит от порядка исключения столбцов. Каждое оптимальное покрытие всегда будет простым.

Пусть J' — простое покрытие. Тогда вектор x', для которого

$$x'_j = \begin{cases} 0, \ j \notin J', \\ 1, \ j \in J', \end{cases} \quad j \in N$$

есть крайняя точка множества допустимых решений ЛП-задачи.

Действительно, $|I(j)|$ неравенств выполняются как равенства для каждого $j \in J'$. Для различных j и j' из J' имеем $I(j) \cap I(j') = \varnothing$, следовательно, $|J'| \le m$, а линейная независимость $|J'|$ столбцов следует из их вида.

Добавляя $m - |J'|$ минус-единичных столбцов, соответствующих слабым переменным, для каждого простого покрытия J' можно построить базисную матрицу B:

а) выбираем переменную S_i, если $\sum\limits_{j \in J'} a_{i,j} \ge 2$;

б) для каждого $j \in J'$ такого, что $|I(j)| > 1$, произвольно выбираем среди $|I(j)|$ слабых переменных, соответствующих $I(j), |I(j)| - 1$ переменных.

Пример 3.1. Пусть дана матрица A

$$\begin{pmatrix} 1 & 1 & 1 & 0 & 1 & 0 & 1 & 1 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 1 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 \end{pmatrix}.$$

Множество $J = \{1, 2, 3, 4\}$ является покрытием. Первый и третий столбцы избыточны относительно этого покрытия. Исключаем первый столбец. Тогда покрытие $\{2, 3, 4\}$ простое: $I(2) = \{3\}$, $I(3) = \{5\}$, $I(4) = \{4, 7\}$. Вводим в базис столбцы, соответствующие слабым переменным s_1, s_2, s_6, s_7 (вместо s_7 может быть выбрано s_4. Тогда базисная матрица B имеет вид

$$\begin{pmatrix} 1 & 1 & 0 & -1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & -1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & -1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & -1 \end{pmatrix}.$$

Перестановкой строк из матрицы B можно получить такую матрицу $\widehat{B}$, что $\widehat{B^{-1}} = \widehat{B}$.

Покажем это. Строкой i матрицы $\widehat{B}$ служит та строка матрицы B, которая является единичным вектором e_i $(i = \overline{1, |J'|})$. Для $i = \overline{|J'| + 1, m}$ строкой i матрицы $\widehat{B}$ является строка из B, имеющая -1 в столбце i. Матрица $\widehat{B}$ имеет вид

$$\begin{pmatrix} I_1 & 0 \\ P & -I_2 \end{pmatrix},$$

где I_1 и I_2 — единичные матрицы порядков $|J'| \times |J'|$ и $(m - |J'|) \times (m - |J'|)$ соответственно. Легко проверить,

$$\widehat{B}\widehat{B} = \begin{pmatrix} I_1 & 0 \\ 0 & I_2 \end{pmatrix}.$$

Следовательно, $\widehat{B}^{-1} = B$. Существование обратной матрицы $\widehat{B}^{-1}$ влечёт существование обратной матрицы B^{-1}.

Выпишем для предыдущего примера матрицу

$$\widehat{B} = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & -1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & -1 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & -1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & -1 \end{pmatrix}$$

Пусть J' — разбиение. Тогда вектор x', для которого

$$x'_j = \begin{cases} 0, & j \notin J', \\ 1, & j \in J', \end{cases} \quad j \in N,$$

является крайней точкой множества допустимых решений ЛР-задачи.

Действительно, $|J'| \leq m$, так как $|I(j)| = |p_j| \geq 1 \forall j \in J'$. Столбцы линейно независимы, так как никакие два столбца не имеют положительных элементов в одной и той же позиции.

Базис В ЛР-задачи называется целочисленным, если $B^{-1}e$ — бинарный вектор. Два базиса смежны, если они отличаются только одним столбцом.

Сформулируем еще два свойства, опустив их доказательства. Для любого целочисленного базиса ЛР-задачи существует, по крайней мере, столько смежных допустимых целочисленных базисов, сколько и небазисных столбцов. Следует отметить, что из-за вырожденности целочисленного решения большое количество базисов соответствует одной и той же крайней точке.

Пусть x_1 — решение, связанное с базисной матрицей B_1 ЛР-задачи, а x_2 — другое решение, которое по значению целевой функции не хуже чем x_1. Пусть J_1 и J_2 — соответствующие разбиения. Тогда базисная матрица B_2, связанная с x_2, может быть получена из B_1 при помощи последовательности из $|J_1 \cap Q_2|$ симплекс-итераций таких, что каждое из решений является допустимым, целочисленным и не хуже, чем предшествующее. Здесь Q_2 — множество индексов небазисных столбцов, связанных с x_2.

Свойства задач о покрытии, разбиении и упаковке, а также алгоритмы их решения рассматриваются, например, в [53, 69].

3.5 ЗАДАЧА О ПОТОКЕ МИНИМАЛЬНОЙ СТОИМОСТИ

В этом параграфе показывается, что некоторые хорошо известные задачи являются частными случаями задачи о потоке минимальной стоимости, которая формулируется как задача оптимизации на ориентированном графе и представляет собой Л-задачу специальной структуры. Матрица ограничений задачи о потоке минимальной стоимости абсолютно унимодулярна, т. е. определитель всякой невырожденной квадратной подматрицы этой матрицы равен либо -1, либо $+1$. Для целочисленного столбца ограничений каждое допустимое базисное (в частности, и оптимальное базисное) решение такой Л-задачи является целочисленным. Таким образом, задача о потоке минимальной стоимости, рассматриваемая как ЦЛ-задача, может численно решаться симплекс-методом. Однако для решения этой задачи и её частных случаев существуют более эффективные алгоритмы, использующие свойства ориентированного графа.

Задача о потоке минимальной стоимости. Рассмотрим ориентированный граф $G = (V, E)$ (в частности, он может изображать транспортную сеть), где $V = \{1, 2, \ldots, m\}$, а $(i, j), (i, j) \in E$ — дуга, идущая из вершины i в вершину j. Предполагается, что граф G не содержит петель и параллельных дуг, т. е. дуг, имеющих одни и те же начальную и конечную вершины. Множество вершин V разбито на три непересекающихся подмножества: V_1 — источники (пункты отправления); V_2 — внутренние вершины (промежуточные пункты); V_3 — стоки (пункты назначения). Такой граф G с отмеченными вершинами называют сетью. Для каждого i из V определим множества $V(i) = \{j \mid (i, j) \in E\}$, $V'(i) = \{j \mid (j, i) \in E\}$. Будем считать, что на дугах сети заданы ограничения, т. е. каждой дуге (i, j) приписаны два числа 0 и $d_{i,j}$, $0 \leq d_{i,j} \leq +\infty$. Число $d_{i,j}$ называется пропускной способностью дуги (i, j).

Потоком в сети G называется множество чисел $\{x_{i,j} \mid (i,j) \in E\}$, удовлетворяющих линейным ограничениям

$$\sum_{j \in V(i)} x_{i,j} - \sum_{j \in V'(i)} x_{j,i} \begin{cases} \le a_i, \ i \in V_1, \\ = 0, \ i \in V_2, \\ \le -b_i, \ i \in V_3, \end{cases}$$
$$0 \le x_{i,j} \le d_{i,j}, \ (i,j) \in E.$$

Поток в сети G удобно изображать как действительнозначную функцию $f(e)$, определённую на всех дугах сети и удовлетворяющую этим линейным ограничениям. Задача о потоке минимальной стоимости состоит в нахождении потока, для которого стоимость $\sum\limits_{i \in V} \sum\limits_{j \in V(i)} c_{i,j} x_i$, минимальна.

Сформулированные условия отражают следующие факты. Во внутренних вершинах имеет место сохранение потока. В источниках разность между значением выходного потока и значением входного потока не превышает предложение. В стоках разность между значением входного потока и значением выходного потока не меньше спроса. Значение дугового потока по дуге (i,j) не отрицательное и не превышает пропускной способности этой дуги. Переменная $x_{i,j}$ входит в ограничения, соответствующие вершинам графа, только два раза: один раз с коэффициентом -1, а другой — с коэффициентом $+1$ (граф без петель). Очевидно, что для задачи, имеющей допустимое решение, с необходимостью выполняется неравенство $\sum\limits_{i \in V_i} a_i \ge \sum\limits_{i \in V_3} b_i$. Это неравенство превращается в равенство для задачи, в которой должны быть точно удовлетворены все спросы при использовании полностью всех предложений.

В формулировке задачи о потоке минимальной стоимости величины $c_{i,j}$ могут быть как отрицательными, так и неотрицательными, а величины $d_{i,j}$ — равными $+\infty$. Поэтому при рассмотрении этой задачи считается выполненным следующее естественное предположение. Если $((i_i, i_2), (i_2, i_3), \ldots, (i_k, i_1))$ — контур ориентированного графа G, то либо

а) $c_{i_1,i_2} + c_{i_2,i_3} + \ldots + c_{i_k,i_1} \ge 0$,

либо

б) $\min\{d_{i_1,i_2}, d_{i_2,i_3}, \ldots, d_{i_k,i_1}\} < +\infty$.

Если не выполняется ни а), ни б), то существует поток, дуговые потоки которого вдоль рассматриваемого контура могут быть произвольно большими, что даёт произвольно большую по абсолютному значению отрицательную стоимость.

Под путём и контуром в ориентированном графе G понимаем ориентированный путь и замкнутый ориентированный путь, направления которых совпадают с направлениями всех принадлежащих им дуг. Путь (контур) без повторяющихся вершин будем называть простым. Под потоком вдоль простого пути из вершины i в вершину j или вдоль простого контура понимаем функцию, принимающую на дугах соответствующего пути или контура одно и то же ненулевое значение и равную нулю на остальных дугах.

Теперь перейдем к рассмотрению частных случаев задачи о потоке минимальной стоимости (исходной задачи).

Задача о транспортировке. Эта задача получается из исходной, если $d_{i,j} = +\infty, (i,j) \in E$. Матрицей ограничений задачи является матрица инциденций ориентированного графа G.

Рассмотрим случай, когда $\sum\limits_{i \in V_i} a_i = \sum\limits_{i \in V_3} b_i$. Тогда неравенства, соответствующие вершинам графа, для любого допустимого решения выполняются как равенства. Проделав очевидные преобразования и переобозначения, можно записать следующую Л-задачу.

Максимизировать

$$\sum_{i \in V} \sum_{j \in V(i)} c_{i,j} x_{i,j}$$

при условиях

$$\sum_{j \in V(i)} x_{i,j} - \sum_{j \in V'(i)} x_{j,i} = b_i, \ i \in V, \ x_{i,j} \ge 0, \ (i,j) \in E.$$

Здесь $\sum\limits_{i \in V} b_i = 0$.

Задача, двойственная к этой, имеет вид: минимизировать

$$\sum_{i \in V} b_i y_i$$

при условиях

$$y_i - y_j \geq c_{i,j},\ (i,j) \in E.$$

Приложения только что сформулированных прямой и двойственной задач были рассмотрены, например, в § 2.10.

Транспортная задача с ограниченными пропускными способностями. Эта задача получается из исходной, если положить $V_2 = \varnothing$, $V'(i) = \varnothing$, $i \in V_1$, $V(i) = \varnothing$, $i \in V_3$. Рассматриваемая задача может быть записана следующим образом. Минимизировать

$$\sum_{i \in V_1} \sum_{j \in V(i)} c_{i,j} x_{i,j}$$

при условиях

$$\sum_{j \in V'(i)} x_{i,j} \leq a_i,\ i \in V_1, \quad \sum_{j \in V'(i)} x_{i,j} \geq b_i,\ i \in V_3,\ 0 \leq x_{i,j} \leq d_{i,j},\ (i,j) \in E.$$

Релаксация задачи о назначениях. Рассмотрим частный случай предыдущей задачи, который можно получить, если положить $a_i = 1, i \in V_1, b_i = 1, i \in V_3, d_{i,j} = +\infty, (i,j) \in E, |V_1| = |V_3|$. Из условия $|V_1| = |V_3|$ следует, что все ограничения, соответствующие вершинам графа, должны удовлетворяться как равенства. Следовательно, можно сформулировать следующую Л-задачу.

Минимизировать

$$\sum_{i \in V_1} \sum_{j \in V(i)} c_{i,j} x_{i,j}$$

при условиях

$$\sum_{j \in V(i)} x_{i,j} = 1,\ i \in V_1, \quad \sum_{j \in V'(i)} x_{j,1} = 1,\ i \in V_3,\ x_{i,j} \geq 0,\ (i,j) \in E.$$

Задача о назначениях отличается от этой Л-задачи тем, что условия $x_{i,j} \geq 0$ заменены на условия $x_{i,j} = 0, 1\ ((i,j) \in E)$. Таким образом, любое оптимальное базисное решение рассматриваемой Л-задачи решает задачу о назначениях, поскольку оно целочисленное.

Задача о максимальном потоке. Эта задача получается из исходной, если положить $V_1 = \{l\}$, $V_3 = \{m\}$, $V'(l) = \varnothing$, $V(m) = \varnothing$, $a_1 = +\infty$, $b_m = 0$. Ограничения на дугах являются существенным фактором. Задача о максимальном потоке состоит в нахождении потока, для которого значение входного потока в вершине m является максимальным. Можно получить целевую функцию, положив $c_{i,m} = -1$, $i \in V'(m)$, и $c_{i,j} = 0$ — в остальных случаях. Итак, задача о максимальном потоке может быть записана следующим образом.

Максимизировать

$$\sum_{i \in V'(m)} x_{i,m}$$

при условиях

$$\sum_{j \in V(i)} x_{i,j} - \sum_{j \in V'(i)} x_{j,i} = 0,\ i \in V_2 = \{2, 3, \ldots, m-1\},\ 0 \leq x_{i,j} \leq d_{i,j},\ (i,j) \in E.$$

Задача о кратчайшем пути. Пусть $c_{i,j}$ интерпретируется как длина дуги (i,j). Определим длину пути в графе G как сумму длин всех дуг, входящих в этот путь. Требуется найти путь из вершины 1 в вершину m минимальной длины. Предполагается, что все контуры имеют неотрицательную длину. Эту задачу можно получить из исходной, если положить $V_1 = \{l\}$, $V_3 = \{m\}$, $a_1 = 1$, $b_m = 1$, $d_{i,j} = +\infty$, $(i,j) \in E$.

Ясно, что ограничения, соответствующие вершинам 1 и m графа G, будут удовлетворяться как равенства. Пропускные способности $d_{i,j} = 1$ учитываются неявно, так как условие $a_i = l$ и отсутствие отрицательных контуров влекут существование оптимального решения, в котором $x_{i,j} \leq 1$, $(i,j) \in E$. Последнее утверждение можно доказать, например, привлекая теорему о разложении потока на сумму потоков вдоль простых путей и простых контуров. При этом алгоритм разложения может быть

использован для построения самого решения [1]. Целочисленность потока даёт, что на всех дугах простого пути из вершины 1 в вершину m дуговой поток равен 1. Оптимальность исходного решения гарантирует, что это — кратчайший путь из вершины 1 в вершину m.

Абсолютная унимодулярность. Теперь покажем, что матрица ограничений $\begin{pmatrix} A \\ I \end{pmatrix}$ задачи о потоке минимальной стоимости является абсолютно унимодулярной. Ясно, что если A абсолютно унимодулярная, то и матрица $\begin{pmatrix} A \\ I \end{pmatrix}$ будет такой же. Ниже сформулируем и докажем достаточный признак абсолютной унимодулярности, из которого следует, что матрица инциденций A ориентированного графа G является абсолютно унимодулярной.

Целочисленная матрица A, каждый элемент $a_{i,j}$ которой равен либо -1, либо 0, либо $+1$ ($i = \overline{1,m};\ j = \overline{1,n}$), является абсолютно унимодулярной, если выполняются следующие условия:

а) каждый столбец матрицы A содержит не больше двух ненулевых элементов;

б) строки матрицы A можно разбить на подмножества T_1 и T_2 таким образом, что два ненулевых элемента противоположного (одинакового) знака из каждого столбца находятся в одном подмножестве (в различных подмножествах).

Доказательство проведем по индукции. Ясно, что всякая квадратная подматрица порядка 1 из A является абсолютно унимодулярной. Предположим, что утверждение справедливо для любой квадратной подматрицы, имеющей порядок $k-1$ или меньше. Пусть C — произвольная подматрица порядка k из A. Если матрица C содержит нулевой столбец, то $\det C = 0$. Если же в C находится хотя бы один единичный (минус-единичный) столбец, то разложим определитель $\det C$ по этому столбцу и применим индукционное предположение.

Наконец, рассмотрим случай, когда каждый столбец матрицы C содержит два ненулевых элемента. Для каждого столбца c_j из условий а), б) следует

$$\sum_{i\in T_1'} c_{i,j} = \sum_{i\in T_2'} c_{i,j} \quad (j = \overline{1,k};\ T_1' \subseteq T_1,\ T_2' \subseteq T_2).$$

Это даёт $\sum\limits_{i\in T_1'} r_i - \sum\limits_{i\in T_2'} r_i = 0$, $\det C = 0$ (r_i — i-я строка матрицы C).

Задача о потоке минимальной стоимости рассматривается, например, в [1, 58].

3.6 ПРЕОБРАЗОВАНИЕ ЗАДАЧИ О РАНЦЕ В ЗАДАЧУ О КРАТЧАЙШЕМ ПУТИ

Рассмотрим задачу о ранце

$$\max\Big\{z(x) = \sum_{j=1}^{n} c_j x_j \mid \sum_{j=-1}^{n} a_j x_j = b,\ x_j \geq 0 \wedge x_j \equiv 0,\ j = \overline{1,n}\Big\},$$

где данные задачи a_j, $j = \overline{1,n}$, и b — неотрицательные целые числа. Без ограничения общности будем считать, что $a_k \neq a_l$ для всех $k \neq l$, $a_j > 0$, $j = \overline{1,n}$ ($1 \leq k \leq n, 1 \leq l \leq n$).

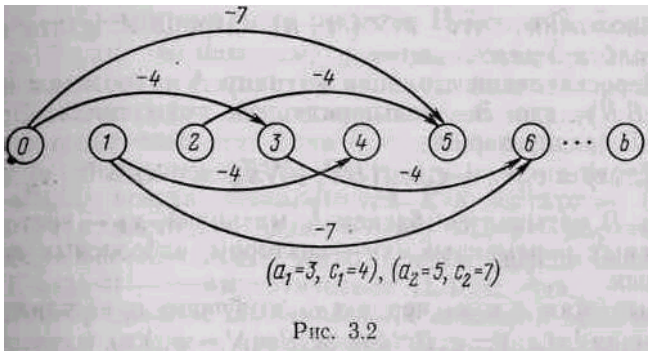


Рис. 3.2

Пусть $E_j = \{(i,k) \mid k - i = a_i,\ 0 \le i < k \le b\}$. Рассмотрим ориентированный граф $G = (V, E)$, где $V = \{0, 1, 2, \ldots, b\}$, $E = \bigcup_{j=1}^{n} E_j$. Пусть $-c_j$, будет длиной каждой дуги из E_j. Граф G не содержит контуров и имеет строение, показанное на рис. 3.2.

Каждый путь из вершины 0 в вершину b в графе G даёт допустимое решение задачи о ранце, если положить x_j равным числу дуг из E_j в этом пути.

Любое допустимое решение x задачи о ранце можно интерпретировать как путь из вершины 0 в вершину b, если каким-либо способом выбрать x_j дуг из E_j ($j = \overline{1,n}$). Длина такого пути есть $-z(x)$, так что кратчайший путь из вершины 0 в вершину b даёт оптимальное решение задачи о ранце.

Бинарная задача о ранце этим путём не может быть преобразована, так как необходимо дополнительно учитывать условия $x_j \le 1$, $j = \overline{1,n}$.

Преобразование в задачу о кратчайшем пути используется, например, при рассмотрении групповой задачи [58, 73].

3.7 ПРЕОБРАЗОВАНИЯ ЦЕЛОЧИСЛЕННОЙ ЛИНЕЙНОЙ ЗАДАЧИ НАД КОНУСОМ

Рассмотрим Л-задачу

$$\max\{z(x) = cx \mid Ax = b,\ x \ge 0\}.$$

Предположим, что $m \times (m+n)$-матрица A ранга m и m-столбец b целочисленные.

Перестановкой столбцов матрицу A приводим к виду $A = (BN)$, где B — невырожденная подматрица. Запишем Л-задачу в виде $\max\{z(x) = c_B x_B + c_N x_N \mid Bx_B + Nx_N = b,\ x_B \ge 0,\ x_N \ge 0\}$. Здесь B называется базисной матрицей, x_B — вектором базисных переменных, x_N — вектором небазисных переменных.

Выражая z и x_B через x_N, получаем представление

$$\max\{z(x_N) = c_B B^{-1} b - (c_B B^{-1} N - c_N) x_N \mid x_B = B^{-1} b - B^{-1} N x_N,\ x_B \ge 0,\ x_N \ge 0\}.$$

Решение $(x_B, x_N) = (B^{-1}b, 0)$ называется базисным решением Л-задачи. Если $B^{-1}b \ge 0$, то базисное решение называется прямо допустимым. Если же $c_B B^{-1} N - c_N \ge 0$, то базисное решение называется двойственно допустимым. Легко проверить, что прямо и двойственно допустимое базисное решение является оптимальным.

Теперь рассмотрим двойственно допустимое базисное решение и добавим условия целочисленности на переменные x. Это даёт ЦЛ-задачу

$$\max\{z(x) = c_B x_B + c_N x_N \mid Bx_B + Nx_N = b,\ x_B \ge 0,\ x_B \equiv 0,\ x_N \ge 0,\ x_N \equiv 0\}.$$

Исключая в этой задаче условия неотрицательности $x_B \ge 0$, получаем целочисленную линейную задачу над конусом (ЦЛК-задачу)

$$\min\{z'(x_N) = (c_B B^{-1} N - c_N) x_N \mid B^{-1} b - B^{-1} N x_N \equiv 0,\ x_N \ge 0,\ x_N \equiv 0\}.$$

Здесь исключены переменные x_B и константа $c_B B^{-1} b$, а также выполнен переход от задачи максимизации к задаче минимизации. Таким образом, множество допустимых решений ЦЛК-задачи представляет собой множество всех целочисленных точек из неотрицательного ортанта $0x_N$ (конуса), удовлетворяющих системе сравнений.

Важность ЦЛК-задачи состоит в том, что если x_N^* есть ее оптимальное решение и если

$$x_B^* = B^{-1} b - B^{-1} N x_N^* \ge 0,$$

то (x_B^*, x_N^*) — оптимальное решение ЦЛ-задачи. Если выбрать B так, что базисное решение является прямо и двойственно допустимым, то часто оптимальное решение ЦЛК-задачи будет удовлетворять исключенным условиям неотрицательности. Если базисное решение не является вырожденным ($B^{-1}b > 0$), то эти условия неотрицательности всегда выполняются для достаточно больших b', в

частности для $b' = \mu b$, где μ — достаточно большое положительное число. Отсюда другое название ЦЛК-задачи — "асимптотическая ЦЛ-задача".

Преобразование целочисленной линейной задачи над конусом в задачу о кратчайшем пути. Для выполнения дальнейших преобразований воспользуемся хорошо известным фактом о том, что для произвольной невырожденной целочисленной матрицы B существуют такие две унимодулярные целочисленные матрицы R и C, что $RBC = D$ есть нормальная диагональная матрица, т. е. диагональная матрица с диагональными элементами $d_{1,1}, d_{2,2}, \ldots, d_{m,m}$, удовлетворяющими условиям: $d_{i,i} > 0$, $i = \overline{1,m}$; $d_{i,i}$ является делителем $d_{i+1,i+1}$, $\prod_{i=1}^{m} d_{i,i} = |\det B| = d$.

Умножая $Bx_B + Nx_N = b$ на R и учитывая $RB = DC^{-1}$ получаем $DC^{-1}x_B + Rx_N = Rb$. Введя переменные $y, y = C^{-1}x_B$, последнее равенство можно записать в виде $Dy + RNx_N = Rb$. Так как C^{-1} — унимодулярная целочисленная матрица и переменные x_B не обязаны быть неотрицательными, вектор (y', x'_N) является целочисленным тогда и только тогда, когда вектор $(x'_B = Cy', x'_N)$ является целочисленным. Система уравнений $d_{j,i}y_i + r_iNx_N = r_ib$, $i = \overline{1,m}$, эквивалентна системе сравнений $r_iNx_N \equiv r_ib(\mathrm{mod}\, d_{i,i})$, $i = \overline{1,m}$ (r_i — i-я строка матрицы R). Сравнение $r_iNx_N \equiv r_ib(\mathrm{mod}\, d_{i,i})$ означает, что $r_iNx_N - r_ib$ кратно $d_{i,i}$.

Таким образом, ЦЛК-задача может быть записана в виде

$$\min\{z'(x_N) = (c_BB^{-1}N - c_N)x_N \mid r_iNx_N \equiv r_ib(\mathrm{mod}\, d_{i,i}),$$

$$i = \overline{1,m},\ x_N \geq 0,\ x_N \equiv 0\}.$$

Заметим, что для $d_{i,i} = 1$, $i = \overline{1,k}$, сравнения становятся тривиальными и могут быть исключены из рассмотрения ($k \leq m$).

Пусть $N = (a_1, a_2, \ldots, a_n)$($a_1$ — j-й столбец матрицы A). Рассмотрим ориентированный граф $G = (V, E)$, где $V = \{0, 1, 2, \ldots, d-1\}$. Вершина k графа G изображает столбец $p_k = (p_{1,k}, p_{2,k}, \ldots, p_{m,k})$, где $0 \leq p_{i,k} \leq d_{i,i} - 1$, $i = \overline{1,m}$ ($k = \overline{0, d-1}$). Пусть

$$E_i = \{(k,t) \mid r_ia_j + p_{i,k} \equiv p_{i,t}(\mathrm{mod}\, d_{i,i}),\ i = \overline{1,m};$$

$$0 \leq k \leq d-1, 0 \leq t \leq d-1\} \text{ и } E = \bigcup_{j=1}^{n} E_j.$$

Каждой дуге из E_j приписывается длина $c_BB^{-1}a_j - c_j$ ($j = \overline{1,n}$).

Все допустимые решения ЦЛК-задачи можно представить как пути из вершины 0, изображающей $(0, 0, \ldots, 0)$, в вершину, изображающую $(r_1b(\mathrm{mod}\, d_{1,1}), r_2b(\mathrm{mod}\, d_{2,2}), \ldots, r_mb(\mathrm{mod}\, d_{m,m}))$, и наоборот, используя соответствие $x_j \longleftrightarrow$ количество дуг из E_j ($j = \overline{1,n}$). Длина такого пути есть $(c_BB^{-1}N - c_N)x_N$. Таким образом, кратчайший путь даёт оптимальное решение ЦЛК-задачи.

Преобразование целочисленной линейной задачи над конусом в целочисленную линейную задачу над группой. Рассмотрим конечную абелеву группу $G(|G| = d)$, элементами g_k, которой являются неотрицательные целочисленные векторы $P_k = (p_{1,k}, p_{2,k}, \ldots, p_{m,k})$, где $0 \leq p_{i,k} \leq d_{i,i} - 1, i = \overline{1,m}$ ($k = \overline{0, d-1}$). Групповая операция состоит в покомпонентном сложении векторов по вектор-модулю $\delta = (d_{1,1}, d_{2,2}, \ldots, d_{m,m})$.

Каждому небазисному столбцу a_j поставим в соответствие элемент группы $g_{k_j} = Ra_j(\mathrm{mod}\, \delta)$ ($j = \overline{0,n}$; $a_0 = b$). Неотрицательный целочисленный вектор $x_N = (x_1, x_2, \ldots, x_n)$ удовлетворяет системе сравнений $RNx_N \equiv Ra_0(\mathrm{mod}\, \delta)$ тогда и только тогда, когда он удовлетворяет групповому уравнению

$$\sum_{j=1}^{n} x_jq_{k_j} = g_{k_0}.$$

Таким образом, ЦЛК-задача эквивалентна целочисленной линейной задаче над группой (ЦЛГ-задаче)

$$\min\{z'(x_N) = (c_BB^{-1}N - c_N)x_N \mid \sum_{j=1}^{n} x_jg_{k_j} = g_{k_0},\ x_N \geq 0,\ x_N \equiv 0\}.$$

Подобно задаче о ранце, ЦЛГ-задача содержит одно основное ограничение.

Отметим три случая, когда возможно уменьшение размеров задачи. Если $d_{i,i} = 1$, то из вектора можно исключить его нулевую компоненту с номером i. Не рассматриваются столбцы, образом которых является нулевой элемент группы G. Среди нескольких столбцов, отображающихся в один и тот же ненулевой элемент группы G, выбирается только один столбец с наименьшим коэффициентом целевой функции.

Можно получить ту же ЦЛГ-задачу (с точностью до изоморфизма), если к уравнению

$$Bx_B + Nx_N = b$$

применить отображение φB^{-1}, где φ отображает вектор в вектор дробных частей его компонент.

Эквивалентность целочисленных линейных задач над конусом. Для любой ЦЛК-задачи можно построить диагональную и эрмитову канонические задачи, сильно эквивалентные ей (см. § 3.8). Специальный вид этих задач облегчает их решение, а тем самым и решение исходной задачи.

Преобразования ЦЛК-задачи рассматриваются, например, в [53, 69].

3.8 ЭКВИВАЛЕНТНОСТЬ ЦЕЛОЧИСЛЕННЫХ ЛИНЕЙНЫХ ЗАДАЧ

Будет показано, что каждая ЦЛ-задача эквивалентна бесконечному числу других ЦЛ-задач. Эквивалентность такова, что оптимальное решение любой задачи из класса эквивалентности определяет оптимальное решение любой другой задачи из этого же класса. В каждом классе эквивалентности можно выбрать несколько задач (в эрмитовом виде, в диагональном виде и т. д.), специальная структура которых облегчает их решение. Вычислительная эффективность некоторых алгоритмов может существенно отличаться для различных задач из одного и того же класса эквивалентности. Для решения специальных классов целочисленных задач, таких как задача о ранце, задача о паросочетании, уже развиты эффективные алгоритмы. Понятие эквивалентности позволяет охарактеризовать класс задач, которые могут быть решены с помощью этих алгоритмов, а также установить связи между алгоритмами секущих плоскостей.

Хорошо известно, что общее целочисленное решение (множество всех целочисленных решений) совместной системы уравнений $Cx = b$, для которой $m \times n$-матрица C ранга m и m-столбец b целочисленные, имеет вид

$$X(C, b) = \{x \mid Cx = b,\ x \in Z_n\} = \{x \mid x = d + Ey,\ y \in Z_{n-m}\}.$$

Здесь d — некоторое частное целочисленное решение этой системы, а E — целочисленная $n \times (n-m)$ -матрица ранга $n-m$. Для любого целочисленного $(n-m)$-вектора h вектор $x = d+Eh$ также является частным решением рассматриваемой системы уравнений.

Пусть K — унимодулярная целочисленная $(n-m) \times (n-m)$-матрица. Тогда множество всех целочисленных решений системы уравнений может быть представлено как

$$X(C, b) = \{x \mid x = d + Eh + EKz,\ z \in Z_{n-m}\}.$$

Таким образом, выбирая различные h и K, получаем бесконечное множество представлений $X(C, b)$.

Система уравнений $Ax + Is = b$ с целочисленными переменными s, x, для которой $m \times n$-матрица A, единичная $m \times m$-матрица I и m-столбец b целочисленные, совместна и имеет ранг m. Общее целочисленное решение этой системы имеет вид

$$X((AI), b) = \left\{ \begin{pmatrix} s \\ x \end{pmatrix} \Big\| \begin{pmatrix} s \\ x \end{pmatrix} = \begin{pmatrix} b \\ 0 \end{pmatrix} + \begin{pmatrix} -A \\ I \end{pmatrix} y,\ y \in Z_n \right\},$$

а также вид

$$X((AI), b) = \left\{ \begin{pmatrix} s \\ x \end{pmatrix} \Big\| \begin{pmatrix} s \\ x \end{pmatrix} = \begin{pmatrix} (b - Ah) \\ h \end{pmatrix} + \begin{pmatrix} -AK \\ K \end{pmatrix} z,\ z \in Z_n \right\}.$$

Применим этот результат для построения класса эквивалентных ЦЛ-задач.

Рассмотрим ЦЛ-задачу в каноническом, стандартном и таккеровом видах:

$$\max\{cx \mid Ax \le b,\ x \ge 0,\ x \in Z_n\} \text{ (исходная задача)},$$

$$\max\{cx \mid Ax + Is = b,\ \begin{pmatrix} s \\ x \end{pmatrix} \ge 0,\ \begin{pmatrix} s \\ x \end{pmatrix} \in Z_{m+n}\},$$

$$\max\{0 + cy \mid s = b - Ay \ge 0,\ x = 0 + Iy \ge 0,\ y \in Z_n\}.$$

Здесь $m \times n$-матрица A, m-столбец b и n-строка c целочисленные. Допустимыми решениями ЦЛ-задачи в таккеровом виде являются только те целочисленные y, которые дают неотрицательные векторы из множества $X((AI), b)$. Поскольку все векторы из множества $X((AI), b)$, в том числе и неотрицательные, имеют и другое представление, ЦЛ-задачу в таккеровом виде можно записать таким образом:

$$\max\{ch + cKz \mid s = b - Ah - AKz \ge 0,\ x = h + Kz \ge 0,\ z \in Z_n\}.$$

Последняя задача называется задачей, эквивалентной исходной. Существует бесконечное число ЦЛ-задач, эквивалентных исходной, так как унимодулярная целочисленная матрица K, и целочисленный вектор h могут быть произвольными. Соотношения $x = h + Kz$, $z = K^{-1}(x - h)$ устанавливают взаимно однозначное соответствие между допустимыми решениями исходной и эквивалентной ей ЦЛ-задач, а также между допустимыми решениями соответствующих им Л-задач. Поэтому исходная задача сильно эквивалентна преобразованной.

Построение эквивалентной ЦЛ-задачи геометрически соответствует такому выбору начала координат и базиса в пространстве, что ЦЛ-задача остается ЦЛ-задачей.

Если ограничения ЦЛ-задачи заданы в виде системы уравнений $Ax = b$ с неотрицательными целочисленными переменными, то их можно преобразовать в систему неравенств $d + Ey \ge 0$ с целочисленными переменными. При этом целевая функция принимает вид $cd + cEy$. Здесь, как и раньше, выражение $x = d + Ey$ представляет общее целочисленное решение системы уравнений $Ax = b$.

Эрмитова каноническая задача. Пусть переменные s, x ЦЛ-задачи в стандартном виде разбиты на базисные переменные x_B и небазисные переменные x_N. В ЦЛ-задаче, записанной в таккеровом виде, выделим две группы ограничений, одна из которых соответствует небазисным переменным, а другая — базисным. Тогда ЦЛ-задачу можно представить в виде

$$\max\{0 + cy \mid x_N = b_1 + A_1 y \ge 0,\ x_B = b_2 + A_2 y \ge 0,\ y \in Z_n\},$$

где b_1 и b_2 — подвекторы вектора $\begin{pmatrix} b \\ 0 \end{pmatrix}$, A_1 и A_2 — подматрицы матрицы $\begin{pmatrix} -A \\ I \end{pmatrix}$. Такое представление ЦЛ-задачи называют таккеровым видом с выделенным базисом.

В каждом классе эквивалентности существует такая ЦЛ-задача, что матрица A_1K является эрмитовой, а столбец $b_1 + A_1h$ неотрицательный и имеет компоненты, строго меньшие соответствующих диагональных элементов матрицы A_1K. Такую ЦЛ-задачу, эквивалентную исходной, называют эрмитовой канонической задачей.

Покажем, как для исходной ЦЛ-задачи построить её эрмитову каноническую задачу. Квадратная матрица A_1 имеет ранг n и является целочисленной. Для данной матрицы A_1 существует такая унимодулярная целочисленная матрица K, что A_1K является эрмитовой, т. е. нижней треугольной матрицей, в которой все элементы главной диагонали положительные, а внедиагональные элементы неположительные и по модулю меньше соответствующих диагональных элементов. Таким образом, сначала для A_1 находим её эрмитову матрицу. Затем к столбцу b_1 прибавляем первый столбец матрицы A_1K, умноженный на целое так, чтобы преобразованный первый элемент столбца b_1 оказался неотрицательным и меньше первого диагонального элемента A_1K. Для получения требуемого второго элемента прибавляем второй столбец матрицы A_1K, умноженный на целое, и т. д.

В результате получаем эрмитову каноническую задачу

$$\max\{ch + cKz \mid x_N = b_1 + A_1h + A_1Kz \ge 0,\ x_B = b_2 + A_2h + A_2Kz \ge 0,\ z \in Z_n\}.$$

Если элемент $d_{i,i}$ главной диагонали эрмитовой матрицы A_1K равен 1, то соответствующее ему неравенство является тривиальным ограничением $z_i \ge 0$.

Число нетривиальных ограничений в эрмитовой канонической задаче, например, для ЦЛК-задачи не превышает числа неединичных элементов главной диагонали матрицы A_1K. Заметим, что это число зависит от порядка строк матрицы A_1.

Диагональная каноническая задача. При рассмотрении этой задачи матрица A_1 приводится к диагональному виду, например, при помощи выполнения элементарных строковых и столбцовых преобразований. Сначала, введя слабые переменные $t = (t_1, t_2, \ldots, t_n)$, превращаем неравенства, соответствующие небазисным переменным в равенства $b_1 - It + A_1y = 0$, $t \geq 0$, $t \in Z_n$.

Для невырожденной целочисленной матрицы A_1 существуют такие две унимодулярные целочисленные матрицы K_1 и K_2, что $K_1AK_2 = D$ является диагональной матрицей. Учитывая это, можно записать задачу

$$\max\{cK_2z \mid K_1b - K_1t + K_1A_1K_2z = 0,\ x_B = b_2 + A_2K_2z \geq 0,\ t \geq 0,\ t \in Z_n,\ z \in Z_n\}.$$

Замена переменных

$$\begin{pmatrix} t \\ z \end{pmatrix} = \begin{pmatrix} I_n & 0 \\ K & I_n \end{pmatrix} \begin{pmatrix} t \\ u \end{pmatrix} + \begin{pmatrix} 0 \\ h \end{pmatrix}$$

выражает последовательность выполненных элементарных столбцовых преобразований и приводит к диагональной канонической задаче

$$\max\{cK_2h + cK_2Kt + cK_2u \mid K_1b_1 + Dh + (-K_1 + DK)t + Du = 0,$$

$$x_B = b_2 + A_2K_2h + A_2K_2Kt + A_2K_2u \geq 0,\ t \geq 0,\ t \in Z_n,\ u \in Z_n\}.$$

Здесь целочисленный n-вектор h выбирается из условия, что каждый элемент i столбца $K_1b_1 + Dh$ является неотрицательным и меньше, чем $|d_{i,i}|$, а целочисленная $n \times n$-матрица K определяется тем, что каждый элемент i-й строки матрицы $-K_1 + DK$ неположительный и по модулю меньше, чем $|d_{i,i}|$ $(i = \overline{1,n})$.

Если $|d_{i,i}| = 1$, то $u_i = 0$. Такую переменную u_i и соответствующее ей равенство можно исключить из рассмотрения. Заметим, что в нормальной диагональной Матрице находится наибольшее количество единичных Диагональных элементов. Поэтому для такой матрицы в преобразованной задаче будет наименьшее количество переменных.

Существуют и другие сильно эквивалентные преобразования ЦЛ-задачи. В частности, одно из них было использовано в преобразовании ЦЛК-задачи ($D = RBC$; см. § 3.7).

Сильно эквивалентные преобразования ЦЛ задачи рассматриваются, например, в [53, 69].

3.9 ПРЕОБРАЗОВАНИЕ СМЕШАННОЙ ЦЕЛОЧИСЛЕННОЙ ЛИНЕЙНОЙ ЗАДАЧИ В ЦЕЛОЧИСЛЕННУЮ

Подставляя в СЦЛ-задачу

$$\min\{cx + dy \mid Ax + By \geq b,\ x \geq 0,\ x \equiv 0,\ y \geq 0\}$$

любой неотрицательный целочисленный вектор x', получаем Л-задачу

$$\min\{cx' + dy \mid By \geq b - Ax',\ y \geq 0\}.$$

Двойственная к ней задача имеет вид

$$\max\{cx' + u(b - Ax') \mid uB \leq d,\ u \geq 0\}.$$

Множество допустимых решений двойственной задач $P = \{u \mid uB \leq d,\ u \geq 0\}$ не зависит от x'.

Пусть $\{u_t \mid t \in T\}$ — совокупность всех крайних точек множества P ($T = \{1, 2, \ldots, |T|\}$), $\{z_q \mid q \in Q\}$ — совокупность всех направлений крайних лучей множества P ($Q = \{1, 2, \ldots, |Q|\}$) (могут быть параллельные лучи, т. е. лучи, имеющие одно и то же направление). Заметим, что если множество P пусто,

то СЦЛ-задача не имеет оптимального решения, так как для фиксированного x' либо ограничения прямой задачи несовместны либо значения её целевой функции не ограничены снизу на множестве допустимых решений.

Исходную СЦЛ-задачу можно преобразовать в задачу, в которой используются элементы множеств $\{u_t \mid t \in T\}$ и $\{z_q \mid q \in Q\}$. Для этого выполним элементарные преобразования [см. упражнения 2.1 б) и д)]:

$$\min\{cx + dy \mid Ax + By \geq b,\ x \geq 0,\ x \equiv 0,\ y \geq 0\} =$$

$$\min_x\{cx + \min\{dy \mid By \geq b - Ax,\ y \geq 0\}\} =$$

$$= \min_x\{cx + \max\{u(b - Ax) \mid uB \leq d,\ u \geq 0\}\} =$$

$$\to \min_x\{cx + \max_{t\in T}\{u_t(b - Ax) \mid z_q(b - Ax) \leq 0,\ q \in Q\}\} =$$

$$= \min\{cx + \max_{t\in T}(u_t(b - Ax)) \mid z_q(b - Ax) \leq 0,\ q \in Q,\ x \geq 0,\ x \equiv 0\} =$$

$$= \min\{w \mid w \geq cx + \max_{t\in T}(u_t(b - Ax)),\ z_q(b - Ax) \leq 0,\ q \in Q,\ x \geq 0,\ x \equiv 0\} =$$

$$= \min\{w \mid w \geq cx + u_t(b - Ax),\ t \in T,\ z_q(b - Ax) \leq 0,\ q \in Q,\ x \geq 0,\ x \equiv 0\}.$$

Последняя задача представляет собой СЦЛ-задачу одной непрерывной переменной w. В отличие от исходной задачи её часто называют целочисленной. Любой метод решения целочисленной задачи может быть незначительно модифицирован для её решения. Преобразованная ванная задача, вообще говоря, может иметь огромное количество неравенств. Однако эту задачу можно решать итеративно, строя линейные неравенства лишь тогда, когда они необходимы. Для построения неравенств используется линейная задача, определяемая матрицей B. В частности, это преобразование может быть полезным, когда структура матрицы B облегчает решение линейкой задачи (прямой или двойственной), дающей неравенства.

Изложенная процедура разбиения рассматривается, например, в [53, 58].

3.10 ПРЕОБРАЗОВАНИЕ ЦЕЛОЧИСЛЕННОЙ ЛИНЕЙНОЙ ЗАДАЧИ В ЛИНЕЙНУЮ

Рассмотрим ЦЛ-задачу $\max\{cx \mid Ax = b,\ x \geq 0,\ x \equiv 0\}$, для которой множество допустимых решений является конечным. Эта ЦЛ-задача в принципе может быть преобразована усечением множества $X' = \{x \mid Ax = b,\ x \geq 0\}$ линейными ограничениями в такую Л-задачу, что её система ограничений задаёт выпуклую оболочку допустимых решений ЦЛ-задачи. Если матрица ограничений A ЦЛ-задачи абсолютно унимодулярная, то не требуется никакого усечения, так как очевидно, что каждая крайняя точка множества X' является целочисленной для всех целочисленных векторов. Если $X' = \{x \mid Ax \leq b,\ x \geq 0\}$, то верно и обратное: если все крайние точки множества X' являются целочисленными для всех целочисленных векторов b, то матрица A абсолютно унимодулярная (см., например, [20]).

Точное представление выпуклой оболочки всех допустимых решений ЦЛ-задачи может быть получено в очень немногих случаях. Один из наиболее значительных случаев есть задача о рёберной упаковке (о паросочетании), в которой выпуклая оболочка допустимых решений задаётся неравенствами

$$Ax \leq e,\ \sum_{e_j \in E(S)} x_j \leq \frac{|S| - 1}{2}\ \forall S \subseteq V,\ x \geq 0,$$

где $|S|$ — нечётное число, $|S| \geq 3, A$ — вершинно-рёберная матрица инциденций неориентированного графа $G, E(S)$ — подмножество рёбер, инцидентных двум вершинам из S (см., например, [26]).

Можно охарактеризовать вершины и грани многогранника ЦЛК-задачи. Эта характеризация основана на том, что небазисным столбцам ЦЛК-задачи соответствуют элементы конечной абелевой группы, которая является прямой суммой циклических групп порядков $d_{i,i}$, где $d_{i,i}$, $i = \overline{1, m}$ — диагональные элементы нормальной матрицы D базисной матрицы B (матрицы, связанной с B). Грани многогранника, называемого угловым, могут быть получены из прямо допустимого базисного решения Л-задачи.

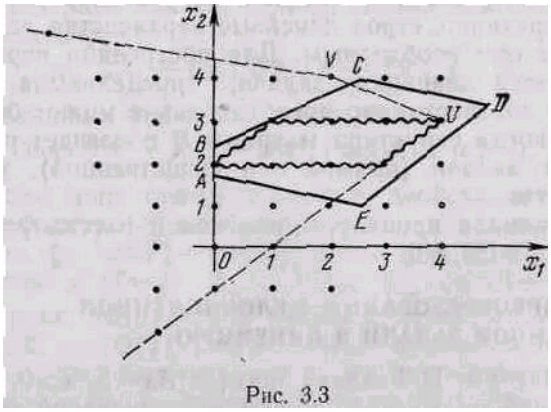


Рис. 3.3

На рис. 3.3 изображен двумерный случай. Допустимые решения Л-задачи есть точки, находящиеся внутри и на границе многоугольника ABCDE. Выпуклая оболочка допустимых решений ЦЛ-задачи изображается многоугольником, ограниченным волнистой линией. Угловой многогранник, определяемый крайней точкой D, есть неограниченная область между штриховыми линиями, и имеет крайние точки U, V. Очевидно, что угловой многогранник содержит выпуклую оболочку допустимых решений ЦЛ-задачи.

3.11 ЛИНЕАРИЗАЦИЯ НЕЛИНЕЙНОЙ ЗАДАЧИ

Заменяя каждую целочисленную переменную системой бинарных переменных, любую Ц-задачу с ограниченными переменными

$$\max\{f(x) \mid g_i(x) \le 0,\ i = \overline{1,m},\ 0 \le x \le d,\ x \in Z_n\}$$

можно преобразовать в сильно эквивалентную ей Б-задачу (см. § 2.1).

Преобразование бинарной задачи в бинарную полиномиальную задачу. Всякая действительнозначная функция $f(x_1, x_2, \ldots, x_n)$ от n бинарных переменных $x_1, x_2, \ldots, x_n$ является линейной относительно каждого из своих аргументов. Используя это свойство, такую функцию можно представить как полином (многочлен) от n бинарных переменных с действительными коэффициентами (с точностью до порядка слагаемых и сомножителей).

Действительно, для каждого j имеем

$$f(x_1, \ldots, x_j, \ldots, x_n) = x_j g_j(x_1, \ldots, x_{j-1}, x_{j+1}, \ldots, x_n) +$$

$$+ h_j(x_1, \ldots, x_{j-1}, x_{j+1}, \ldots, x_n),$$

где

$$g_j(x_1, \ldots, x_{j-1}, x_{j+1}, \ldots, x_n) = f(x_1, \ldots, x_{j-1}, 1, x_{j+1}, \ldots, x_n) - f(x_1, \ldots, x_{j-1}, 0, x_{j+1}, \ldots, x_n),$$

$$h_j(x_1, \ldots, x_{j-1}, x_{j+1}, \ldots, x_n) = f(x_1, \ldots, x_{j-1}, 0, x_{j+1}, \ldots, x_n)\ (j = \overline{1,n}).$$

Повторив n раз это преобразование, придем к многочлену от переменных $x_1, x_2, \ldots, x_n$ коэффициентами которого являются суммы произведений значений функции $f(x_1, x_2, \ldots, x_n)$. Единственность многочлена может быть доказана индукцией по числу переменных.

Легко показать, что всякую функцию $f(x_1, x_2, \ldots, x_n)$ от бинарных переменных можно представить также и в виде $f(x_1, x_2, \ldots, x_n) = \sum\limits_{\alpha \in B_n} c(\alpha) x_1^{\alpha_1} x_2^{\alpha_2} \ldots x_n^{\alpha_n}$ (каноническое задание). Здесь B_n — множество всех 2^n бинарных n-векторов, $\alpha = (\alpha_1, \alpha_2, \ldots, \alpha_n)$ — бинарный вектор,

$$x_j^1 = x_j,\ x_j^0 = \overline{x_j} = 1 - x_j\ (j = \overline{1,n}).$$

Пример 3.2. Теперь покажем, как можно преобразовать задачу

$$\max\{z=\prod_{j=1}^{n}(1-p_j^{x_j+1}) \mid A(x+e)\le b,\ 0\le x\le d,\ x\equiv 0\},$$

возникающую в приложениях при исследовании надежности систем, в БП-задачу. Ограничимся рассмотрением лишь целевой функции.

Величины t_j определяются из условий $2^{t_j}\le d_j<2^{t_j+1}(j=\overline{1,n})$. Над целевой функцией выполняем преобразования:

$$z=\prod_{j=1}^{n}(1-p_j^{x_j+1})=\prod_{j=1}^{n}(1-p_jp_j^{x_j})=\prod_{j=1}^{n}\left(1-p_jp_j^{\sum\limits_{r=0}^{t_j}2^rx_{j,r}}\right)=$$

$$=\prod_{j=1}^{n}\left(1-p_j\prod_{r=0}^{t_j}p_{j,r}^{x_{j,r}}\right),$$

где $p_{j,r}=p_j^{(2^r)}, x_{j,r}=0,1(j=\overline{1,n}, r=\overline{0,t_j})$. Из линейности следует

$$p_{j,r}^{x_{j,r}}=x_{j,r}(p_{j,r}-1)+1=1-(1-p_{j,r})x_{j,r}.$$

Тогда получаем

$$z=\prod_{j=1}^{n}(1-p_j\prod_{r=0}^{t_j}(1-(1-p_{j,r})x_{j,r})).$$

Преобразование бинарной полиномиальной задачи в бинарную линейную задачу. Это преобразование осуществляется путём линейного представления всех входящих в задачу полиномов от бинарных переменных (см. §2.1).

Если с наперед заданной точностью каждую непрерывную переменную заменить системой бинарных переменных (см. § 2.1), то преобразования, рассмотренные в этом параграфе, позволяют представить нелинейную задачу оптимизации с ограниченными переменными

$$\max\{f(x) \mid g_i(x)\le 0,\ i=\overline{1,m},\ 0\le x\le d, x\in R_n\}$$

как бинарную линейную задачу.

Изложенные в этом параграфе преобразования рассматриваются, например, в [53].

УПРАЖНЕНИЯ

3.1. Л-задача $\max\{cx \mid Ax=b, x\ge 0\}$ с рациональными данными A, b, c может быть преобразована в Л-задачу с целочисленными данными таким образом, что множества допустимых (оптимальных) решений исходной н преобразованной задач совпадают. Верно ли это для действительных данных?

3.2. Преобразование Л-задачи с переменными произвольного знака в Л-задачу с неотрицательными переменными. Пусть x^* — оптимальное решение задачи $\max\{cx \mid Ax\le b\}$. Тогда вектор (y_0^*, y_0) с компонентами

$$y_0^*=\max_{j=1}^{n}|x_j^*|,\ y_j^*=x_j^*+y_0^*,\ j=\overline{1,n},$$

является оптимальным решением задачи

$$\max\{cy-c_0y_0 \mid Ay-a_0y_0\le b,\ y_0\ge 0,\ y\ge 0\},$$

и наоборот, где $c_0=\sum\limits_{j=1}^{n}c_j$, $a_0=\sum\limits_{j=1}^{n}a_j$ (a_j — j-й столбец матрицы A).

Привести пример, показывающий, что это преобразование задачи оптимизации не является сильно эквивалентным.

3.3. Преобразовать нелинейную задачу оптимизации

$$\min\Big\{\sum_{i=1}^{m}|z_i| \mid \sum_{j=1}^{n}a_{i,j}x_j + z_i = b_i,\ i=\overline{1,m},\ x_j \geq 0,\ j=\overline{1,n}\Big\}$$

в линейную.

3.4. Рассмотрим две задачи $\max\limits_{x\in X} f_1(x)$ и $\max\limits_{x\in X} f_2(x)$, где $f_2(x) \geq f_1(x)$, $x \in X$. Записать эти задачи в таком виде, который показывает, что вторая задача является расширением первой.

3.5. Преобразовать БЛ-задачу, в которой, по крайней мере, один коэффициент c_j целевой функции отрицательный, в БЛ-задачу с $c'_j \geq 0$ для всех j. Показать, что это преобразование является сильно эквивалентным.

3.6. Преобразовать БЛ-задачу $\max\{cx \mid Ax \leq b,\ x \in B_n\}$, где матрица A и столбец b неположительные ($Ae \leq b$), в БЛ-задачу, ограничения которой $A'y \leq b'$, $y \in B_n$ содержат уже неотрицательные данные A' и b'. Показать, что это преобразование сильно эквивалентно.

3.7. Преобразовать задачу вида $\max\{cx \mid A^1x \leq b,\ A^2x = b^2,\ x \in B_n\}$ в задачу вида $\min\{dy \mid Ay \geq b,\ y \in B_n\},\ d \geq 0$.

3.8. Рассмотрим бинарную задачу о ранце $\max\{cx \mid Ax \leq b,\ x \in B_{n'}\}$, т. е. $m' \times n'$-матрица A и m'-столбец b неотрицательные.

Преобразовать эту задачу в эквивалентную ей бинарную задачу о ранце, данные которой удовлетворяют следующим условиям:

$$a_{i,j} \leq b_i,\ i \in M,\ j \in N;\ b_i > 0,\ i \in M;$$

$$c_j > 0,\ j \in N;\ \sum_{j=1}^{n} a_{i,j} > b_i,\ i \in M$$

$$(M \subset M' = \{1,2,\ldots,m'\};\quad N \subset N' = \{1,2,\ldots,n'\})$$

Какие из преобразований A–E, рассмотренных в § 3.1, при этом используются?

3.9. Пусть $S = \{x \mid 0 \leq x_1 \leq 5,\ 0 \leq x_2 \leq 3,\ 0 \leq x_3 \leq 3,\ x \equiv 0\}$. Показать, что на множестве S система уравнений $x_1 - x_2 + x_3 = 3$, $x_1 + 2x_3 = 6$ эквивалентна уравнению $4x_1 - 7x_2 + x_3 = 3$.

3.10. Пусть матрица

$$A = \begin{pmatrix} 1 & 1 & 1 & 0 & 1 & 0 & 1 & 1 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 1 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 \end{pmatrix}$$

и вектор $c = (10, 5, 8, 6, 9, 13, 11, 4, 6)$ служат данными для П-и Р-задач.

Выполнив исключение строк и столбцов, показать, что а) покрытие $J^* = \{489\}$ является оптимальным, б) Р-задача не имеет допустимых решений.

3.11. Доказать справедливость следующего правила исключения для Р-задачи. Пусть для строки r_k и r_t матрицы A не выполняет ни одно из условий $r_k \geq r_t$, $r_t \geq r_k$, а $K = \{j \mid a_{k,j} \geq a_{t,j}\}$, $T = \{j \mid a_{t,j} \geq a_{k,j}\}$. Если существует строка r_s с $a_{s,j} = 1$ для всех $j \in K$ и $a_{s,j} = 1$ для некоторого $j' \in T$, то столбец a_j может быть исключен.

Использовав это правило, исключить столбцы a_2 и a_5 из матрицы

$$A = \begin{pmatrix} 0 & 0 & 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 1 & 1 & 0 \\ 0 & 1 & 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 & 1 & 1 \end{pmatrix}.$$

3.12. Было показано что для каждой Р-задачи имеющей допустимое решение существует эквивалентная ей П-задача. Верно ли обратное?

3.13. Показать, что предположение $c_j > 0$ для всех $j \in J$ не нарушает общности:

а) для П-задачи минимизации существует оптимальное покрытие, для которого $x_j = 1$ для всех $c_j \leq 0$;

б) Р-задача эквивалентна другой Р-задаче со стоимостями $c'_j = c_j + d_j$ для всех j, где $d_j = \sum\limits_{i=1}^{m} k_i a_{j,i}$, а k'_i, $i = \overline{1,m}$, — произвольные действительные числа.

3.14. Было сформулировано необходимое и достаточное условие существования оптимального решения для П-задачи. Применимо это условие для Р-задачи?

3.15. Преобразовать Р-задачу минимизации в П-задачу минимизации путём изменения целевой функции (см. § 3.3).

3.16. Является ли матрица

$$A = \begin{pmatrix} 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 \end{pmatrix}$$

абсолютно унимодулярной?

3.17. Показать, что матрица инциденций двудольного неориентированного графа является абсолютно унимодулярной.

3.18. Доказать, что следующие утверждения эквивалентны:

а) матрица A абсолютно унимодулярная;

б) матрица, транспонированная к A абсолютно унимодулярная;

в) матрица (AA) абсолютно унимодулярная.

3.19. Рассмотрим матрицу a с элементами $-1, 0, +1$, каждый столбец которой содержит не больше двух ненулевых элементов. Если матрица A абсолютно унимодулярная, то её строки можно разбить на подмножества T_1 и T_2 — таким образом что два ненулевых элемента противоположного (одинакового) знака из каждого столбца находятся в одном подмножестве (в различных подмножествах). Привести пример, показывающий, что для произвольной матрицы A с элементами $-1, 0, +1$ это не так.

3.20. Для каждой из групп $G(8), G(2,4), G(2,2,2)$ построить таблицу, задающую в ней групповую операцию. Использовав эти таблицы, показать, что ни одна из трёх рассматриваемых групп не изоморфна другой.

3.21. Доказать, что различные смежные классы конечной абелевой группы по подгруппе $\{g\}$, порожденной элементом g дают разбиение всех элементов группы G на $|G|/|\{g\}|$ подмножеств.

3.22. Рассмотрим ЦЛ-задачу максимизировать: $2x_1 + x_2$ при условиях $x_1 + x_2 \leq 5$, $-x_1 + x_2 \leq 0$, $6x_1 + 2x_2 \leq 21$, $x_j \geq 0$, $x_j \equiv 0$ $(j = 1, 2)$.

А. Записать эту ЦЛ-задачу в таккеровом виде с выделенным базисом $B, x_B = (s_2, x_1, x_2)$.

Б. Для рассматриваемой ЦЛ-задачи построить эквивалентную ей эрмитову каноническую задачу.

3.23. Преобразовать СЦЛ-задачу $\max\{cx + dy \mid Ax + By \leq b,\ x \geq 0,\ x \equiv 0,\ y \geq 0\}$ в целочисленную (см § 3.9).

3.24. Задать функцию $f(x_1, x_2, x_3) = 2x_1\overline{x_2} + 6x_1x_3 - 5\overline{x_2x_3}$ в табличном, полиномиальном и каноническом видах, где $x_j = 0, 1, j = 1, 2, 3$.

Решение некоторых упражнений

3.9. Пусть $r(x) = 3 - x_1 + x_2 - x_3$, $s(x) = 6 - x_1 - 2x_3$. Тогда $\inf\limits_{x \in S} r(x) = -2$, $\sup\limits_{x \in S} s(x) = 6$. Для множителей $\alpha = 7$ и $\beta = -3$ выполняются, например, неравенства $\alpha > s_3 = 6$ и $\beta < i_1 = 0$.

3.10. А. Возможная последовательность исключение: 3) $t = 7, p = 4$; 2) $r_4 = e_4$; 3) $t = 1, p = 2$; 4а) $S = \{2\}, k = 7$; 4а) $S = \{3\}, k = 1$; 4а) $S = \{5\}, k = 6$; 4а) $S = \{9\}, k = 5$; 3) $t = 6, p = 5$; 4) $S = \{8\}, k = 2$.

Б. Возможная последовательность исключений: 3) $t = 7, p = 4$; 3а) $k = 1, 5, 8, 9$; 2) $r_4 = e_4$; 3) $t = 1, p = 2$; 3а) $k = 7$.

3.11. Для численного примера $k = 1$, $s = 2$, $t = 3$.

3.16. Да. Разбиение строк $T_1 = \{1, 4, 5\}$ и $T_2 = \{2, 3, 6\}$ удовлетворяют достаточному признаку абсолютной унимодулярности (см. § 3.5).

3.22. Б. Искомыми являются матрицы

$$\underset{A_1}{\begin{pmatrix} -1 & -1 \\ -6 & -2 \end{pmatrix}} \times \underset{K}{\begin{pmatrix} 1 & -1 \\ -2 & 1 \end{pmatrix}} = \underset{T}{\begin{pmatrix} 1 & 0 \\ -2 & 4 \end{pmatrix}}.$$

Глава 4

МЕТОДЫ РАСПАРАЛЛЕЛИВАНИЯ ВЫЧИСЛЕНИЙ

Вопросам, связанным с распараллеливанием вычислений на конкретных вычислительных системах, посвящено много работ (например, [37, 75]). В данной главе, отвлекаясь от архитектуры таких систем, рассмотрим основные приемы построения параллельных вычислительных методов. При этом будем считать, что каждый процессор параллельной вычислительной системы обладает неограниченной памятью, может работать независимо от других процессоров и выполнять все необходимые операции, а между процессорами существуют связи. Следует также помнить, что при построении параллельных вычислений важно использовать возможность одновременной подачи многих данных, сопровождаемых одной (режим SIMD) или многими инструкциями (режим MIMD).

В последние годы большая работа велась по автоматическому расчленению последовательных алгоритмов, точнее, процедур или подпрограмм, написанных на алгоритмических языках Алгол и Фортран [74]. Это важное направление исследований здесь не представлено. Не рассматриваются также результаты несколько пессимистического характера. Например, в некоторых случаях за счёт увеличения числа операций оценка ошибок округления при параллельных вычислениях может быть хуже, чем при последовательных. Основное внимание уделяется тем преимуществам, которые обусловлены параллельными вычислениями.

4.1 ПРИНЦИП РАСПАРАЛЛЕЛИВАНИЯ ДАННОГО ПОСЛЕДОВАТЕЛЬНОГО АЛГОРИТМА

Рассмотрим конечную вычислительную задачу, т. е. задачу, в которой по нескольким данным числам требуется найти числа, составляющие ответ. Для простоты будем считать, что эта задача решается прямым (неитерационным) алгоритмом, не содержащим условных переходов. Отдельные операции, из которых состоит алгоритм, являются бинарными или унарными.

Разобьем все числа, появляющиеся при выполнении последовательного алгоритма, на уровни следующим образом. К нулевому уровню отнесем числа, являющиеся входными данными задачи, а к i-му уровню, $i \geq l$, — числа, получающиеся в результате выполнения какой-либо операции, одна из компонент которой принадлежит $(i-1)$-му уровню, а вторая компонента для бинарной операции находится на том же самом уровне или на уровне с меньшим номером.

Номер последнего уровня называется высотой алгоритма. Число результатов i-го уровня называется его шириной, а максимальная ширина всех уровней, начиная с первого, — шириной алгоритма.

Пусть требуется вычислить значение выражения $b = ((a_1 a_2) + (a_3 a_4))((a_5 a_6) + (a_7 a_8))$ для заданных значений $a_1, a_2, \ldots, a_8$. Эту вычислительную задачу можно решать алгоритмом, который представляет собой естественную последовательность действий, заданную расстановкой скобок в данном выражении (сначала вычисляется первая компонента бинарной операции, а затем — вторая). Высота такого алгоритма равна 3, а ширина — 4 (рис. 4.1). Знак умножения опускается в записи рассматриваемого выражения, а также в схемах на рис. 4.1—4.3, 4.6—4.8.

Если число процессоров, действующих одновременно и независимо, не меньше ширины алгоритма, и если каждый процессор выполняет любую из операций за один такт (единицу времени), то каждый переход к следующему уровню может быть осуществлен за один такт работы параллельной вычислительной системы, а число тактов равно высоте алгоритма. В частности, это имеет место для схемы на рис. 4.1, если число процессоров не меньше четырёх.

Уровень	$b = ((a_1a_2) + (a_3a_4))((a_5a_6) + (a_7a_8))$
0	$a_1 \quad a_2 \quad a_3 \quad a_4 \quad a_5 \quad a_6 \quad a_7 \quad a_8$
1	$a_1a_2 \quad a_3a_4 \quad a_5a_6 \quad a_7a_8$
2	$(a_1a_2) + (a_3a_3) \quad (a_5a_6) + (a_7a_8)$
3	$((a_1a_2) + (a_3a_4))((a_5a_6) + (a_7a_8))$
	Высота схемы 3, ширина 4

Рис. 4.1

Если же число процессоров меньше ширины алгоритма, то получение части чисел некоторых уровней приходится откладывать на следующие такты. Располагая операции по уровням согласно времени их выполнения, получаем схему, ширина которой не превышает числа процессоров, и на i-м уровне находятся результаты операций над результатами предшествующих уровней без требования, чтобы одна из компонент операции была точно на $(i-1)$-м уровне (см., например, схему на рис. 4.2).

Построение такой схемы, вообще говоря, неоднозначно, и от выбора схемы может зависеть её высота, которую будем называть p-высотой, подчёркивая зависимость от числа процессоров p (например, рис. 4.2 и 4.3). Наивыгоднейший выбор следования операций (p-высота минимальна) есть, по существу, частный случай задачи о наилучшем расписании очередности выполнения работ. Её решение даётся известным алгоритмом Ху.

На некоторых параллельных вычислительных системах одновременно действующие процессоры могут выполнять только одну и ту же инструкцию (система SIMD), и при распараллеливании алгоритма с этим приходится считаться.

Пусть t_1 — число тактов при выполнении алгоритма на одном процессоре, а t_p — число тактов при выполнении того же самого алгоритма на p процессорах. Ясно, что t_1 равно количеству операций, входящих в алгоритм, и всегда $t_1 \leq pt_p$.

Уровень	$b = ((a_1a_2) + (a_3a_4))((a_5a_6) + (a_7a_8))$
0	$a_1 \quad a_2 \quad a_3 \quad a_4 \quad a_5 \quad a_6 \quad a_7 \quad a_8$
1	$a_1a_2 \quad a_3a_4$
2	$a_5a_5 \quad a_7a_8$
3	$(a_1a_2) + (a_3a_4) \quad (a_5a_6) + (a_7a_8)$
4	$((a_1a_2) + (a_3a_4))((a_5a_6) + (a_7a_8))$
	2 высота схемы 4

Рис. 4.2

Уровень	$b = ((a_1a_2) + (a_3a_4))((a_5a_6) + (a_7a_8))$
0	$a_1 \quad a_2 \quad a_3 \quad a_4 \quad a_5 \quad a_6 \quad a_7 \quad a_8$
1	$a_1a_2 \quad a_3a_4$
2	$(a_1a_2) + (a_3a_4) \quad a_5a_6$
3	a_7a_8
4	$(a_5a_6) + (a_7a_8)$
5	$((a_1a_2) + (a_3a_4))((a_5a_6) + (a_7a_8))$
	2 высота схемы 5

Рис. 4.3

Число $S_p = t_1/t_p$, характеризующее ускорение при переходе от последовательного выполнения алгоритма к параллельному на p процессорах, не превышает числа процессоров и может ему равняться, если на всех тактах заняты все процессоры. Величина $e_p = s_p/p$ называется эффективностью.

Обозначим через $\hat{t}_1$ наименьшее количество операций, нужных для вычисления результата алгоритма. Ясно, что $\hat{t}_1 \le t_1$. Некоторые авторы за меру ускорения при выбранном алгоритме на p процессорах принимают отношение $\hat{s}_p \le \hat{t}_1/t_p \le t_1/t_p \le p$ и за меру эффективности — величину $\hat{e}_p = \hat{s}_p/p$.

Рассматриваются и некоторые другие характеристики, в частности цена алгоритма $c_p = pt_p$,и ценность $f_p = s_p/C_p = e_p/t_p = t_1/pt_p^2$.

Для однотипных задач разных размеров величины t_1 и t_p зависят от параметра n (скажем, от количества сомножителей в произведении, размерности вектора, порядка матрицы), значение которого определяет размер задачи. При учёте числа тактов алгоритма (вычислительной схемы) целесообразно указывать зависимость этих величин от n. В этом случае, например, ускорение параллельной схемы принимает вид $S_p(n) = t_1(n)/t_p(n)$. Считается, что распараллеливание построено удачно, если ускорение пропорционально n с коэффициентом, не зависящим от размера задачи.

Пример 4.1. Пусть алгоритм состоит в вычислении значения арифметического выражения, связывающего n букв, для некоторого набора значений этих букв. При этом порядок действий строго установлен, например, посредством расстановки скобок, в каждой паре из которых записана одна из бинарных операций $+, -, \times, /$ над значениями, вычисленными раньше. Такому алгоритму можно естественным образом сопоставить ориентированный граф, вершинами которого являются входные данные и числа, получающиеся в результате выполнения действий, а ориентация дуг определяется порядком действий.

Для выражения $b = (((a_1 \times a_2) \times a_3) + a_4) \times a_5$, соответствующий ориентированный граф приведен на схеме рис. 4.4. Знаки операций в кружочке, используемые в схемах на рис. 4.4 и 4.5, а также в схемах на рис. 4.9 и 4.10, означают результаты выполнения этих операций (например, $\otimes$ есть произведение). Ширина рассматриваемого алгоритма равна единице, высота — четырём.

Чтобы вычислить значение данного арифметического выражения, можно изменить порядок действий, использовав их формальные свойства. Это может изменить число операций и высоту алгоритма (число тактов) после распараллеливания. В ряде работ исследуется проблема оптимизации при изменении порядка действии с помощью коммутативного, ассоциативного и дистрибутивного свойств действий (см., например, [74]).

Изменив порядок действий в приведённом выше конкретном выражении, получив $b = ((a_1 \times a_2) \times (a_3 \times a_5) + (a_4 \times a_5)$. Это выражение уже хорошо распараллеливается на двух процессорах по схеме рис. 4.5. Здесь высота $t2$ равна трём вместо четырёх, зато число операций t_1 увеличилось на единицу (пять вместо четырёх).

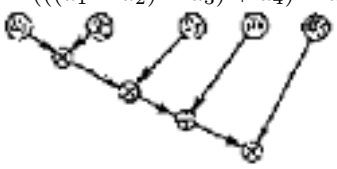
Уровень	$b = (((a_1 \times a_2) \times a_3) + a_4) \times a_5$
0 1 2 3 4	
	Высота схемы 4, ширина 1

Рис 4.4

Уровень	$b = ((a_1 \times a_2) \times (a_3 \times a_5) + (a_4 \times a_5)$
0 1 2 3	
	2-высота схемы 3

Рис 4.5

Отметим, что для итерационного процесса также можно ставить вопрос о распараллеливании, понимая под этим вычислительную схему, в которой каждая итерация протекает по параллельной схеме.

Если отвлечься от ограничения на число процессоров и от особенностей управления ими, то распараллеливание алгоритмов, по существу, представляет собой построение алгоритмов возможно меньшей высоты. Может оказаться, что число операций последовательной схемы совпадает с числом операций параллельной схемы (см., например, схемы на рис. 4.6 и 4.7).

Однако в некоторых случаях, для того чтобы уменьшить высоту, приходится увеличивать, причем иногда значительно, число операций. Это обстоятельство позволяет выделить два способа распараллеливания. Первый способ состоит в распараллеливании данного последовательного алгоритма. При этом не возникает необходимости в лишних операциях. Второй способ — построение нового, уже распараллеленного алгоритма, число операций в котором может увеличиться (см., например, схемы на рис. 4.6 и 4.8).

4.2 МЕТОД СДВАИВАНИЯ

Одна и та же вычислительная задача может решаться алгоритмами различной высоты и различной сложности (под сложностью понимаем число операций). В данном параграфе проиллюстрируем это двумя примерами, идея которых играет большую роль в построении алгоритмов малой высоты.

Пример 4.2. Пусть требуется умножить n чисел $a_1, a_2, \ldots, a_n$, используя p процессоров ($n \geq 2$; $p \geq 1$). Построим схему, p-высота которой $m(n,p) = t_p(n)$ минимальна.

Сначала рассмотрим три частных случая. Пусть $n = 2^k, p = 1$. Тогда последовательное умножение n чисел даёт схему, высота которой равна $n-1$, а ширина — единице (например, схема на рис. 4.6). Более детально рассмотрим ситуацию, когда $n = 2^k$, $p = n/2$. Метод сдваивания состоит в том, что каждое число i-го уровня получается как произведение двух соседних чисел, находящихся на $(i-1)$-м уровне ($i \geq 1$) (например, схема на рис. 4.7). Очевидно, что число уровней равно k (не считая нулевого), $k = \lceil \log_2 n \rceil$. Таким образом,

Уровень	$b = a_1 a_2 \ldots a_8$
0	$a_1 \quad a_2 \quad a_3 \quad a_4 \quad a_5 \quad a_6 \quad a_7 \quad a_8$
1	$a_1 a_2$
2	$(a_1 a_2) a_3$
3	$(a_1 a_2 a_3) a_4$
4	$(a_1 a_2 a_3 a_4) a_5$
5	$(a_1 a_2 a_3 a_4 a_5) a_6$
6	$(a_1 a_2 a_3 a_4 a_5 a_6) a_7$
7	$(a_1 a_2 a_3 a_4 a_5 a_6 a_7) a_8$
	Высота схемы 7, ширина 1

Рис. 4.6

высота схемы равна $\lceil \log_2 n \rceil$ ширина $n/2$. Случай, когда $2^{k-1} < n < 2^k, p = 2^{k-1}$, сводится к схеме на рис. 4.7 путём добавления в произведение $2^k - n$ сомножителей, равных 1. Высота такой схемы также равна $\lceil \log_2 n \rceil$, а ширина 2^{k-1}(здесь уже появляются лишние операции).

Рассмотрим две более общие ситуации. Если $p \geq n/2$, то применима схема, незначительно отличающаяся от схемы рис. 4.7: если на $(i-1)$-м уровне находится нечётное число результатов, то последнее число i-го уровня получается умножением последнего результата предыдущего уровня на единицу. Высота и этой схемы остается равной $\lceil \log_2 n \rceil$, а ширина равна$\lceil n/2 \rceil$.

Уровень	$b = a_1 a_2 \ldots a_8$
0	$a_1 \quad a_2 \quad a_3 \quad a_4 \quad a_5 \quad a_6 \quad a_7 \quad a_8$
1	$a_1 a_2 \quad a_3 a_4 \quad a_5 a_6 \quad a_7 a_8$
2	$(a_1 a_2)(a_3 a_4) \quad (a_5 a_6)(a_7 a_8)$
3	$(a_1 a_2 a_3 a_4)(a_5 a_6 a_7 a_8)$
	Высота схемы 3, ширина 4

Рис. 4.7

Пусть теперь $p < n/2$. На первом шаге можно составить p попарных произведений, и число сомножителей уменьшится на p. Если $n - p > 2p$, то процесс повторяется с загрузкой всех процессоров и т. д. Этот процесс продолжается до получения n_1 сомножителей, $n_1 = p + r, 0 \le r < p$, где r — неотрицательный остаток от деления n на p. Очевидное представление $n = p\lfloor n/p \rfloor + r$ даёт $n - p(\lfloor n/p \rfloor - 1) = p + r$. Следовательно, для получения $n_1 = p + r$ потребуется $(\lfloor n/p \rfloor - 1)$ тактов. Теперь уже $p > n_1/2$, и p-высота, как было показано выше, равна $\lceil \log_2 n_1 \rceil = \lceil \log_2(p + r) \rceil$. Полная p-высота равна $\lfloor n/p \rfloor - 1 + \lceil \log_2(p + r) \rceil = \lfloor n/p \rfloor + \lceil \log_2((p + r)/2) \rceil$.

Нетрудно проверить, что оба случая охватываются формулой

$$m(n, p) = \lfloor n/p \rfloor + \lceil \log_2 \min\{n, (p + r)/2\} \rceil.$$

Величина $m(n, p)$ является нижней границей для p-высоты любого алгоритма, который состоит в вычислении одного числа на p процессорах, исходя из n данных чисел, если все эти числа фактически участвуют в вычислении.

Ускорение в методе сдваивания является небольшим, так как $(n - 1)/\lceil \log_2 n \rceil \ll \lceil n/2 \rceil = p$. Это объясняется тем, что все процессоры используются только на первом такте.

Уменьшение числа процессоров может лишь незначительно увеличить высоту алгоритма. Так, умножение 64 чисел методом сдваивания при наличии 32 процессоров осуществляется алгоритмом, высота которого равна шести. В данном случае при нахождении результатов только первого уровня все процессоры загружены. Если использовать 10 процессоров, т. е. уменьшить их число в три раза, то высота алгоритма будет девять, то есть увеличится только в полтора раза. Зато при нахождении результатов пяти первых уровней будут работать все 10 процессоров.

Метод сдваивания можно применить не только к умножению, но и к сложению n чисел, к нахождению наибольшего среди n чисел, и вообще для вычисления результата любой ассоциативной операции, в частности для умножения нескольких матриц. Пусть выполнение бинарной операции $\times$ требует t тактов. Тогда нахождение результата $a_1 \times a_2 \times \ldots \times a_n$ при помощи метода сдваивания выполняется за $t\lceil \log_2 n \rceil$ тактов. Здесь a_i, $i = \overline{1, n}$, — не обязательно числа.

Уровень	$a_1a_2 \quad a_1a_2a_3 \ldots a_1a_2 \ldots a_8$
0	$a_1 \quad a_2 \quad a_3 \quad a_4 \quad a_5 \quad a_6 \quad a_7 \quad a_8$
1	$a_1a_2 \quad \underline{a_3a_4} \quad \underline{a_5a_6} \quad \underline{a_7a_8}$
2	$(a_1a_2)a_3 \quad (a_1a_2)(a_3a_4)$
	$\underline{(a_5a_6)a_7} \quad \underline{(a_5a_6)(a_7a_8)}$
3	$(a_1a_2a_3a_4)a_5 \quad (a_1a_2a_3a_4)(a_5a_6)$
	$(a_1a_2a_3a_4)(a_5a_6a_7) \quad (a_1a_2a_3a_4)(a_5a_6a_7a_8)$
	Высота схемы 3, ширина 4

Рис. 4.8

Пример 4.3. При умножении часто бывает нужно получить все последовательные произведения $a_1a_2, a_1a_2a_3, \ldots, a_1a_2 \ldots a_n$. Небольшое видоизменение схемы на рис. 4.7 позволяет дать алгоритм логарифмической высоты для решения и этой задачи.

Согласно схеме на рис. 4.8 четыре процессора работают на всех тактах. В результате кроме семи нужных чисел было получено пять лишних чисел (на рис. 4.8 они подчеркнуты). Число бинарных операций 12 вместо 7. Однако высота схемы для вычисления произведения n чисел по-прежнему равна $\lceil \log_2 n \rceil$, а не $n - 1$, как при последовательном умножении.

4.3 РАСПАРАЛЛЕЛИВАНИЕ РЕКУРСИЙ

Очень важный класс вычислений возникает при рассмотрении рекуррентных соотношений. В этом случае нужно решить систему уравнений, левые части которых зависят от значений параметров, вычисленных на предыдущих шагах. Такие рекуррентные соотношения (рекурсии) часто встречаются в матричных операциях. Они являются основой многих методов для вычисления значений функций и содержатся в некоторых методах численного интегрирования.

Наиболее часто встречается линейная рекурсия первого порядка. Рассмотрим её с точки зрения распараллеливания. Линейная рекурсия первого порядка имеет вид $x_i = a_i x_{i-1} + b_i, i = \overline{1, n}$, где $x_0 = b_0$ —некоторое начальное значение, а все коэффициенты a_i, b_i, известны $(i = \overline{1, n})$.

Вычислительный процесс, задаваемый этими рекуррентными соотношениями, является сугубо последовательным, так как каждое x_i, не может быть вычислено, пока значение x_{i-1} не известно. Таким образом, для вычисления x_n потребуется $2n$ операций (тактов). Однако ниже будет показано, что и в этом случае можно ввести параллелизм в вычисления.

Типичным примером линейной рекурсии первого порядка является схема Горнера для вычисления значения многочлена (см. упражнение 4.3). Для решения этой задачи рассмотрим еще один прием, который успешно используется. Он состоит в разбиении многочлена n-й степени на r слагаемых и в параллельном вычислении значений этих слагаемых. Разбиение имеет вид

$$p_n(x) = a_0 + a_1x + a_2x^2 + \ldots + a_nx^n = p_m(x) + \sum_{j=1}^{r-1} q_k^j(x)x^{m_j}.$$

Здесь $p_n(x) = a_0 + a_1x + a_2x^2 + \ldots + a_nx^n, m < n$, а многочлены $q_k^j(x)$ имеют одну и ту же степень k. Показатели степеней $m_1, m_2, \ldots, m_{r-1}$ образуют арифметическую прогрессию ($r \geq 4$). Величина r может изменяться с изменением уровня.

Вычисление значений многочленов $q_k^j(x)$ осуществляется по одинаковым ветвям дерева. При этом параллельно вычисляются степени x^{m_j} ($j = \overline{1, r-1}$). Если s— число тактов для вычисления значения многочлена $q_k^j(x)$, то многочлен $p_m(x)$ должен быть выбран таким, чтобы его значение вычислялось параллельно по другой ветви дерева за $s+1+\lceil \log_2(r+1) \rceil$ тактов. В этом состоит суть рассматриваемого приема.

Разбиение многочлена можно систематически осуществлять различными способами. Существуют алгоритмы, как для фиксированного числа процессоров, так и для случая, когда их число не ограничено.

На рис. 4.9 изображена часть дерева для вычисления значения многочлена $p_{36}(x)$, а p_m и q_k^jзначения многочленов $p_m(x)$ и $q_k^j(x)$.

Метод свертки, подобный только что описанному приему, существует и для линейной рекурсии. Суть этого метода состоит в том, что группируются вместе несколько членов рекуррентной последовательности.

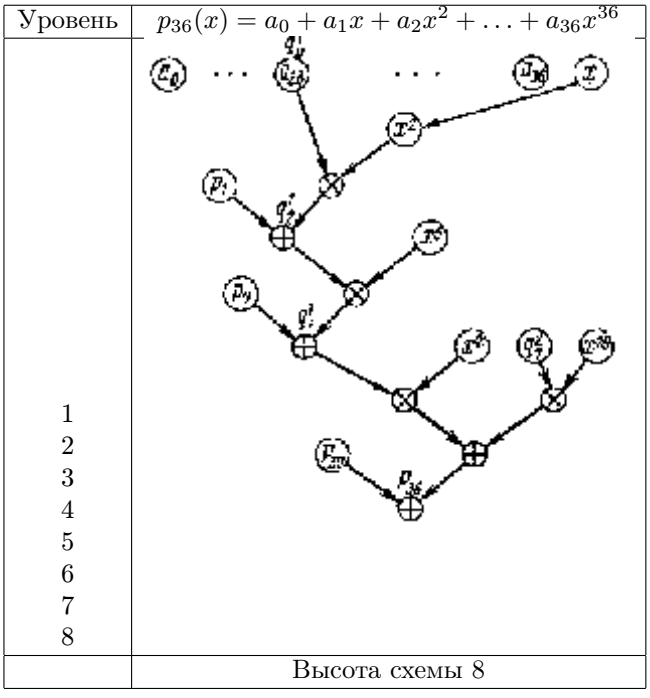


Рис. 4.9

Рассмотрим случай, когда объединяются вместе последовательные члены исходной рекурсии, чтобы получить две новые рекурсии. Каждая новая рекурсия имеет свое начальное значение, свои коэффи-

циенты и короче исходной рекурсии в два раза. Члены двух новых рекурсий чередуются в исходной рекурсии. С каждой из двух новых рекурсий поступаем аналогично и т. д.

Пусть на шаге i исходная рекурсия имеет вид $x_i = a_i x_{i-1} + b_i$. Тогда предыдущий шаг есть $x_{i-1} = a_{i-1}x_{i-2} + b_{i-1}$. Выразив из этих двух уравнений x_i через x_{i-2}, получим

$$x_i = (a_i a_{i-1})x_{i-2} + (a_i b_{i-1} + b_i),$$

что также является линейной рекурсией. Однако теперь уже каждое x_i может быть вычислено, если известно значение x_{i-2} и не требуется ждать вычисления x_{i-1}. Параллелизм вводится в вычисление не только новых коэффициентов, (они находятся в круглых скобках) но и значений x_i и x_{i-1}.

Чтобы проиллюстрировать этот процесс, на рис. 4.10 показано дерево для вычисления значений восьми первых членов линейной рекурсии. Все произведения коэффициентов a_i, необходимые на уровнях 3 и 5, вычисляются отдельно и параллельно.

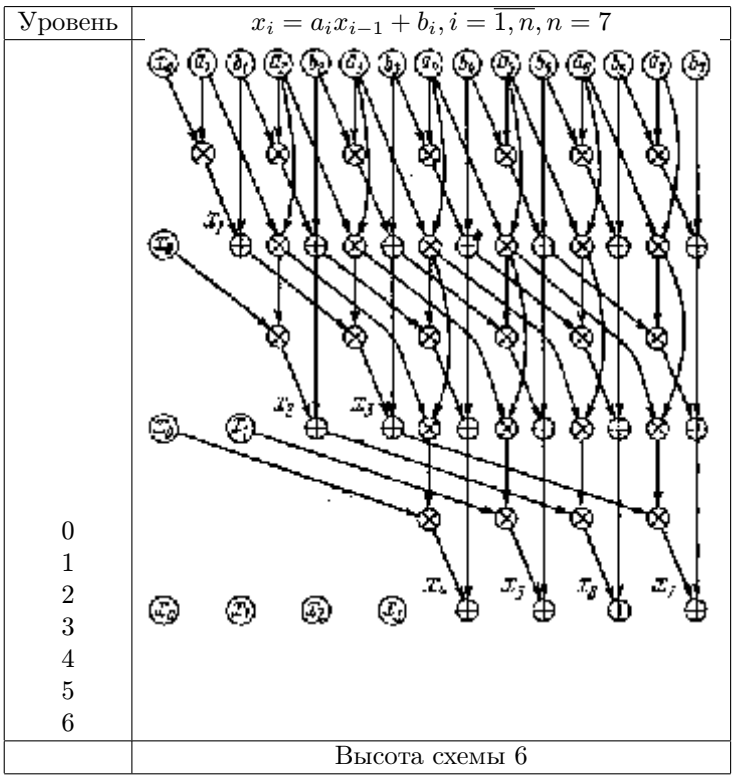


Рис. 4.10

Проиллюстрированный метод известен как циклическая редукция. Удивительно, как много параллелизма он может ввести в алгоритмы, которые до этого рассматривались как сугубо последовательные. Этот метод исключительно хорош при решении, например, трёхдиагональных систем при помощи исключения Гаусса.

Приведем еще один способ распараллеливания выделений для рассматриваемой рекурсии. Исходное рекуррентное соотношение $x_i = a_i x_{i-1} + b_i$ равносильно равенству

$$\begin{pmatrix} x_i \\ 1 \end{pmatrix} = \begin{pmatrix} a_i & b_i \\ 0 & 1 \end{pmatrix} \begin{pmatrix} x_{i-1} \\ 1 \end{pmatrix}$$

и, следовательно,

$$\begin{pmatrix} x_n \\ 1 \end{pmatrix} = \begin{pmatrix} a_n & b_n \\ 0 & 1 \end{pmatrix} \cdots \begin{pmatrix} a_2 & b_2 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} a_1 & b_1 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} b_0 \\ 1 \end{pmatrix}.$$

Умножение двух матриц вида $\begin{pmatrix} a & b \\ 0 & 1 \end{pmatrix}$ на одном процессоре выполняется за три такта, умножение матрицы этого вида на вектор — за два. Теперь можно применить прием сдваивания для нахождения произведения n матриц на p процессорах.

4.4 УМНОЖЕНИЕ МАТРИЦЫ НА ВЕКТОР

Действие умножения двух матриц соответственно порядков $m \times n$ и $n \times l$ естественным образом распараллеливается, так как умножения элементов первой матрицы на элементы второй параллельны. Затем нужно произвести параллельные сложения по n слагаемым, что по схеме сдваивания осуществляется за $\lceil \log_2 n \rceil$ тактов. Полная высота оказывается равной $\lceil \log_2 n \rceil + 1$ независимо от числа строк первой и от числа столбцов второй матрицы. Конечно, ширина алгоритма и количество операций сильно зависят от этих чисел.

Количество сложений сокращается, если строки первого множителя (или столбцы второго) слабо заполнены. Так, если число ненулевых элементов в каждого строке первого множителя не превосходит q, то сложение осуществляется за $\lceil \log_2 q \rceil$ тактов. В частности при $q = 2$ высота алгоритма становится равной двум. Это имеет место для двухдиагональных матриц и для квазидиагональных, состоящих из клеток второго порядка. Число умножений и ширина алгоритма тоже резко уменьшаются.

Если матрица раскладывается в произведение не большого числа разреженных матриц, то это обстоятельство может быть использовано при умножении матрицы на вектор (или на матрицу) для значительного уменьшения числа операций, а иногда и для уменьшения высоты алгоритма. Типичным примером может служит тензорное произведение.

Тензорным произведением $m \times n$ -матрицы A и $p \times q$ матрицы B называется $mp \times nq$-матрица C,

$$C = A \otimes B = \begin{pmatrix} a_{1,1}B & a_{1,2}B & \dots & a_{1,n}B \\ a_{2,1}B & a_{2,2}B & \dots & a_{2,n}B \\ \dots & \dots & \dots & \dots \\ a_{m,1}B & a_{m,2}B & \dots & a_{m,n}B \end{pmatrix}.$$

Пусть квадратная матрица C порядка mn есть тензорное произведение $C = A \otimes B$ матриц соответственно порядков m и n. Тогда, как известно, $C = (A \otimes I_n)(I_m \otimes B)$, где I_m и I_n — единичные матрицы соответствующих порядков.

Строки матрицы $A \otimes I_n$ содержат m, а строки матрицы $I_m \otimes B$ — n ненулевых элементов. Высота алгоритма умножения матрицы C на вектор становится равной $\lceil \log_2 m \rceil + 1 + \lceil \log_2 n \rceil + 1 = 2 + \lceil \log_2 m \rceil + \lceil \log_2 n \rceil$. Правая часть последнего равенства равна либо $2 + \lceil \log_2 mn \rceil$, либо $3 + \lceil \log_2 mn \rceil$. Таким образом, высота даже увеличивается на 1 или 2. Но число умножений резко снижается: вместо $т^2п^2$ делается $m(mn) + n(mn) = mn(m + n)$ умножений.

Соответственно если матрица A порядка $n_1 n_2 \dots n_k$ раскладывается в тензорное произведение матриц $A_1, A_2, \dots, A_k$ порядков $n_1, n_2, \dots, n_k$, то она может быть представлена в виде

$$A = (A_1 \otimes I_{n_2} \otimes \dots \otimes I_{n_k}) \times (I_{n_1} \otimes A_2 \otimes \dots \otimes I_{n_k} \times \dots \times (I_{n_1} \otimes I_{n_2} \otimes \dots \otimes A_{n_k}),$$

и использование этого факта снижает число операций умножения при умножении матрицы на вектор с $n_1^2 n_2^2 \dots n_k^2$ до $n_1 n_2 \dots n_k$ $(n_1 + n_2 + \dots + n_k)$. А так как операции умножения элементов матрицы на элементы вектора могут выполняться параллельно, то уменьшение их числа даёт значительное понижение p-высоты алгоритма при использовании небольшого числа процессоров — все процессоры будут загружены почти на всех тактах.

Укажем более общую ситуацию, в которой аналогичное обстоятельство имеет место.

Матрица A называется вполне приводимой, если она Может быть преобразована в квазидиагональную матрицу, состоящую из двух или больше клеток, при помощи перестановок строк и столбцов.

Тензорное произведение, в котором все сомножители, кроме одного, являются единичными матрицами, вполне приводимо и состоит из одинаковых клеток после приведения. Однако тот факт, что клетки одинаковы, не использовался в предыдущих рассуждениях. Поэтому верно следующее утверждение.

Если матрица A порядка $n = n_1 n_2 \dots n_k$ является произведением вполне приводимых матриц B_1, B_2, …, B_k (после приведения матрицы B_i получаются клетки раз мера n_i), то число операций умножения при нахождении произведения матрицы A на вектор равно $n_1 n_2 \dots n_k$ $(n_1 + n_2 + \dots + n_k)$.

Вполне приводимые матрицы часто возникают в задачах, в которых компоненты вектора нумеруются не одним, а двумя или большим числом индексов. Например, это имеет место в сетевых аппроксимациях.

Еще один очень важный пример — это быстрое преобразование Фурье. В нем используется естественно разложение матрицы преобразования в произведение вполне приводимых матриц, каждая из которых после приведения состоит из клеток второго порядка.

4.5 ХАРАКТЕРИСТИКА НЕКОТОРЫХ ПАРАЛЛЕЛЬНЫХ ПРОЦЕССОВ

Изучение численных алгоритмов с точки зрения их распараллеливания только начинается. Среди приемов, используемых для уменьшения числа процессоров в задачах линейной алгебры, следует отметить приемы разложения на клетки и различные формы факторизации

Предложены приемы для матриц общего вида, атак же для треугольных, трёхдиагональных, квазитрёхдиагональных, ленточных и квазиленточных матриц. Во многих работах изучаются варианты метода исключения Гаусса.

В табл. 4.1 указаны некоторые векторные и матричные операции [48]. В этой таблице функция $m(n,p)$ определена как

$$m(n,p) = \lfloor n/p \rfloor + \lceil \log_2 \min\{n, (p+r)/2\} \rceil,$$

где n — количество заданных чисел; p — число процессоров, r — неотрицательный остаток от деления n на p.

Рассмотрим систему линейных уравнений вида $Ax = B$, где A — некоторая невырожденная матрица. Не говоря о структуре матрицы A, укажем на два общих обстоятельства, которые имеют место. При параллельных вычислениях выбор алгоритма и соответствующей структуры данных производится намного тщательнее, чем при последовательных. Приемы, которые хороши для заполненных матриц, ни в коем случае не являются оптимальными для разреженных.

Таблица 4.1

Операция	Количество чисел	Число процессоров	Число тактов
$x + y$	$2n$	n	1
$x + y$	$2n$	p	$\lceil n/p \rceil$
(x, y)	$2n$	n	$\lceil \log_2 n \rceil + 1$
(x, y)	$2n$	p	$m(2n, p)^*$
(x, y)	$2n$	p	$\lceil n/p \rceil + m(2n, p)^{**}$
$\|x\|_1$	n	p	$\lceil n/p \rceil + m(2n, p)$
$\|x\|_2$	n	p	$\lceil n/p \rceil + m(2n, p) + 1$
$A \times B$	$mn + nl$	mnl	$\lceil \log_2 n \rceil + 1$
$A \times x$	$n^2 + n$	n^2	$\lceil \log_2 n \rceil + 1$
A^{-1}	n^2	$O(n^4/log_2 n)$	$0(log_2^2 n)$
			* Режим MIMD ** Режим SIMD

При численном решении систем вида $Ax = b$ используются прямые и итерационные методы.

В прямых методах, как правило, осуществляется факторизация матрицы A так, чтобы получить треугольную систему уравнений, которая затем решается. Приведем две хорошо известные факторизации: $PA = LU$ и $QA = R$, где P и Q — перестановочная и ортогональная матрицы, L — нижняя треугольная, U и R — верхние треугольные матрицы. Примером прямого метода может служить метод исключения неизвестных ($PA = LU$), предложенный Гауссом. В этом методе можно ввести параллелизм в вычисления, как при факторизации, так и при подстановке значений неизвестных. Аналогичное обстоятельство имеет место и в других прямых методах.

Основной прием, который используется в итерационных методах, состоит в том, что матрица A представляется в виде $A = M - N$, а затем при помощи рекуррентного соотношения $Mx^i = Nx^{i-1} + b$ находится последовательность векторов $\{x^i\}$. Начиная с вектора x^0, повторяется решение приведённого рекуррентного уравнения, пока последовательность $\{x^i\}$ не сходится по выбранной норме.

В итерационном методе Якоби используется представление $M = A_D$, $N = -(A_L + A_U)$, где матрица A развивается на части: $A = A_D + A_L + A_U$. Здесь A_D —диагональная часть матрицы A, A_L и A_U

— её нижняя и верхняя треугольные части. В итерационном методе Гаусса — Зейделя матрица A представляется в виде $A = M - N$, где $M = A_D + A_l, N = -A_u$.

Метод Якоби сходится чрезвычайно медленно. В нем, как это видно из представления матрицы A, при вычислении новых значений компонент вектора участвуют только старые значения. В методе же Гаусса — Зейделя при нахождении новых значений компонент вектора используются как старые, так и уже вычисленные новые значения. Это во много раз ускоряет сходимость (в отдельных случаях в сотни раз).

Оба итерационных метода допускают распараллеливание вычислений. Они, как правило, не используются для решения систем с заполненными матрицами, но применяются при решении систем с разреженными матрицами специальной структуры.

Параллельным вычислениям в линейной алгебре посвящена обширная литература (например, [48, 49]).

В настоящее время разработаны параллельные про процессы для численного решения многих задач алгебры математического анализа, математической физики, экономики статистики.

УПРАЖНЕНИЯ

4.1. Привести два последовательных алгоритма для решения одной и той же вычислительной задачи, один из которых удобен для распараллеливания, а другой нет.

4.2. Рассмотрим алгоритм, состоящий в вычислении несколько чисел, по крайней мере одно из которых зависит от всех n данных чисел ($p \geq n/2$). Показать, что величина $\lceil \log_2 n \rceil$ является нижней границей для высоты такого алгоритма. Привести пример, когда эта нижняя граница достигается.

4.3. Обосновать схему Горнера для вычисления значения многочлена в точке a (вывести систему рекуррентных соотношений), использовав одно из двух его представлений

$$p_n(x) = b_0 x^n + b_1 x^{n-1} + \ldots + b_{n-1} x + b_n =$$

$$= x(\ldots(x(xb_0 + b_1) + b_2) + b_3) + \ldots) + b_n =$$

$$= (x - a)(c_0 x^{n-1} + c_1 x^{n-2} + \ldots + c_{n-2} x + c_{n-1}) + r.$$

4.4. Достроить дерево рис. 4.9.

4.5. Для вычисления значений 23 первых членов линейной рекурсии продолжить дерево рис. 4.10.

4.6. Подсчитать высоту и сложность схемы, осуществляющей распараллеливание вычислений для линейной рекурсии при помощи матрицы.

4.7. Доказать следующие свойства тензорного произведении:

а) если существуют произведения AC и BD, то $(A \otimes B)(C \otimes D) = AC \otimes BD$;

б) $(A \otimes B)^T = A^T \otimes B^T$;

в) если A и B — невырожденные матрицы, то $(A \otimes B)^{-1} = A^{-1} \otimes B^{-1}$.

4.8. Доказать результаты из табл. 4.1.

Глава 5

ПАРАЛЛЕЛЬНЫЕ АЛГОРИТМЫ ВЕТВЕЙ И ГРАНИЦ

Эта глава в основном посвящена параллельным алгоритмам перебора (p-алгоритмам перебора). Описан общий p-алгоритм ветвей и границ и рассмотрены его конкретизации для решения СЦЛ-, БЛ- и ЦЛ-задач. работа некоторых алгоритмов иллюстрируется решением конкретных задач.

В алгоритмах перебора существенно используется тот факт, что целочисленные переменные принимают конечное число значений. Алгоритм перебора, как правило, представляет собой совокупность процедур, выполнение которых даёт решение задачи в результате анализа сравнительно небольшого числа вариантов без рассмотрения всех остальных. Эффективность метода перебора существенно зависит от интуиции и фантазии исследователя.

5.1 ВЫПОЛНЕНИЕ p-ВЕТВЕВОГО ОБХОДА ДЕРЕВА ПЕРЕБОРА

В упрощённой ситуации покажем, как можно осуществить p-ветвевой обход дерева перебора. При решении, например, БЛ-задачи максимизации, когда все время уточняются нижняя и верхняя границы для оптимального значения целевой функции, использование информации о границах сразу в p ветвях может ускорить счёт практически в p раз, поскольку обмен между ветвями чрезвычайно мал.

Будем различать левую и правую текущие ветви. Номера элементов, просматриваемых левой (правой) текущей ветвью, образуют возрастающую (убывающую) последовательность. Для левой (правой) текущей ветви нужно задать верхнюю (нижнюю) границу, до которой она движется. Таким образом, две ветви из p текущих ветвей могут либо двигаться друг другу навстречу, либо расходиться, либо двигаться в одном направлении (либо влево, либо вправо). Отметим, что в любой части дерева перебора можно сконцентрировать несколько текущих ветвей.

Пример 5.1. Найти все бинарные решения уравнения $1x_1+2x_2+3x_3+4x_4+5x_5+6x_6+7x_7 = 7$, $x_j = 0, 1$, $j = \overline{1,7}$.

Поэлементный перебор требует подстановки в уравнение всех $2^7 = 128$ бинарных векторов и отбора тех из них, которые удовлетворяют ему. Ниже также осуществляется перебор всех 128 бинарных векторов. Однако ненужные элементы исключаются не по одному, а целыми множествами. Это самое главное!

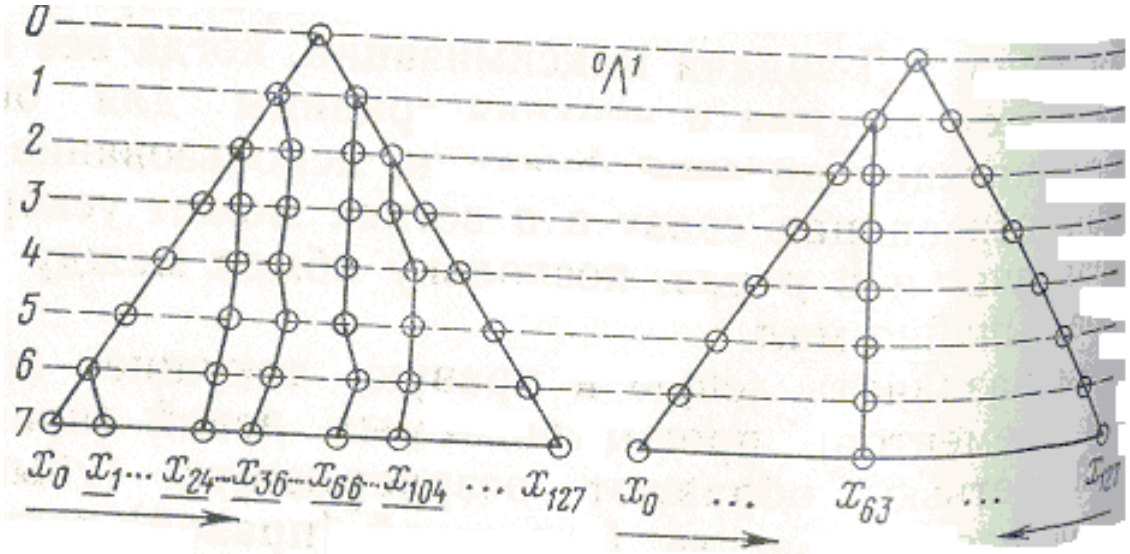


Рис. 5.1 Рис. 5.2

На рис. 5.1 и 5.2 бинарные решения уравнения подчёркнуты, а исключаемые множества — нет. Вектор x_i представляет собой бинарный вектор $(\alpha_{i,1}, \alpha_{i,2}, :, \alpha_{i,7})$, где $i = \sum_{j=1}^{7} \alpha_{i,j} 2^{7-j}$. Например,

$$x_{23} = (0, 0, 1, 0, 1, 1, 1),$$

$$x_{24} = (0, 0, 1, 1, 0, 0, 0).$$

Таким образом, бинарный вектор x_i есть двоичное представление своего индекса i. Поэтому нет необходимости выписывать векторы x_i для всех $i = \overline{0, 127}$.

Левая ветвь одноветвевого обхода дерева перебора даёт последовательность векторов

$$x_0, \underline{x_1}, x_2, \ldots, x_{23}, \underline{x_{24}}, x_{25}, \ldots, x_{35}, \underline{x_{36}}, x_{37}, \ldots, x_{65}, \underline{x_{66}}, x_{67}, \ldots, x_{103}, \underline{x_{104}}, x_{105}, \ldots, x_{127}.$$

На рис. 5.1 и 5.2 изображены одноветвевой (левая) и двухветвевой (левая и правая) обходы дерева перебора. Все бинарные решения уравнения $\sum_{j=1}^{7} j x_j = 7$ исчерпываются векторами

$$x_1 = (0, 0, 0, 0, 0, 0, 1),\ x_{24} = (0, 0, 1, 1, 0, 0, 0),$$

$$x_{36} = (0, 1, 0, 0, 1, 0, 0),\ x_{66} = (1, 0, 0, 0, 0, 1, 0),$$

$$x_{104} = (1, 1, 0, 1, 0, 0, 0).$$

5.2 ОБЩИЙ p-АЛГОРИТМ ВЕТВЕЙ И ГРАНИЦ

Для решения задачи оптимизации $\max\{z(x),\ x \in S_0\}$ описан параллельный алгоритм (p-алгоритм) ветвей и границ. Процесс решения этой задачи иллюстрируется с помощью дерева, называемого деревом перебора (деревом вариантов). Вершина v_0 дерева перебора соответствует сформулированной задаче, а вершины v_k - задачам $\max\{z(x) \mid x \in S_k \subseteq S_0\}$ $(k = 1, 2, 3, \ldots)$. Здесь S_0 — множество элементов произвольной природы. Элементы множества S_0 называются допустимыми решениями задачи оптимизации $\max\{z(x) \mid x \in S_0\}$.

Покрытие. Дерево перебора строится с помощью покрытия P_k множества S_k. Покрытие P_k представляет собой систему подмножеств $S_{k,s}$ множества S_k $(s = \overline{1, |P_k|})$, для которой $\bigcup_{s=1}^{|P_k|} S_{k,s} = S_k$. Во

многих реализациях алгоритма покрытие P_k множества S_k является его разбиением. Для вершины v_k строим $|P_k|$ новых вершин (ее последователей, точнее, её непосредственных последователей) и $|P_k|$ рёбер, соединяющих её с этими вершинами. При этом каждому подмножеству $S_{k,s}$ соответствует только одна новая вершина.

Вычисление границ. В вершине v_k задача $\max\{z(x) \in S_k\}$ не решается, а решается релаксированная (ослабленная) задача $\max\{z(x) \in T_k \supseteq S_k\}$. Заметим, что эффективный выбор множества T_k требует большого искусства. Обычно необходим компромисс между точностью верхней границы $\overline{z_k}$, даваемой ослабленной задачей, и трудоёмкостью решения этой задачи. Если известно какое-нибудь допустимое решение x' из S_k, то оно даёт нижнюю границу $\underline{z_k} = z(x')$ для неизвестного оптимального значения $z_k^* = z(x^*(k))$.

Вырождение. Вершина v_k дерева перебора называется вырожденной, если известно, что множество S_k не содержит ни одного допустимого решения со значением целевой функции, большим, чем $\underline{z_0}$. Таким образом, вершина v_k является вырожденной, если $\overline{z_k} < \underline{z_0}$, где $\underline{z_0}$ значение целевой функции для наилучшего уже известного допустимого решения из S_0. Ясно, что нет смысла продолжать построение дерева перебора из вырожденной вершины.

Ветвление. Вершина дерева перебора, которая не является вырожденной и не имеет последователей, называется активной. Ветвление есть выбор активной вершины для её исследования. Существует много возможных правил ветвления. Так, по виду используемой в текущий момент части дерева перебора различают одноветвевую и многоветвевую схемы перебора (на рис. 5.3 вырожденные вершины подчёркнуты).

Одноветвевая схема перебора состоит в выборе одного из последователей вершины, которая рассматривается как текущая. Если текущая вершина v_k является вырожденной, то просто возвращаемся назад вдоль пути p_k, соединяющего вершины v_0 и v_k, пока не встречается вершина, имеющая, по крайней мере, один активный последователь. Для дальнейшего исследования выбирается один из этих последователей.

В многоветвевой схеме перебора выбор вершины для исследования осуществляется по какому-либо правилу из множества всех вершин, являющихся в данный момент активными. Если не существует ни одной активной вершины, то перебор в обеих схемах завершен. В одноветвевой схеме перебора хранится и обрабатывается информация только для одного пути, а в многоветвевой — для многих. Придерживаясь какой-нибудь одной схемы перебора, можно распараллеливать связанные с ней вычисления. Однако допустим и такой параллельный алгоритм перебора, на каждой ветви которого используется либо одноветвевая, либо многоветвевая схема перебора.

Для определённости будем считать, что осуществляется p-ветвевой обход дерева перебора. Это означает, что одновременно рассматриваются p путей (текущих ветвей дерева), выходящих из вершины v_0. Вершины дерева, пройденные l-й текущей ветвью, обозначаются через $v_{l,k}$, где индекс k указывает порядок, в котором вершины были рассмотрены на l-й текущей ветви ($l = \overline{1,p};\ k = 0,1,2,\dots$).

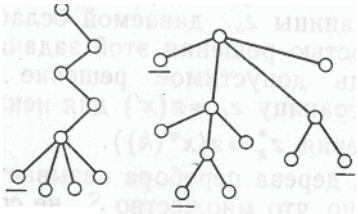


Рис. 5.3

p-алгоритм ветвей и границ

Шаг 1 (начало). Полагаем $\underline{z_0} = -\infty$, $\overline{z_0} = +\infty$. Исходя из активной вершины v_0, строим часть дерева перебора, достаточную для организации p текущих ветвей.

Шаг 2 (ветвление для $l = \overline{1,p}$). Если на l-й текущей ветви нет ни одной активной вершины, то переходим к шагу 6. По определённому правилу выбираем активную вершину $v_{l,k}$. Если задача $\max\{z(x) \mid x \in T_{l,k} \supseteq S_{l,k}\}$ еще не решена в вершине $v_{l,k}$, то переходим к шагу 4.

Шаг 3 (покрытие для $l = \overline{1,p}$). Строим покрытие $P_{l,k}$ множества $S_{l,k}$, $|P_{l,k}|$ вершин и $|P_{l,k}|$ рёбер, соединяющих эти вершины с $v_{l,k}$.

Шаг 4 (вычисление границ для $l = \overline{1,p}$). Решаем задачу $\max\{z(x) \mid x \in T_{l,k} \supseteq S_{l,k}\}$. Очевидно, что для $T_{l,k} = \varnothing$ вершина $v_{l,k}$ является вырожденной, и переходим к шагу 2. Если же целевая функция $z(x)$ не ограничена сверху на множестве $T_{l,k}$, то верхняя граница $\overline{z_{l.k}}$ равна $+\infty$, и переходим к шагу 2. Пусть $x^0(lk)$ — оптимальное решение рассматриваемой задачи. Тогда полагаем $\overline{z_{l,k}} = z(x^0(l,k))$. Если $x^0(l,k) \in S_{l,k}$, то $\underline{z_{l,k}} = \max\{\underline{z_{l,0}}, z(x^0(l,k))\}$.

Полагаем $\underline{z_0} = \max\limits_{l=1}^{p} \underline{z_{l,0}}$.

Шаг 5 (вырождение для $l = \overline{1,p}$). Каждая вершина $v_{l,k}$, для которой $\overline{z_{l,k}} \le \underline{z_0}$, является вырожденной. Переходим к шагу 2.

Шаг 6. Если l-я текущая ветвь закончила свою работу, то переносим её в еще не исследованную часть Дерева. Все p текущих ветвей заканчивают свою работу.

Шаг 7 (окончание). Если $\underline{z_0} > -\infty$, то допустимое решение, которое даёт это z_0, является оптимальным Для задачи $\max\{z(x) \mid x \in S_0\}$. Если же $\underline{z_0} = -\infty$, то либо $S_0 = \varnothing$, либо целевая функция $z(x)$ не ограничена сверху на множестве S_0.

Конечность. Укажем условия, выполнение которых гарантирует построение конечного дерева перебора.

А. Для каждой вершины v_k покрытие P_k множества S_k состоит из конечного числа элементов.

Б. Существует функция $f(S), S \subseteq S_0$ принимающая неотрицательные целочисленные значения и удовлетворяющая условиям: из $S_k \in P_l$ следует $f(S_k) < f(S_l)$; если $f(S_k) \le 1$, то может быть установлена вырожденность вершины v_k.

Для конечного множества S_0 в качестве такой функции можно взять, например, $f(S) = |S|$. Для некоторых конкретизации общего алгоритма будет показано существование таких функций.

ε-оптимальность. Важной особенностью любого алгоритма ветвей и границ является то, что он даёт допустимое решение задачи оптимизации и верхнюю границу для неизвестного оптимального значения целевой функции этой задачи. Таким образом, вычислитель всегда имеет возможность считать вершину v_k вырожденной, если верхняя граница $\overline{z_k}$ удовлетворяет неравенству $\overline{z_k} \le \underline{z_0}(1+\varepsilon)$, где величина ε задаётся вычислителем ($\varepsilon \ge 0$). Это может значительно уменьшить исследуемую часть дерева перебора.

5.3 ЛИНЕЙНАЯ РЕЛАКСАЦИЯ

Рассмотрим СЦЛ-задачу

$$\max\{z(x_1,x_2) = c_1x_1 + c_2x_2 \mid (x_1,x_2) \in S_0 = \{(x_1,x_2) \mid A_1x_1 + A_2x_2 = b,\ x_1 \ge 0,\ x_1 \in Z_{n_1},\ x_2 \ge 0\}\}.$$

Для этой задачи

$$S_k = \{(x_1,x_2) \mid A_1x_1 + A_2x_2 = b,\ l_k \le x_1 \le u_k,\ x_1 \in Z_{n_1},\ x_2 \ge 0\},$$

$$T_k = \{(x_1,x_2) \mid A_1x_1 + A_2x_2 = b,\ l_k \le x_1 \le u_k,\ x_2 \ge 0\}.$$

Здесь $l_k, u_k, l_k \le u_k$ — неотрицательные целочисленные векторы. Таким образом, релаксированная задача представляет собой Л-задачу оптимизации с двусторонними ограничениями на отдельные переменные.

Если оптимальное решение $(x_1^0(k), x_2^0(k))$ Л-задачи в вершине v_k не является элементом множества S_k, то строится покрытие. В этом случае некоторая базисная переменная x_{B_i} имеет значение $y_{i,0} = \lfloor y_{i,0} \rfloor + f_{i,0}$, где дробная часть $f_{i,0}$ числа $y_{i,0}$ положительная. Таким образом, в качестве покрытия множества S_k можно взять покрытие

$$P_k = \Big\{S_k \bigcap \{(x_1,x_2) \mid x_{B_i} \le \lfloor y_{i,0} \rfloor\},$$

$$S_k \bigcap \{(x_1,x_2) \mid x_{B_i} \ge \lceil y_{i,0} \rceil\}\Big\}.$$

Покрытие P_k является разбиением множества S_k. Оно даёт нижние и верхние границы для двух новых вершин. Вначале, когда еще ничего неизвестно о границах для переменных, в вершине v_0 полагаем $l_0 = (0,0,\ldots,0)$, $u_0 = (+\infty,+\infty,\ldots,+\infty)$.

Рассматриваемую Л-задачу целесообразно решать двойственным симплекс-алгоритмом, учитывающим алгоритмически двусторонние ограничения на отдельные переменные. На самом деле часто нет необходимости решать Л-задачу до конца, так как целевая функция убывает и на некоторой итерации двойственного симплекс-алгоритма может достигнуть значения $\underline{z_0}$, а этого вполне достаточно для того, чтобы установить вырожденность вершины.

Конечность алгоритма может быть гарантирована, если u_0 — заданный неотрицательный целочисленный вектор. В этом случае в качестве функции $f(S_k)$ можно взять функцию

$$f(S_k) = \prod_{j=1}^{n_1} (u_{j,k} - l_{j,k} + 1)$$

Очевидно, что так определённая функция удовлетворяет всем вышеуказанным требованиям. В частности, $f(S_k) = 1$ влечёт $x_1(k) = l_k = u_k$, что устанавливает вырожденность вершины.

Ветвление. Используемое здесь разбиение множества оставляет открытым вопрос о том, какую из базисных переменных следует выбирать для разбиения, если имеется больше одной переменной с $f_{i,0} > 0$. В действующих алгоритмах ветвей и границ выбор базисной переменной опирается на штрафы, вычисляемые для всех возможных разбиений. Штраф представляет собой нижнюю границу уменьшения целевой функции и может быть вычислен различными способами.

Одно правило, эффективность которого доказана, состоит в вычислении для каждого i, $f_{i,0} > 0$, нижнего штрафа g_i, верхнего штрафа h_i, и штрафа $p_i = \min\{g_i, h_i\}$. Штраф $g_i(h_i)$ есть уменьшение целевой функции после добавления ограничения $x_{B_i} \le \lfloor y_{i,0} \rfloor (x_{B_i} \ge \lceil y_{i,0} \rceil)$ и выполнения одной итерации двойственного симплекс-алгоритма. Приведем формулы, по которым вычисляют штрафы g_i,h_i.

Рассмотрим уравнение из оптимального базисного представления $x_{B_i} = y_{i,0} - \sum\limits_{j \in J} y_{i,j} x_j$, $f_{i,0} > 0$, где J — множество индексов небазисных переменных (переменная x_{B_i} является целочисленной). Тогда

$$x_{B_i} \le \lfloor y_{i,0} \rfloor \longleftrightarrow s_i = -f_{i,0} + \sum_{j \in J} y_{i,j} x_j.$$

Это даёт

$$g_i = \begin{cases} \min\limits_{y_{i,j} > 0, j \in J} \frac{f_{i,0} y_{0,j}}{y_{i,j}}, \\ +\infty,\ y_{i,j} \le 0,\ j \in J. \end{cases}$$

Аналогично вычисляется верхний штраф

$$h_i = \begin{cases} \min\limits_{y_{i,j} < 0, j \in J} \frac{(f_{i,0} - 1) y_{0,j}}{y_{i,j}}, \\ +\infty,\ y_{i,j} \ge 0,\ j \in J \end{cases}$$

Пусть $p_{i_0} = \max\limits_{f_{i,0} > 0} \min\{g_i, h_i\}$.Тогда выполняем разбиение по переменной $x_{B_{i_0}}$. При этом для $p_{i_0} = g_{i_0}$ выполняем ветвление вниз, а для $p_{i_0} = h_{i_0}$ — вверх.

Теперь покажем, как для ЦЛ-задачи можно более точно вычислить верхнюю границу. Если нулевые значения небазисных переменных дают нецелочисленное базисное решение, то по крайней мере, одна из небазисных переменных принимает положительное значение. Это даёт нижнюю границу $p = \min\limits_{j \in J} y_{0,j}$ для уменьшения целевой функции. Как следует из метода секущих плоскостей, для любого допустимого решения ЦЛ-задачи выполняется ограничение

$$s_i = -f_{i,0} + \sum_{j \in J} f_{i,j} x_j,\ s_i \ge 0,\ s_i \equiv 0.$$

Это уравнение даёт штраф

$$p'_i = \min_{f_{i,j} > 0, j \in J} \{f_{i,0} y_{0,j} / f_{i,j}\}.$$

Полагаем

$$p^* = \max\{p, \max_{f_{i,0} > 0} \max\{p_i, p'_i\}\}.$$

Тогда верхней границей в вершине v_k служит величина

$$\overline{z_k} = \lfloor z_k^0 - p^* \rfloor.$$

В алгоритмах ветвей и границ наиболее употребительным является выбор только одного из последователей текущей вершины. Это даёт преимущество при решении текущих задач. В текущей вершине хранится определённая информация, например обратная матрица, верхняя и нижняя границы. Поэтому гораздо экономнее рассматривать только одну текущую вершину (одну задачу), а не несколько сразу. Если текущая вершина становится вырожденной, то возвращаемся назад по единственному пути к вершине v_0, пока не встретится первая вершина, имеющая активный последователь. В некоторых случаях при выборе активной вершины привлекается дополнительная информация, а именно: число активных вершин, местоположение на дереве перебора, число целочисленных переменных.

Разбиение для одного простого ограничения. Рассмотрим ограничение $\sum\limits_{q\in Q} x_q = 1$, $x_q = 0, 1$, $q \in Q$. Пусть в оптимальном решении $0 < x_t^0(k) < 1$ для некоторого $t \in Q$. Тогда получаем разбиение $P_k = \{S_k \bigcap \{x \mid x_t = 0\},\ S_k \bigcap \{x \mid x_t = 1\}\} = \{S_{k,1}, S_{k,2}\}$. Ясно, что $S_{k,1} = S_k \bigcap \{x \mid \sum\limits_{q\in Q/t} x_q = 1\}$. Поэтому вполне правдоподобно, что множество $S_{k,1}$ намного больше множества $S_{k,2}$ и вершина $v_{k,1}$ может оказаться более трудной для исследования, чем $v_{k,2}$. По этой причине желательно выбрать такое разбиение $P'_k = \{S_{k,3}, S_{k,4}\}$, в котором подмножества $S_{k,3}$ и $S_{k,4}$ примерно одинаковы по числу элементов.

Заметим, что из $0 < x_t^0(k) < 1$ следует $0 < x_{t'}^0(k) < 1$ для некоторого $t' \in Q$. Рассмотрим разбиение $\{Q_1, Q_2\}$ множества Q, где $t_1 \in Q_1$, $t' \in Q_2$. Пусть

$$S_{k,3} = S_k \bigcap \Big\{x \mid \sum_{q\in Q_1} x_q = 1\},\ S_{k,4} = S_k \bigcap \{x \mid \sum_{q\in Q_2} x_q = 1\Big\}.$$

Если $Q_1 = \{t\}$, то $S_{k,3} = S_{k,2}$, $S_{k,4} = S_{k,1}$. Следует выбрать подмножества Q_1, Q_2 примерно одинаковыми по числу элементов.

5.4 НЕЯВНЫЙ ПЕРЕБОР

Неявный перебор — название класса алгоритмов ветвей и границ для решения БЛ-задачи

$$\max\{z(x) = \sum_{j=1}^{n} c_j x_j \mid x \in S_0 = \{x \mid \sum_{j=1}^{n} a_{i,j} x_j \le b_i,\ i = \overline{1,m},\ x_j = 0, 1,\ j = \overline{1,n}\}\}.$$

Без ограничения общности считаем, что $c = (c_1, c_2, \ldots, c_n) \le (0, 0, \ldots, 0)$, так как этого можно достигнув заменой переменных $x_j = 1 - x'_j$ для всех $c_j > 0$.

В вершине v_k дерева перебора разбиваем множество индексов $J = 1, 2, \ldots, n$ на три подмножества $J = J_k^- \bigcup J_k^+ \bigcup F_k$, где $J_k^- = \{j \mid x_j = 0,\ j \in J\}$, $J_k^+ = \{j \mid x_j = 1,\ j \in J\}$, $W_k = J_k^- \bigcup J_k^+$ (множество индексов зафиксированных переменных), $F_k = \{j \mid x_j = 0, 1,\ j \in J\}$ (переменные с индексами из множества F_k не зафиксированы). В вершине v_k рассматриваем БЛ-задачу

$$\max\{z_k(x) = \sum_{j\in F_k} c_j x_j + z_k^0 \mid \sum_{j\in F_k} a_{i,j} x_j \le s_{i,k},\ i = \overline{l,m},\ x_j = 0, 1,\ j \in F_k\},$$

где $z_k^0 = \sum\limits_{j\in J_k^+} c_j$, $s_{i,k} = b_i - \sum\limits_{j\in J_k^+} a_{i,j}$.

В качестве релаксированной задачи выступает задача

$$\max\{z_k(x) \mid x \in T_k = \{x \mid x_j = 0, 1,\ j \in F_k\}\}.$$

Поскольку $c \le 0$, то $x_j = 0$, $j \in F_k$, — оптимальное решение этой задачи со значением z_k^0 целевой функции. Таким образом, если $s_k = (s_{1,k}, s_{2,k}, \ldots, s_{m,k}) \ge (0, 0, \ldots, 0)$ или $z_k^0 \le \underline{z_0}$, то вершина v_k является вырожденной.

Если вершина v_k не является вырожденной, то покрытие имеет вид

$$P_k = \{S_k \bigcap \{x \mid x_j = 0\},\ S_k \bigcap \{x \mid x_j = 1\}\}$$

для некоторого $j \in R_k$. Здесь R_k — множество тех индексов j из F_k, для которых $a_{i,j} < 0$ по крайней мере для одного i с $s_{i,k} < 0$. Заметим, что если $R_k = \varnothing$, то вершина v_k является вырожденной. Общепринятое правило состоит в выборе такого индекса j_0, который минимизирует суммарную невязку $i_k(j) = \sum\limits_{i=1}^{m} \max\{0, -s_{i,k} + a_{i,j}\}$. Затем выбираем последователь вершины v_k соответствующий $x_{j_0} = 1$.

Рассматриваемое покрытие является разбиением и удовлетворяет требованиям конечности дерева перебора. В качестве $f(S_k)$ можно взять функцию $f(S_k) = 2^{|F_k|}$ ($|F_k| = 0$ для $F_k = \varnothing$).

Заменяющее ограничение. Вершина v_k также является вырожденной, если $t_{i,k} = \sum\limits_{i \in F_k} \min\{0, a_{i,j}\}$ $> s_{i,k}$ для некоторого i. Существуют и другие тесты, аналогичные этому и основанные на бинарности переменных. Эти тесты также целесообразно использовать и для заменяющего ограничения

$$\sum_{i=1}^{m} \sum_{j \in F_k} u_i a_{i,j} x_j \le \sum_{i=1}^{m} u_i s_{i,k},$$

где $u_i \ge 0$, $i = \overline{1,m}$. Ясно, что если не существует бинарных векторов, удовлетворяющих заменяющему ограничению, то их нет и для системы неравенств, из которой это заменяющее ограничение получено.

Важно выбрать вектор $u = (u_1, u_2, \ldots, u_m)$ так, чтобы возможность вырождения была большой. Пусть

$$S_k(u) = \Big\{x \mid \sum_{i=1}^{m} \sum_{j \in F_k} u_i a_{i,j} x_j \le \sum_{i=1}^{m} u_i s_{i,k},\ x_j = 0, 1,\ j \in F_k\Big\},$$

$$g(u) = \max\{z_k(x) \mid x \in S_k(u)\}.$$

Тогда под самым сильным заменяющим ограничением разумно понимать неравенство, определяемое вектором u^*, который минимизирует $g(u)$ на множестве всех неотрицательных u. К сожалению, это определение не даёт конкретного способа вычисления.

Однако приближение к u^*, которое легко вычисляется, может быть получено в результате замены условий бинарности $x_j = 0, 1, j \in F_k$, на неравенства $0 \le x_j \le 1, j \in F_k$. Тогда заменяющее ограничение, самое сильное относительно функции $g'(u)$, даётся вектором u^*, где (u^*, v^*) — оптимальное решение Л-задачи

$$\min\Big\{w_k(u,v) = \sum_{i=1}^{m} s_{i,k} u_i + \sum_{j \in F_k} v_j + z_k^0 \mid \sum_{i=1}^{m} a_{i,j} u_i + v_j \ge c_j,\ j \in F_k,\ u_i \ge 0,\ i = \overline{1,m},\ v_j \ge 0,\ j \in F_k\Big\}.$$

Покажем это.

Рассмотрим двойственную Л-задачу

$$\max\Big\{z_k(x) = \sum_{j \in F_k} c_j x_j + z_k^0 \mid \sum_{j \in F_k} a_{i,j} x_j \le s_{i,k},\ i = \overline{1,m},\ x_j \le 1,\ j \in F_k,\ x_j \ge 0,\ j \in F_k\Big\}.$$

Если ограничения этой Л-задачи несовместны, то вершина v_k является вырожденной. Совместность же ограничений влечёт существование оптимального решения x^*. Для произвольного вектора $u, u \ge 0$, имеем

$$w_k(u^*, v^*) = z_k(x^*) = \sum_{j \in F_k} c_j x_j^* + z_k^0 \le g'(u),$$

так как вектор x^* удовлетворяет любому заменяющему ограничению. Пусть вектор x', $0 \le x' \le 1$, удовлетворяет заменяющему ограничению, определяемому $u = u^*$. Тогда имеют место соотношения

$$\sum_{j \in F_k} c_j x'_j + z_k^0 \le \sum_{j \in F_k} \Big(\sum_{i=1}^{m} a_{i,j} u_i^* + v_j^*\Big) x'_j + z_k^0 =$$

$$= \sum_{i=1}^{m} \sum_{j \in F_k} u_i^* a_{i,j} x'_j + \sum_{j \in F_k} v_j^* x'_j + z_k^0 \le \sum_{i=1}^{m} u_i^* s_{i,k} + \sum_{j \in F_k} v_j^* + z_k^0 = w_k(u^*, v^*).$$

Следовательно, $g'(u^*) \le w_k(u^*, v^*) = z_k(x^*) \le g'(u)$ для любого $u \ge 0$. Таким образом, $g'(u^*) = \min\limits_{u \ge 0} g'(u)$.

Итак, для нахождения самого сильного заменяющего ограничения в вершине v_k нужно решить соответствующую Л-задачу. В некоторых случаях более эффективно вычислять заменяющие ограничения периодически, а не в каждой вершине.

Ветвление. Путь p_k, соединяющий вершины v_0 и v_k дерева перебора, представляет собой последовательность рёбер или вершин. Каждое ребро пути выражает либо ограничение $x_j = 0$, либо ограничение $x_j = 1$, а каждая вершина — совокупность ограничений, соответствующих ребрам единственного пути из v_0 в эту вершину. Для представления пути p_k часто используется вектор, который также будем обозначать p_k. Компонентами вектора p_k являются индексы переменных из множества W_k. Символ j из W_k означает, что $x_j = 1$, а j^- — что $x_j = 0$. Число компонент вектора p_k указывает уровень $|W_k|$ дерева перебора.

Правило ветвления, в котором полагаем $x_j = 1 (x_j = 0)$ и в случае вырождения возвращаемся вдоль пути назад в ближайшую активную вершину, называется правосторонним (левосторонним) обходом дерева перебора. Это правило определяет правую (левую) текущую ветвь, а сам путь p_k разбивает дерево перебора на две части, одна из которых уже исследована, а вторая — нет.

Путь p_9 может быть представлен в виде вектора $(3^-, 2, 4^-)$ (рис. 5.4). Если этот путь изображает правую текущую ветвь, то часть дерева, находящаяся правее его, уже исследована. Оставшаяся часть дерева содержит шесть решений ($n = 4$).

Вектор p_k очень прост в использовании. Если переходим из вершины v_k в новую вершину по ребру $x_q = 1$, то вектор p_k заменяется на вектор (p_k, q). При возвращении вдоль пути назад помечаем чертой самую правую неотмеченную компоненту и исключаем все компоненты, находящиеся правее от нее. Если все компоненты p_k отмечены, то перебор завершен для правой текущей ветви.

Если в вершине v_9 полагаем $x_1 = 1$, то путь $(3^-, 2, 4^-)$ преобразуется в путь $(3^-, 2, 4^-, 1)$ (рис. 5.4). При возвращении вдоль пути назад из вершины v_9 в вершину v_7 вектор $(3^-, 2, 4^-)$ заменяется на вектор $(3^-, 2^-)$.

Если в вершине v_k допускается выбор ребра $x_q = 0$ перед выбором ребра $x_q = 1$, то нужно изменить запись пути p_k. Индекс j из W_k появляется в векторном представлении пути в одном из следующих четырёх видов:
j, если $j \in J_k^+$ и $x_j = 0$ еще не было рассмотрено;
$\underline{j}$, если $j \in J_k^+$ и $x_j = 0$ уже было рассмотрено;
j^-, если $j \in J_k^-$ и $x_j = 1$ еще не было рассмотрено;
$\underline{j^-}$, если $j \in J_k^-$ и $x_j = 1$ уже было рассмотрено.

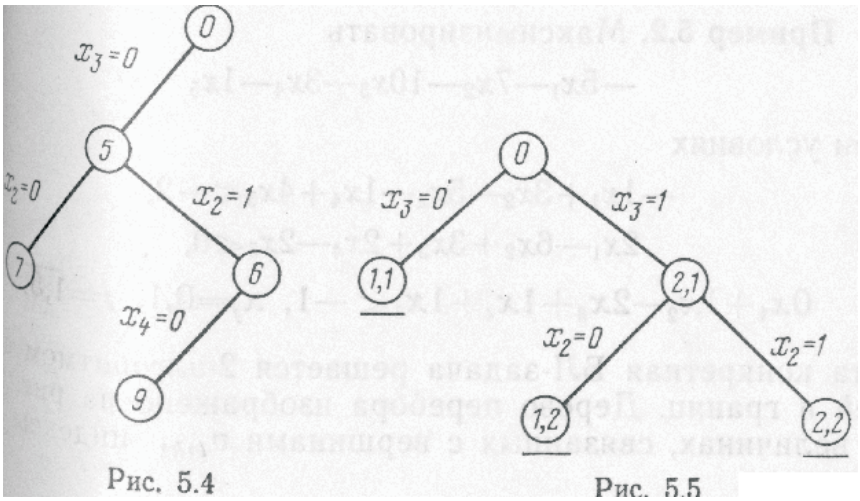


Рис. 5.4

Рис. 5.5

Вектор p_k обновляется, как и раньше, за исключением случая, когда имеет место возвращение. После удаления подчёркнутых компонент, находящихся правее самой правой неподчёркнутой компоненты, подчёркиваем последнюю компоненту и отмечаем ее чертой справа вверху, если черта отсутствует, или убираем черту, сели она присутствует.

Например, при одноветвевом обходе дерева последовательности вершин $v_{2,1}, v_{1,2}, v_{2,2}, v_{1,1}$ соответствует последовательность путей $(3), (3, 2^-), (3, \underline{2}), (\underline{3^-})$ (рис. 5.5).

Также используется ветвление, в котором осуществляется выбор более чем одного индекса за один раз для включения в множество W_k. Вместо непосредственного последователя вершины v_k можно выбирать её более далекий последователь, т. е. любую вершину на пути, выходящем из v_k. Возвращение выполняется, как и раньше. При этом нет никакой опасности пропустить оптимальное решение.

Этот выбор особенно важен для начала перебора, так как позволяет начать с вершины, отличной от v_0. Если, в частности, известно хорошее допустимое решение x', то можно начать с вершины, соответствующей, этому решению. Пусть $\{j \mid x'_j = 1\} = \{j_1, j_2, \ldots, j_l\}$. Тогда векторное представление соответствующего пути есть $(j_1, j_2, \ldots, j_l)$. Поскольку решение допустимо, то эта вершина является вырожденной, т. е. $\underline{z_0} = cx'$, и имеет место возвращение. Можно начать с вершины, не являющейся допустимой, но, по-видимому, близкой к оптимальной. В этом случае возвращение не имеет места.

Пример 5.2. Максимизировать

$$-5x_1 - 7x_2 - 10x_3 - 3x_4 - 1x_5$$

при условиях

$$-1x_1 + 3x_2 - 5x_3 - 1x_4 + 4x_5 \le -2,$$

$$2x_1 - 6x_2 + 3x_3 + 2x_4 - 2x_5 \le 0,$$

$$0x_1 + 1x_2 - 2x_3 + 1x_4 + 1x_5 \le -1,\ x_j = 0, 1,\ j = \overline{1, 5}.$$

Эта конкретная БЛ-задача решается 2-алгоритмом ветвей и границ. Дерево перебора изображено на рис. 5.5. В величинах, связанных с вершинами $v_{l,k}$, индексы l, k опущены.

В вершине v_0 вычисляем величины: $F_0 = \{1, 2, 3, 4, 5\}$, $J_0^+ = J_0^- = \varnothing, \underline{z_0} = -\infty$, $\overline{z_0} = +\infty$, $s_0 = (-2, 0, -1)$, $i_0 = 3$, $R_0 = \{1, 3, 4\}$. Для разбиения сравниваем величины $i_0(1) = 4$, $i_0(3) = 3, i_0(4) = 5$ и выбираем переменную x_3.

Левая текущая ветвь. В вершине $v_{1,1}$ вычисляем величины: $p = (3^-)$, $s = (-2, 0, -1)$, $R = \{1, 4\}$, $t_3 = 0 > s_3 = -1$. Вершина $v_{1,1}$ является вырожденной.

Правая текущая ветвь. В вершине $v_{2,1}$ вычисляем величины: $p = (3)$, $\overline{z} = -10$, $s = (3, -3, 1)$, $R = \{2, 5\}$, $i_1(2) = 0$, $i_1(5) = 2$. Для разбиения выбираем переменную x_2.

На левой текущей ветви нет активных вершин. Переносим её начало в вершину $v_{2,1}$, полагаем $x_2 = 0$ и получаем вершину $v_{1,2}$. В вершине $v_{1,2}$ вычисляем величины: $p = (3, 2^-)$, $s = (3, -3, 1)$, $R = \{5\}$, $t_2 = -20 > s_3 = -3$. Вершина $v_{1,2}$ является вырожденной.

Правая текущая ветвь. В вершине $v_{2,2}$ вычисляем величины: $p = (3, 2)$, $\underline{z} = \overline{z} = -17$, $\underline{z_0} = -17$, $s = (0, 3, 0)$. Вершина $v_{2,2}$ является вырожденной и даёт допустимое решение.

На обеих текущих ветвях нет ни одной активной вершины. Перебор завершен. Задача решена. Оптимальное решение: $x_1 = 0$, $x_2 = 1$, $x_3 = 1$, $x_4 = 0$, $x_5 = 0$, $x_0 = z^* = -17$.

Тесты для неявного перебора. Приведем ряд тестов, которые сокращают перебор за счёт дополнительных вычислений. Все тесты иллюстрируются конкретными примерами, а их доказательства становятся очевидными после некоторого размышления.

Дополним систему ограничений БЛ-задачи в вершине v_k неравенством $\sum\limits_{j \in F_k} c_j x_j + z_k^0 \ge \underline{z_0}$, которое выполняется для любого оптимального решения. Тогда получим БЛ-задачу

$$\max\Big\{ \sum_{j \in F_k} -a_{0,j} x_j + z_k^0 \mid \sum_{j \in F_k} a_{i,j} x_j \le s_{i,k},\ i = \overline{0, m},\ x_j = 0, 1,\ j \in F_k \Big\},$$

где $a_{0,j} = -c_j$, $j \in F_k$, $s_{0,k} = z_k^0 - \underline{z_0}$.

Считаем, что вершина v_k является активной, т. е. выполняются условия:
а) $s_{0,k} \ge 0$; б) $Q_k = \{i \mid s_{i,k} < 0\} \ne \varnothing$; в) $t_i = \sum\limits_{j \in F_k} \min\{0, a_{i,j}\} \le s_{i,k}$, $i = \overline{0, m}$.

Тест 1. Если для некоторых индексов i и p выполняется неравенство $\sum\limits_{j\in F_k}\min\{0,a_{i,j}\}+|a_{i,p}|>s_{i,k}$ то в любом допустимом расширении множества W_k из $a_{i,p}>0$ ($a_{i,p}<0$) следует $x_p=0$ ($x_p=1$) ($i=\overline{0,m},\ p\in F_k$).

Например, из ограничения $-3x_1+2x_2-5x_3+6x_4\le -4$ следует, что $x_3=1, x_4=0$.

В тестах 2-4 используются нижняя граница l и верхняя граница u для числа переменных, принимающих значение 1. Это означает, что в любом допустимом расширении множества W_k имеет место неравенство $l\le\sum\limits_{j\in F_k}x_j\le u$.

Для каждого ограничения i, $i=\overline{0,m}$, упорядочиваем переменные так, что коэффициенты при них образуют неубывающую последовательность $a_{i,j_1},a_{imj_2},\dots,a_{i,j_q}(q=|F_k|)$, т. е. из $s<t$ следует $a_{i,j_s}\le a_{i,j_t}$. Пусть

$$t_i(0)=0, t_i(p)=\sum_{s=1}^{p}a_{i,j_s},\ p=\overline{1,q}.$$

Величина $t_j(p)$ представляет собой наименьшую из сумм, даваемых p переменными, равными 1.

Для каждого $i\in Q_k$ определяем величину l_i из условия $t_i(l_i)\le s_{i,k}<t_i\ (l_i-1)$. Следовательно, $1\le l_i\le q$. Полагаем $l=\max\limits_{i\in Q_k}l_i$. Для каждого ограничения $i, i=\overline{0,m}$, вычисляем величину u_i из условий: $t_i(u_i)\le s_{i,k}<t_i(u_i+1)$, если $t_i(q)>s_{i,k}$, $u_i=+\infty$, если $t_i(q)\le s_{i,k}$. Полагаем $u=\min\limits_{i=0}u_i$.

Пример 5.3. Рассмотрим систему ограничений

$$3x_1+x_2+4x_3+x_4\le 11,$$

$$-x_1-x_2+2x_3-3x_4\le -4,$$

$$3x_1-4x_2-2x_3-+6x_4\le -1,\ x_j=0,1,\ j=\overline{1,4}.$$

Вычисляем величины: $l_1=2$, $l_2=1$, $l=2$, $u_0=+\infty$, $u_1=3$, $u_2=3$, $u=3$.

Тест 2. Если $l>u$, то вершина v_k является вырожденной (система ограничений несовместна). Для $l=|F_k|$ имеем $x_j=1, j\in F_k$. Иными словами, если $F_k=\{j_1,j_2,\dots,j_q\}$, то путь p_k заменяется на путь $(p_k,\underline{j_1},\underline{j_2},\dots,\underline{j_q})$.

Тест 3. Через $t_i(l,u)=\min\limits_{p=l}^{u}t_i(p)$, $i=\overline{0,m}$, обозначаем наименьшую из сумм, которые могут быть получены при использовании не меньше i и не больше u свободных переменных, равных 1. Предположим, что вершина v_k не является вырожденной по тесту 2 и что коэффициент $a_{i,p}$ не входит в качестве слагаемого в величину $t_i(l,u)$. Если $t_i(l-1,u-1)>s_{i,k}-a_{i,p}$, то $x_p=0$.

Для примера 5.3. имеем $t_2(1,2)=-6>s_2-a_{2,4}=-7$. Поэтому $x_4=0$.

Тест 4. Предположим, что вершина v_k не является вырожденной по тесту 2 и что коэффициент $a_{i,p}$ входит в качестве слагаемого в величину $t_i(l,u)$. Если $t_i(l+1,u+1)>s_{i,k}+a_{i,p}$, то $x_p=1$.

Для примера 5.3 имеем $t_1(3,4)=-5>s_1+a_{1,4}=-7$. Поэтому $x_4=1$. Тесты 3 и 4 показывают, что система ограничений из примера 5.3 не имеет бинарных решений. В этом также можно убедиться, прибавив ко второму неравенству первое, умноженное на два.

В тестах 5 и 6 пытаемся определить, существует ли ограничение $i, i=\overline{1,m}$, которое может быть удовлетворено только за счёт нарушения неравенства $i=0$.

Тест 5. Для ограничения i и индекса $p, a_{i,p}>s_{i,k}$, вычисляем $a_0^*=\min\limits_{a_{i,j}<0, j\in(F_k/p)}a_{0,j}$ ($i=\overline{1,m},\ p\in F_k$). Если $a_{0,p}+a_0^*>s_{0,k}$, то $x_p=0$.

Пример 5.4. Рассмотрим систему ограничений

$$3x_1+4x_2+7x_3+x_4+8x_5\le 9,$$

$$-x_1-2x_2-x_3+x_4-3x_5\le -2,$$

$$-2x_1-3x_2-2x_3+3x_4-x_5\le -4,$$

$$x_1-x_2-3x_3+5x_4-2x_5\le -4,\ x_j=0,1,\ j=\overline{1,5}.$$

В примере 5.4 для $i=2, p=2$ имеем $a_0^*=7, a_{0,2}+a_0^*=11>s_0=9$, поэтому $x_2=0$.

Тест 6. Для $i \in Q_k$ вычисляем $r_i = \min\limits_{a_{i,j}<0, j \in F_k} \{s_{i,k} a_{0,j} / a_{i,j}\}$. Если $r_i > s_{0,k}$, то вершина v_k является вырожденной.

Для примера 5.4 имеем $r_3 = 28/3 > s_0 = 9$. Поэтому не существует допустимых расширений.

Заметим, что тест 6 является единственным тестом, в котором выполняются операции умножения и деления. Этот тест целесообразно использовать, если он исключает значительную часть дерева перебора.

5.5 КОНИЧЕСКАЯ РЕЛАКСАЦИЯ

Пусть B — базисная матрица оптимального решения Л-задачи $\max\{cx \mid Ax = b,\ x \geq 0\}$, N — подматрица матрицы A, соответствующая небазисным переменным,

$$z(x_N) = c_B B^{-1} b - (c_B B^{-1} N - c_N) x_N.$$

В вершине v_0 дерева перебора рассматриваем ЦЛ-задачу

$$\max\{z(x_N) \mid x_N \in S_0 = \{x_N \mid x_B = B^{-1}b - B^{-1}Nx_N,\ x_B \geq 0,\ x_B \equiv 0,\ x_N \geq 0,\ x_N \equiv 0\}\}.$$

В качестве ее релаксированной задачи выступает ЦЛК-задача

$$\max\{z(x_N) \mid x_N \in T_0 = \{x_N \mid B^{-1}Nx_N \equiv B^{-1}b,\ x_N \geq 0,\ x_N \equiv 0\}\}.$$

Пусть $x_N = (x_1, x_2, \ldots, x_n)$. Ослабленная задача в вершине v_k имеет вид

$$\max\{z(x_N) \mid x_N \in T_k = T_0 \bigcap \{x_N \mid x_N \geq l_k\}\},$$

где $l_k = (l_{1,k}, l_{2,k}, \ldots, l_{n,k})$ — вектор неотрицательных целых чисел. Замена переменных $x_N = x'_N + l_k$ преобразует эту задачу в обычную ЦЛК-задачу.

Покрытие. Вектор l_k определяет покрытие

$$S_{k,j},\ j = \overline{0,n},$$

где $S_{k,0} = S_k \bigcap \{x_N \mid x_N = l_k\}$, $S_{k,j} = S_k \bigcap \{x_N \mid x_j \geq l_{j,k} + 1\}$, $j = \overline{0,n}$, $\bigcup\limits_{j=1}^{n} S_{k,j} = S_k$.

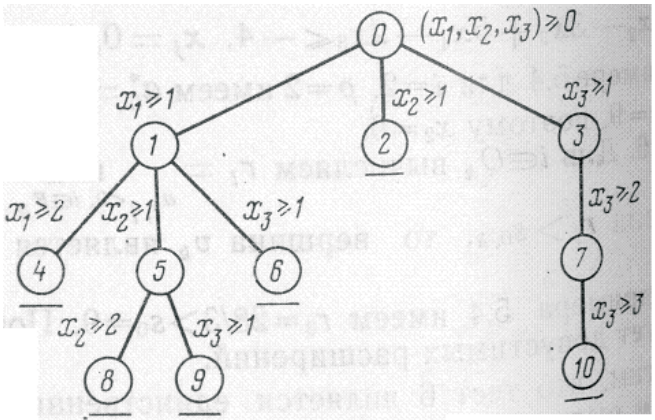


Рис. 5.6

Хотя это покрытие не является разбиением, можно доказать, что нет необходимости рассматривать задачи в вершинах $v_{k,j}, j = \overline{0, j(k) - 1}$, где

$$j(k) = \begin{cases} 1,\ R_k = \varnothing, \\ \max\{j \mid j \in R_k\},\ R_k \neq \varnothing, \end{cases} \quad R_k = \{j \mid l_{j,k} > 0\}.$$

Для $j = 0$ либо $s_{k,0} = \varnothing$, либо $s_{k,0} = \{l_k\}$. В первом случае нет ветвления к вершине $v_{k,0}$, а во втором в силу монотонности целевой функции вершина v_k является вырожденной. Можно показать, что для $j = \overline{1, j(k)-1}$ множества $S_{k,j}$ будут рассмотрены на дереве.

Возможное дерево изображено на рис. 5.6. Заметим, что при ветвлении из вершины v_3 уже не рассматриваем $x_1 \geq 1$, поскольку $S_3 \bigcap \{x_N \mid x_1 \geq 1\} = S_6$.

5.6 РАСПАРАЛЛЕЛИВАНИЕ АЛГОРИТМОВ ДИНАМИЧЕСКОГО ПРОГРАММИРОВАНИЯ

На простом примере покажем возможность распараллеливания вычислений при выполнении алгоритмов динамического программирования. Рассмотрим задачу о ранце

$$\max\Big\{\sum_{j=1}^{n} c_j x_j \mid \sum_{j=1}^{n} a_j x_j \leq b,\ x_j \in Z^+,\ j = \overline{1,n}\Big\}.$$

Считаем, что заданные величины a_j, b, c_j удовлетворяют условиям $a_k \neq a_l$, $k \neq l$, $a_j \in (Z^+ \backslash 0)$, $b \in Z^+$, $c_j \geq 0$ $(j = \overline{1,n},\ k = \overline{1,n},\ l = \overline{1,n})$.

Вместо решения одной конкретной задачи оптимизации (величины b и n зафиксированы) будем решать семейство похожих на нее задач

$$\max\Big\{\sum_{j=1}^{k} c_j x_j \mid \sum_{j=1}^{k} a_j x_j \leq y,\ x_j \in Z^+,\ j = \overline{1,k}\Big\},$$

$$k = \overline{1,n},\ y = 0, 1, 2, \ldots, b.$$

Это семейство состоит из $(b+1)n$ задач.

Для целочисленного аргумента y, $y = 0, 1, 2, \ldots, b$, определим функции

$$\psi_k(y) = \max\Big\{\sum_{j=1}^{k} c_j x_j \mid \sum_{j=1}^{k} a_j x_j \leq y,\ x_j \in Z^+,\ j = \overline{1,k}\Big\},\ k = \overline{1,n}.$$

Значение функции $\psi_k(y)$ есть оптимальное значение целевой функции задачи с фиксированными k и y. Очевидно, что $\psi_k(0) = 0, k = \overline{1,n}$, и $\psi_k(y) = c_1 \lfloor y/a_1 \rfloor$. Для $k = \overline{2,n}$ имеют место рекуррентные соотношения $\psi_k(y) = \max\{\psi_{k-1}(y),\ \psi_k(y - a_k) + c_k\}$. Считаем, что $\psi_k(y) = -\infty$ для $y < 0$, $k = \overline{2,n}$.

Вывод рекуррентного соотношения опирается на факт, что в оптимальном решении либо, $x_k = 0$, либо $x_k \geq 1$.

При помощи функций $i_k(y)$ вычисляется оптимальное решение для фиксированных k и y. Пусть $j_1 = i_k(y)$. Рассмотрим индексы $j_2 = i_k(y - a_{j_1}, j_3 = i_k(y - a_{j_1} - a_{j_2}), \ldots, 0 = i_k(y - a_{j_1} - a_{j_2} - a_{j_3} - \ldots)$. Сколько раз индекс j появляется в ряду $j_1, j_2, j_3, \ldots, 0$, таково значение x_j^* в оптимальном решении.

Отметим, что на шаге l нет необходимости хранить информацию о шагах $k \leq l-2$. Это позволяет значительно уменьшить память, необходимую для вычисления функций $\psi_n(y), i_n(y)$.

Теперь покажем, как на p одновременно работающих процессорах выполнить описанный алгоритм решения задачи о ранце.

Разбиваем множество $S = (1, 2, \ldots, n)$ на P подмножеств $S_l = \{(l-1)n' + 1, (l-1)n' + 2, \ldots, ln'\}$, $l = \overline{1,p}$ (здесь для простоты считаем, что n делится на p, а $n' = n/p$). Это разбиение порождает разбиение строк a и c на p частей. Для каждого подмножества S_l вычисляем функции $\psi_{s_l}(y), i_{s_l}(y)$ от целочисленного аргумента y, $y = 0, 1, 2, \ldots, b$ $(l = \overline{l,p})$. Решаем новую задачу

$$\max\Big\{\sum_{l=1}^{p} \psi_{s_l}(y_l) \mid \sum_{l=1}^{p} y_l \leq b,\ y_l \in Z^+,\ l = \overline{1,p}\Big\}.$$

Эта задача решается при помощи рекуррентных соотношений

$$\overline{\psi_k}(y) = \max_{s=0}^{y}(\overline{\psi_{k-1}}(s) + \psi_{s_k}(y-s)), \overline{y_k}(y) = \overline{s},$$

где $\overline{s}$ — наименьшее значение, для которого достигается максимум ($k = \overline{1,p};\ y = 0,1,2,\ldots,b$).

При вычислении функций $\overline{\psi_k}(y), y = 0,1,2,\ldots,b$, следует так выбрать точки $y_1, y_2, \ldots, y_p$ в интервале $[0,b]$ чтобы количество операций в каждом из p подинтервалов $[y_i, y_{i+1}]$ было примерно одинаковым.

Таблица 5.1

y / ψ_k, i_k	0	1	2	3	4	5	6	7	8	9	10
$\psi_1(y)$	0	0	1	1	2	2	3	3	4	4	5
$i_1(y)$	0	0	1	1	1	1	1	1	1	1	1
$\psi_2(y)$	0	0	1	3	3	4	6	6	7	9	9
$i_2(y)$	0	0	1	2	2	2	2	2	2	2	2
$\psi_3(y)$	0	0	1	3	5	5	6	8	10	10	11
$i_3(y)$	0	0	1	2	3	3	3	3	3	3	3
$\psi_4(y)$	0	0	1	3	5	5	6	9	10	10	12
$i_4(y)$	0	0	1	2	3	3	3	4	3	4	4

Таблица 5.2

y / ψ_{s_k}, i_{s_k}	0	1	2	3	4	5	6	7	8	9	10
$\psi_1(y)$	0	0	1	1	2	2	3	3	4	4	5
$i_1(y)$	0	0	1	1	1	1	1	1	1	1	1
$\psi_{1,3}(y)$	0	0	1	1	5	5	6	6	10	10	11
$i_{1,3}(y)$	0	0	1	1	3	3	3	3	3	3	3
$\psi_2(y)$	0	0	0	3	3	3	3	6	6	6	9
$i_2(y)$	0	0	0	2	2	2	2	2	2	2	2
$\psi_{2,4}(y)$	0	0	0	3	3	3	6	9	9	9	12
$i_{2,4}(y)$	0	0	0	2	2	2	2	4	4	4	4
$\psi_{1,3;2,4}(y)$	0	0	1	3	5	5	6	9	10	10	12
$y_{1,3;2,4}(y)$	0	0	2	0	4	4	0	0	8	2	0
$i_{1,3;2,4}(y)$	0	0	1	2	3	3	2	4	3	4	4

Таким образом, в рассмотренном параллельном алгоритме счёт ускоряется примерно в p раз. Табл. 5.1 и 5.2 иллюстрируют процесс решения конкретной задачу о ранце соответственно на одном и двух процессорах.

Пример 5.5. Максимизировать $1x_1 + 3x_2 + 5x_3 + 9x_4$ при условиях

$$2x_1 + 3x_2 + 4x_3 + 7x_4 \le 10,\ x_j \in Z^+,\ j = \overline{1,4}.$$

Функции $i_4(y)$ и $i_{1,3;2,4}$ отличаются для значения $y = 6$. Это объясняется тем, что для рассматриваемого значения y существуют два оптимальных решения. Из табл. 5.1 и 5.2 видно, что для рассматриваемой конкретной задачи о ранце $\psi_4(10) = \psi_{1,3;2,4}(10) = 12,\ i_4(10) = i_{1,3;2,4} = 4,\ i_4(3) = i_{1,3;2,4} = 2$.

Итак, оптимальное решение есть $x_2 = 1, x_4 = 1$, а оптимальное значение целевой функции равно 12.

5.7 ПРИБЛИЖЕННЫЕ p-АЛГОРИТМЫ

Многие реальные задачи слишком велики для решения их точными алгоритмами. Единственная разумная альтернатива для таких задач — использование приближённых алгоритмов, которые учитывают специфические свойства рассматриваемых задач и привлекают хитроумные, близкие к искусству приемы. Такие алгоритмы, в которых используются нестрогие интуитивные соображения, называют эвристическими. Эти алгоритмы дают экономные способы получения хороших начальных решений для алгоритмов перебора.

Под окрестностью $N(x)$ целочисленной точки x будем понимать множество точек, в некотором смысле близких к x. Окрестность, которая часто используется, представляет собой множество целочисленных векторов, отличающихся от x не более чем заданным числом компонент. Для БЛ- и ЦЛ-задач

точка $x_0 \in S$ называется локальным максимумом относительно окрестности $N(x)$, если $cx_0 \geq cx$ для всех $x \in S \bigcap N(x_0)$. Большинство приближённых алгоритмов находят множество локальных оптимумов.

Общий приближённый p-алгоритм

Шаг 1 (начало, $l = \overline{1,p}$). Выбираем q окрестностей $N_{l,1}(x), N_{l,2}(x), \ldots, N_{l,q}(x)$ таких, что $N_{l,t}(x) \not\subset N_{l,s}(x)$ для $t > s$. Окрестности $N_{l,i}(x)$, $i = \overline{2,q}$, используются для выхода из локального оптимума относительно окрестности $N_{l,1}(x)$. Находим точку $x_{l,0}, x_{l,0} \in S$, полагаем $\underline{z_{l,0}} = cx_{l,0}$ и переходим к шагу 2.

Шаг 2 (решение задач для $l = \overline{1,p}$. Решаем задачу $\max\{cx \mid x \in S \bigcap N_{l,1}(x_{l,0})\}$. Переходим к шагу 3.

Шаг 3 (выполняется для $l = \overline{1,p}$). Если целевая функция решенной задачи не ограничена на множестве её допустимых решений, то целевая функция не ограничена и для исходной задачи. Если $x_{l,1}$ — оптимальное решение решённой задачи и $cx_{l,1} > cx_{l,0}$, то полагаем $x_{l,0} = x_{l,1}$, $\underline{z_{l,0}} = cx_{l,1}$ и переходим к шагу 2. Если же $cx_{l,1} = cx_{l,0}$, то полагаем $i = 2$ и переходим к шагу 4.

Шаг 4 (выполняется для $l = \overline{1,p}$). Пытаемся найти решение x', для которого $cx' > cx_{l,0}$, $x' \in S \bigcap N_{l,i}x_{l,0}$. Если такое решение существует, то полагаем $x_{l,0} = x'$ и переходим к шагу 2. Иначе для $i < q$ заменяем i на $i + 1$ и повторяем шаг 4. Если $i = q$, то переходим к шагу 5.

Шаг 5 (выполняется для $l = \overline{1,p}$). Решение, дающее $\underline{z_{l,0}}$ является наилучшим из известных, полученных на ветви l. Затем либо выбираем другую точку $\overline{x_{l,0}} \in S$ и переходим к шагу 2, либо ветвь l заканчивает свою работу.

Шаг 6 (окончание). Все ветви параллельного алгоритма закончили свою работу. Решение, которое даёт $\underline{z}_0 = \max\limits_{l=1}^{p} \underline{z}_{l,0}$, является наилучшим приближённым решением задачи максимизации.

Эффективность приближённого p-алгоритма зависит от выбора окрестностей, которые определяют трудоёмкость решения задач максимизации на шагах 2 и 4. Для некоторых задач бывает нелегко найти начальную точку на шаге 1. В этом случае следует более детально развить шаг 1. Ясно, что локальный оптимум, получаемый на каждой ветви параллельного алгоритма, зависит от начальной точки. Часто хорошие начальные точки могут быть получены в окрестности оптимального решения Л-задачи.

УПРАЖНЕНИЯ

5.1. Использовав четырёхветвевой обход дерева перебора, установить, что уравнение $2x_1 + 2x_2 + \ldots + 2x_7 = 7$ не имеет неотрицательных целочисленных решений.

5.2. Доказать следующий факт. Если алгоритм ветвей и границ используется для установления того, что для нечётного натурального числа n уравнение $\sum\limits_{j=1}^{n} 2x_j = n$ не имеет неотрицательных целочисленных решений (это очевидно), то он приводит к экспоненциальному росту вершин дерева в зависимости от n.

5.3. Использовав векторное представление пути, для решения примера 5.1 построить 2-алгоритм, в котором исследуется только часть дерева перебора.

5.4. Построить общий p-алгоритм ветвей и границ для решения задачи оптимизации $\min\{z(x) \mid x \in S_0\}$ (см. § 5.2).

5.5. Решить p-алгоритмом ветвей и границ конкретную СЦЛ-задачу ($p = 1, 2$). Максимизировать $-4x_1 - 4x_2 - 2y_1 - 3y_2$ при условиях

$$x_1 - 3x_2 + y_1 - 2y_2 \leq -2,$$

$$-x_1 - 3x_2 - y_1 - y_2 \leq -3,$$

$$x_j \geq 0 \wedge x_j \in Z,\ j = \overline{1,2},\ y_j \geq 0,\ j = \overline{1,2}.$$

5.6. Решить p-алгоритмом неявного перебора конкретную БЛ-задачу ($p = 1, 2$). Минимизировать $5x_1 + 7x_2 + 10x_3 + 3x_4 + 1x_5$ при условиях

$$1x_1 - 3x_2 + 5x_3 + 1x_4 - 4x_5 \geq 2,$$

$$-2x_1 + 6x_2 - 3x_3 - 2x_4 + 2x_5 \geq 0,$$

$$0x_1 - 1x_2 + 2x_3 - 1x_4 - 1x_5 \geq 1,$$

$$x_j = 0, 1,\ j = \overline{1,5}.$$

5.7. Использовав коническую релаксацию, решить p-алгоритмом ветвей и границ конкретную ЦЛ-задачу ($p = 1, 2$). Максимизировать $-7x_1 - 3x_2 - 4x_3$ при условиях

$$x_1 + 2x_2 + 3x_3 - x_4 = 8,$$

$$3x_1 + x_2 + x_3 - x_5 = 5,$$

$$x_j \geq 0 \wedge x_j \in Z,\ j = \overline{1,5}.$$

5.8. Рассмотрим задачу о ранце. Максимизировать $11x_1 + 9x_2 + 5x_3 + 3x_4$ при условиях

$$6x_1 + 5x_2 + 3x_3 + 2x_4 \leq 45,$$

$$x_j \geq 0 \wedge x_j \in Z,\ j = \overline{1,4}.$$

Решить эту задачу двумя способами: последовательным и параллельным алгоритмами.

Глава 6

ПАРАЛЛЕЛЬНЫЕ АЛГОРИТМЫ СЕКУЩИХ ПЛОСКОСТЕЙ

В данной главе для решения целочисленной и смешанной целочисленной линейных задач различными способами строится линейная задача, оптимальное решение которой удовлетворяет условиям целочисленности. Построение эффективных сечений и ускорение сходимости являются основными вопросами в методе секущих плоскостей. Эта часть целочисленной оптимизации богата теоретическими результатами, связанными с другими разделами математики.

Построение параллельных алгоритмов секущих плоскостей основано на весьма разнообразных приемах распараллеливания действий над матрицами. Некоторые алгоритмы, рассмотренные здесь (см., например, § 6.4 и 6.5), служат вспомогательными средствами для распараллеливания более сложных алгоритмов.

6.1 ВЫПОЛНЕНИЕ p-УМНОЖЕНИЯ ДВУХ МАТРИЦ

Пусть даны $m \times k$-матрица A, $k \times n$-матрица B и p параллельно работающих процессоров ($k \gg p$, $m \gg p$, $n \gg p$). Элементы $c_{i,j}$ матрицы $C = AB$, являющейся произведением матриц A и B, вычисляются по формуле

$$c_{i,j} = \sum_{s=1}^{k} a_{i,s} b_{s,j},\ i = \overline{1,m},\ j = \overline{1,n}.$$

Для простоты считаем, что каждое из чисел m и n делится на p. Разобьем матрицу A на p горизонтальных полос одинаковой ширины (каждая полоса состоит из $m' = m/p$ строк), отнеся в полосу l ее строки с номерами $(l-1)m'+1, (l-1)m'+2, \ldots, lm'$ $(l = \overline{1,p})$. Аналогично разобьем матрицы B и C на p вертикальных полос одинаковой ширины (каждая полоса состоит из $n' = n/p$ столбцов), отнеся в полосу l их столбцы с номерами $(l-1)n'+1, (l-1)n'+2, \ldots, ln'(l = \overline{1,p})$ (рис. 6.1).

На процессоре l находятся горизонтальная полоса l матрицы A, вертикальная полоса l матрицы B, а также вертикальная полоса l матрицы C, элементы которой требуется вычислить. Таким образом, на процессоре l имеются данные только для вычисления, элементов клетки $C_{l,l} = (c_{i,j})((l-1)m'+1 \le i \le lm', (l-1)n'+1 \le j \le ln')$. Для вычисления клетки $C_{i,j}, i \ne j$, нужно передать горизонтальную полосу i матрицы A с процессора i на процессор l, а для вычисления клетки $C_{i,j}$ следует переслать горизонтальную полосу l матрицы A с процессора l на процессор j. Таким образом, для этого необходимо выполнить операции обмена.

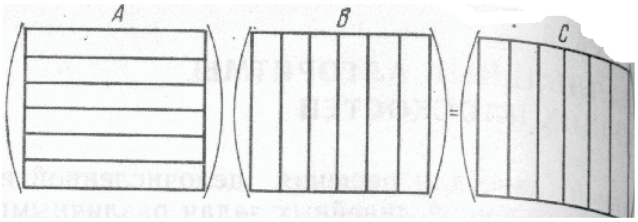


Рис. 6.1

Следует отметить, что время выполнения операций обмена мало по сравнению со временем выполнения всех необходимых арифметических операций. Поэтому рассмотренное p-умножение двух матриц примерно в p раз ускоряет счёт [19].

На рис. 6.2 изображена схема l-ветви параллельного алгоритма. На процессоре l выполняется l-я ветвь.

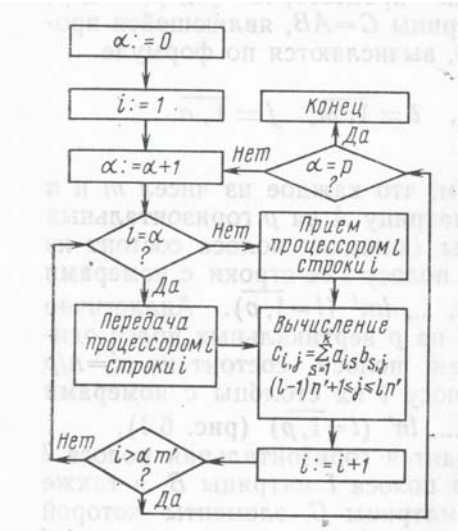


Рис. 6.2

148

6.2 СЕКУЩИЕ ПЛОСКОСТИ

В алгоритмах секущих плоскостей для решения ЦЛ- и СЦЛ-задач строится такая Л-задача оптимизации, что её оптимальное решение удовлетворяет условиям целочисленности. Построение указанной Л-задачи осуществляется путём добавления к ограничениям исходной задачи линейных неравенств, которым удовлетворяют все допустимые решения исходной задачи. В целочисленной оптимизации эти линейные неравенства называются сечениями, а соответствующие им гиперплоскости — секущими плоскостями.

Считаем, что данные для ЦЛ- и СЦЛ-задач целочисленные. Если множество допустимых решений исходной задачи не пусто, то его выпуклая оболочка может быть представлена как пересечение конечного числа подпространств.

Сечение для целочисленной линейной задачи. Пусть имеется базисное представление Л-задачи

$$x_{B_i} = y_{i,0} - \sum_{j \in J} y_{i,j} x_j, \ i = \overline{0,m},$$

где J — множество индексов небазисных переменных (x_{B_0} — целевая функция). Умножая уравнение i на действительное число $h, h \neq 0$, получаем

$$h x_{B_i} + \sum_{j \in J} h y_{i,j} x_j = h y_{i,0}.$$

Для неотрицательных переменных x_{B_i} и x_j выполняются неравенства

$$\lfloor h \rfloor x_{B_i} \le h x_{B_i}, \lfloor h y_{i,j} \rfloor x_j \le h y_{i,j} x_j,\ j \in J.$$

Поэтому справедливы неравенства

$$\lfloor h \rfloor x_{B_i} + \sum_{j \in J} \lfloor h y_{i,j} \rfloor x_j \le h y_{i,0},$$

$$\lfloor h \rfloor x_{B_i} + \sum_{j \in J} \lfloor h y_{i,j} \rfloor x_j \le \lfloor h y_{i,0} \rfloor$$

(второе неравенство следует из первого, так как левая часть первого неравенства целочисленная в силу целочисленности переменных x_{B_i} и x_j). Вычитая последнее неравенство из исходного уравнения i, теперь уже умноженного на $\lfloor h \rfloor$ получаем

$$\sum_{j \in J} (\lfloor h \rfloor y_{i,j} - \lfloor h y_{i,j} \rfloor) x_j \ge \lfloor h \rfloor y_{i,0} - \lfloor h y_{i,0} \rfloor. \tag{6.1}$$

Сечение (6.1) получено только из одного уравнения i. Выбирая определённым образом значения параметра h, получаем различные сечения. Эти сечения используются в алгоритмах, описанных ниже.

Сечение для смешанной целочисленной линейной задачи. Рассмотрим i-е уравнение базисного представления Л-задачи

$$x_{B_i} = y_{i,0} - \sum_{j \in J_1} y_{i,j} x_j - - - \sum_{j \in J_2} y_{i,j} x_j,\ J_1 \bigcup J_2 = J,$$

для которого $fi, 0 > 0$ и переменная x_{B_i} целочисленна. Здесь J_1 и J_2 — множества индексов соответственно целочисленных и непрерывных небазисных переменных, а $f_{i,j} = y_{i,j} - \lfloor y_{i,j} \rfloor$ — дробная часть числа $y_{i,j}$. Беря дробные части, получаем

$$\sum_{j \in J_1} f_{i,j} x_j + \sum_{j \in J_2} y_{i,j} x_j - f_{i,0} = \lfloor y_{i,0} \rfloor - \sum_{j \in J_1} \lfloor y_{i,j} \rfloor x_j - x_{B_i}.$$

Для допустимых решений СЦЛ-задачи правая часть этого равенства целочисленная, следовательно, такова же и левая часть.

Таким образом, должно быть либо

$$\sum_{j \in J_1} f_{i,j} x_j + \sum_{j \in J_2} y_{i,j} x_j - f_{i,0} \ge 0,$$

либо

$$\sum_{j \in J_1} f_{i,j} x_j + \sum_{j \in J_2} y_{i,j} x_j - f_{i,0} \le -1.$$

Пусть

$$J_2^- = \{j \mid y_{i,j} < 0, j \in J_2\},\ J_2^+ = \{j \mid y_{i,j} \ge 0,\ j \in J_2\}.$$

Тогда первое неравенство даёт

$$\sum_{j \in J_1} f_{i,j} x_j + \sum_{j \in J_2^+} y_{i,j} x_j \ge f_{i,0}. \tag{6.2}$$

Из второго неравенства следует

$$\sum_{j \in J_2^-} y_{i,j} x_j \le -1 + f_{i,0}. \tag{6.3}$$

Умножая неравенство (6.3) на отрицательную величину $f_{i,0}/(-1 + f_{i,0})$, получаем

$$-\sum_{j \in J_2^-} \frac{f_{i,0} y_{i,j}}{1 - f_{i,0}} x_j \ge f_{i,0}. \tag{6.4}$$

Поскольку левые части неравенств (6.2) и (6.4) неотрицательные и одно из них всегда имеет место, то получаем

$$\sum_{j\in J_1} f_{i,j}x_j + \sum_{j\in J_2^+} y_{i,j}x_j - \sum_{j\in J_2^-} \frac{f_{i,0}y_{i,j}}{1-f_{i,0}}x_j \geq f_{i,0}. \tag{6.5}$$

Неравенство (6.5) является сечением для СЦЛ-задачи. Слабая переменная, превращающая это неравенство в равенство, не обязана быть целочисленной. Для ЦЛ-задачи $J_2^- = J_2^+ = \varnothing$, и неравенство (6.5) превращается в сечение для алгоритма целых форм.

Аналогичные рассуждения дают более сильное сечение для СЦЛ-задачи

$$\sum_{j\in J_1^+} f_{i,j}x_j + \sum_{j\in J_1^-} \frac{f_{i,0}(1-f_{i,j})}{1-f_{i,0}}x_j + \sum_{j\in J_2^+} y_{i,j}x_j - \sum_{j\in J_2^-} \frac{f_{i,0}y_{i,j}}{1-f_{i,0}}x_j \geq f_{i,0},$$

где $J_1^+ = \{j \mid f_{i,j} \leq f_{i,0}, j\in J_1\}$, $J_1^- = J_1/J_1^+$(очевидно, что $f_{i,j} \leq f_{i,0} \longleftrightarrow f_{i,j} \leq \frac{f_{i,0}(1-f_{i,j})}{1-f_{i,0}}$).

Конечность алгоритма, в котором используется сечение (6.5), может быть показана, если и только если значения целевой функции целочисленные.

Пересекающее сечение. Рассмотрим замкнутое ограниченное выпуклое множество C, внутренность которого содержит оптимальное базисное решение x^0 линейной релаксации ЦЛ-задачи и не содержит допустимых целочисленных точек. Из точки x^0 проведем n крайних лучей. Эти крайние лучи задаются небазисными столбцами симплекс-таблицы (нет вырожденности). Рассматриваемые крайние лучи пересекают границу множества C в n различных точках, которые, в свою очередь однозначно определяют гиперплоскость.

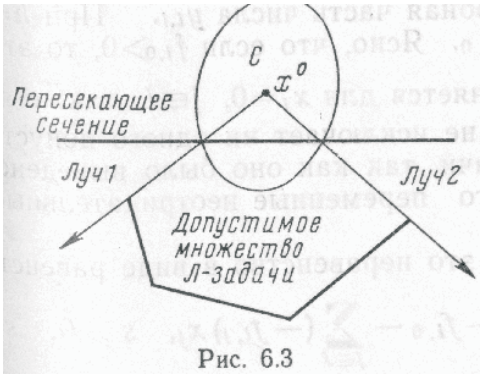


Рис. 6.3

Можно показать, что определённая таким образов гиперплоскость является секущей плоскостью. Это сечение показано на рис. 6.3 [65].

Заметим, что все сечения, рассмотренные в этом параграфе, отсекают текущее решение. Можно получить различные сечения для ЦЛ- и СЦЛ- задач, рассматривая их групповые релаксации (см., например, [58]).

6.3 ПРЯМЫЕ И ДВОЙСТВЕННЫЕ АЛГОРИТМЫ

Каждый из алгоритмов, рассматриваемых здесь, представляет собой последовательность симплекс-итераций. Как хорошо известно, выполнение одной симплекс-итерации равносильно умножению текущей матрицы на матрицу, отличающуюся от единичной одной строкой (одним столбцом). Это обстоятельство и p-умножение двух матриц позволяют строить p-алгоритмы секущих плоскостей, в которых обмен между параллельными ветвями мал. Поэтому счёт ускоряется примерно в p раз.

Опустив вычислительные детали, рассмотрим три подхода к решению ЦЛ-задачи $\max\{cx \mid Ax \leq b,\ x \geq 0,\ x \equiv 0\}$ с целочисленными данными A, b, c [57, 58].

Алгоритм целых форм. Для алгоритма целых форм в неравенстве (6.1) считаем h целым числом. Это даёт сечение

$$\sum_{j\in J}(hf_{i,j} - \lfloor hf_{i,j}\rfloor)x_j \geq hf_{i,0} - \lfloor hf_{i,0}\rfloor,$$

где $f_{i,j}$ — дробная часть числа $y_{i,j}$. При $h = 1$ получаем $\sum_{j\in J} f_{i,j}x_j \geq f_{i,0}$. Ясно, что если $f_{i,0} > 0$, то это неравенство не выполняется для $x_j = 0$, $j \in J$. Также ясно, что это неравенство не исключает ни одного допустимого решения ЦЛ-задачи, так как оно было выведено в предположении, что переменные неотрицательные и целочисленные.

Запишем это неравенство в виде равенства

$$s = -f_{i,0} - \sum_{j\in J}(-f_{i,j})x_j,\ s \geq 0,\ s \equiv 0.$$

Целочисленность переменной s следует из $-s = x_{B_i} - \lfloor y_{i,0}\rfloor + \sum_{j\in J}\lfloor y_{i,j}\rfloor x_j$. Таким образом, это равенство можно присоединить к базисному представлению двойственно допустимого решения и решить новую задачу двойственным симплекс-алгоритмом. Если так полученное решение является целочисленным, прямо и двойственно допустимым, то оно является оптимальным для ЦЛ-задачи. Укажем основные шаги алгоритма.

Шаг 1 (начало). Решаем Л-задачу $\max\{cx \mid Ax = b,\ x \geq 0\}$, соответствующую ЦЛ-задаче. Переходим к шагу 2.

Шаг 2 (проверка оптимальности). Если решение является целочисленным, то оно является и оптимальным для ЦЛ-задачи. Если нет, то переходим к шагу 3.

Шаг 3 (построение сечения и реоптимизация). Выбираем производящую строку r с $f_{r,0} > 0$ и присоединяем строку, соответствующую равенству $s = -f_{i,0} - \sum_{j\in J}(-f_{r,j})x_j$ к нижней части симплекс-таблицы. Выполняем реоптимизацию, используя двойственный симплекс-алгоритм. Переходим к шагу 2.

Конечность этого алгоритма может быть показана, если сделать некоторые конкретизации, а именно:

а) на шаге 3 используется лексикографический двойственный симплекс-алгоритм, так что столбцы остаются лексикографически положительными;

б) если слабая переменная, соответствующая сечению, возвращается в базис, то вычёркивается соответствующая ей строка;

в) выбирается r из условия $r = \min(i \mid f_{i,0} \geq 0)$.

К сожалению, мало известно о верхней границе числа итераций этого алгоритма. При машинной реализации алгоритм оказался чувствительным к ошибкам округления, поскольку в нем необходимо правильно распознавать целое число. Итак, мотивируется поиск сечений, с помощью которых строятся целочисленные симплекс-таблицы.

Двойственный полностью целочисленный алгоритм. Пусть $0 < h < 1$ и $i = r$. Тогда сечение (6.1) есть

$$s = \lfloor hy_{r,0}\rfloor - \sum_{j\in J}\lfloor hy_{r,j}\rfloor x_j,\ s \geq 0,\ s \equiv 0.$$

Можно показать, что для некоторого h элемент $\lfloor hy_{r,k}\rfloor$ в этом сечении равен -1 и удовлетворяет тем требованиям, которые предъявляются в двойственном симплекс-алгоритме к ведущему элементу. Таким образом, это сечение может быть добавлено, а симплекс-итерация, порождаемая им, сохраняет двойственно допустимое целочисленное решение. Конечно, если обычный ведущий элемент есть -1, то не требуется ни какого сечения. Однако чтобы начать алгоритм, необходимо построить начальное двойственно допустимое целочисленное решение. При этом теряется преимущество поиска в окрестности оптимального решения Л-задачи.

Для доказательства конечности двойственного полностью целочисленного алгоритма требуется сделать конкретизации, которые аналогичны конкретизация указанным для алгоритма целых форм. Для этого алгоритма также мало известно о верхней границе числа итераций. Преждевременное окончание даёт недопустимое решение ЦЛ-задачи. Таким образом, было бы желательно производить последовательность допустимый решений ЦЛ-задачи.

Прямой полностью целочисленный алгоритм. Рассмотрим прямо допустимую целочисленную симплекс-таблицу. Если ведущий элемент $y_{r,k}$ обычного прямого симплекс-алгоритма равен 1, то симплекс-итерация сохраняет целочисленность симплекс-таблицы. Если это не так, то присоединяем сечение

$$s = \lfloor y_{r,0}/y_{r,k}\rfloor - \sum_{j\in J}\lfloor y_{r,j}/y_{r,k}\rfloor x_j,\ s \geq 0,\ s \equiv 0,$$

которое получается из неравенства (6.1) при $h = 1/y_{r,k} < 1$, $i = r$. Так как $\lfloor y_{r,0}/y_{r,k} \rfloor \le y_{r,0}/y_{r,k}$, то можно выполнить итерацию прямого симплекс-алгоритма для переменных s и x_k, при этом ведущий элемент равен 1.

Поэтому, начиная с прямо допустимой целочисленной симплекс-таблицы, можно сохранить прямо допустимое целочисленное решение.

Добавляемое сечение не исключает текущего целочисленного решения(оно может быть оптимальным), однако для нецелого $y_{r,0}/y_{r,k}$ это сечение исключает решение, которое может быть получено введением переменной x_k в базис на место переменной x_{B_r}. Доказательство конечности этого алгоритма также требует соответствующей конкретизации. Следует отметить, что добавляемое сечение часто вырождается ($\lfloor y_{r,0}/y_{r,k} \rfloor = 0$) и можно получить длинную, но конечную при определённых условиях последовательность итераций, не изменяющих решения.

6.4 АЛГОРИТМ НАХОЖДЕНИЯ ВСЕХ КРАЙНИХ ЛУЧЕЙ ЗАОСТРЕННОГО КОНУСА

Рассмотрим заострённый конус $C = \{x \mid x \ge 0,\ Ax \ge 0\}$. Здесь элементами $a_{i,j} m \times n$-матрицы A могут быть рациональные числа ($i = \overline{l,m}; j = \overline{l,n}$). Требуется найти управляющие векторы всех крайних лучей конуса C. Такую систему векторов будем называть системой всех крайних направлений.

Итерация k алгоритма состоит в том, что из системы всех крайних направлений $x_1, x_2, \ldots, x_r$ конуса, заданного $n+k-1$ неравенствами, получается система всех крайних направлений нового конуса, задаваемого теми же неравенствами и еще одним добавленным неравенством ($k = \overline{1,m}$). Каждому крайнему направлению x_i, записанному в виде строки, соответствует бинарная строка $y_i = (y_{i,1}, y_{i,2}, \ldots, y_{i,h})$, где

$$y_{i,t} = \begin{cases} 0,\ a_i x_i^T = 0, \\ 1,\ a_i x_i^T > 0, \end{cases} \quad i = \overline{1,r};\ t = \overline{1,h};\ h \le k$$

Здесь a_t — t-я строка матрицы A, h — число добавленных неравенств, менявших систему всех крайних направлений.

Работа алгоритма начинается с конуса, заданного условием неотрицательности $x \ge 0$.

Шаг 1. $r := n, h := 0, k := 1, x_i := e_i, i := \overline{1,r}$($e_i$–$i$-й единичный вектор).

Шаг 2. Если $k > m$, то переходим к шагу 7. Выбираем строку a_k матрицы A.

Шаг 3. Вычисляем скалярные произведения $b_i = a_k x_i^T, i = \overline{1,r}$. Если $b_i \ge 0$ для всех $i = \overline{1,r}$, то $k := k + 1$, и переходим к шагу 2. Если же $b_i < 0$ для всех $i = \overline{1,r}$, то $r := 0$, и переходим к шагу 7.

Шаг 4. Для каждой пары (k,l) такой, что $b_k < 0$и $b_l > 0$, подсчитываем число позиций, в которых стоят нули в обеих строках,

$$(x_{k,1}, x_{k,2}, \ldots, x_{k,n}, y_{k,1}, y_{k,2}, \ldots, y_{k,h}),$$

$$(x_{l,1}, x_{l,2}, \ldots, x_{l,n}, y_{l,1}, y_{l,2}, \ldots, y_{l,h}).$$

Если это число больше или равно $n-2$, то проверяем, существует ли среди строк $(x_{i,1}, x_{i,2}, \ldots, x_{i,n}$, $y_{i,1}, y_{i,2}, \ldots, y_{i,h})$, $i = \overline{1,r} \wedge i \ne k \wedge i \ne l$, строка, имеющая нули во всех тех же позициях. Если такой строки нет, то заносим в новую систему всех крайних направлений вектор $x' = b_l x_k + |b_k| x_l$. Соответствующая вектору x' бинарная строка $y' = (y'_1, y'_2, \ldots, y'_h)$ образуется по правилу

$$y'_t = \begin{cases} 0,\ y_{k,t} = 0 \wedge y_{l,t} = 0, \\ 1,\ y_{k,t} = 1 \vee y_{l,t} = 1, \end{cases} \quad T = \overline{1,h}.$$

Шаг 5. Присоединяем к новой системе всех крайних направлений те из старых, для которых $b_i \ge 0$. Получим систему всех крайних направлений $x'_1, x'_2, \ldots, x'_{r'}$ конуса, заданного $n + h - 1$ неравенствами.

Шаг 6. Добавляем в каждую бинарную строку y'_i новой системы элемент $y'_{i,h+1}$. Для вновь образованных векторов $y'_{i,h+1} = 0$, а для старых

$$y'_{i,h+1} = \begin{cases} 0,\ b_i = 0, \\ 1,\ b_i > 0. \end{cases}$$

Заменяем h на $h+1, k := k+1$, опускаем штрихи при x'_i, y'_i, r' и переходим к шагу 2.

Шаг 7. Работа алгоритма закончена. Система всех крайних направлений $x_1, x_2, \ldots, x_r$ конуса C найдена. Если $r = 0$, то существует только нулевое решение рассматриваемой системы неравенств.

Замечание 1. Если требуется найти направляющие векторы единичной длины (под длиной вектора $a = (a_1.a_2, \ldots, a_n)$ понимаем величину $|a| = \sqrt{\sum_{k=1}^{n} a_j^2}$), то в выражении $x' = \alpha x_k + \beta x_l$ коэффициенты α, β вычисляются по формулам

$$\alpha = \sqrt{(b_l^2)/(b_l^2 - 2b_l b_k x_l x_k^T + b_k^2)},\ \beta = (-\alpha b_k)/b_l$$

(это легко проверить). При этом на шаге 2 следует выполнить нормировку строки a_k матрицы A ($a_k := a_k/|a_k|$).

Замечание 2. Если на итерации k для некоторого фиксированного $t, 1 < t < h$, среди элементов $y_{i,t}$, $i = \overline{1, r}$ находятся $0, 1$ или r нулей, то столбец с номером t далее сохраняет такое же строение. Поэтому столбцы этих трёх типов можно исключить из дальнейшего рассмотрения.

Замечание 3. Нетрудно геометрически истолковать все шаги алгоритма. Распределение памяти, организация счёта и другие вычислительные детали, весьма важные для машинной реализации приведены в [56] (см. также § 8.6). Алгебраическое обоснование алгоритма даётся в § 6.7.

Замечание 4, Не трудно выделить те части алгоритма, при выполнении которых можно осуществить распараллеливание вычислений (например, векторизацию).

Пример 6.1. Найти систему всех крайних направлений конуса C, задаваемого системой неравенств $x \geq 0$, $Ax \geq 0$, где

$$A = \begin{pmatrix} 1 & -1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & -1 & 1 & 0 \\ 0 & 1 & 0 & -1 & 0 & 1 \\ 0 & 0 & 1 & 0 & -1 & 1 \end{pmatrix}.$$

Приведем результаты вычислений.

Результат итерации 1

	x_1	x_2	x_2	x_2	x_5	x_6
1	1	0	0	0	0	0
2	0	0	1	0	0	0
3	0	0	0	1	0	0
4	0	0	0	0	1	0
5	0	0	0	0	0	1
6	1	1	0	0	0	0
7	0	1	1	0	0	0

Результат итерации 2

	x_1	x_2	x_2	x_2	x_5	x_6
1	1	0	0	0	0	0
2	0	0	1	0	0	0
3	0	0	0	0	1	0
4	0	0	0	0	0	1
5	1	1	0	0	0	0
6	0	1	1	0	0	0
7	1	0	0	1	0	0
8	0	0	0	1	1	0
9	1	1	0	1	0	0

Результат итерации 3

	x_1	x_2	x_2	x_2	x_5	x_6
1	1	0	0	0	0	0
2	0	0	1	0	0	0
3	0	0	0	0	1	0
4	0	0	0	0	0	1
5	1	1	0	0	0	0
6	0	1	1	0	0	0
7	1	1	0	1	0	0
8	1	0	0	1	0	1
9	0	0	0	1	1	1
10	0	1	1	1	1	0

Результат итерации 4

	x_1	x_2	x_2	x_2	x_5	x_6
1	1	0	0	0	0	0
2	0	0	1	0	0	0
3	0	0	0	0	0	1
4	1	1	0	0	0	0
5	0	1	1	0	0	0
6	1	1	0	1	0	0
7	1	0	0	1	0	1
8	0	0	0	1	1	1
9	0	1	1	1	1	0
10	0	0	1	1	0	1
11	0	0	0	0	1	1
12	0	1	1	0	1	0

6.5 АЛГОРИТМ НАХОЖДЕНИЯ ВСЕХ ВЕРШИН МНОГОГРАННИКА

Рассмотрим Л-задачу $\min\{cx \mid Ax = b,\ x \geq 0\}$, где элементы $m \times n$-матрицы A, m-столбца b и n-строки c — заданные действительные числа. Требуется найти все вершины многогранника $P = \{x, Ax = b,\ x \geq 0\}$ и упорядочить их по возрастанию значений целевой функции cx.

Каждая вершина $x_i = (x_{1,i}, x_{2,i}, \ldots, x_{n,i})$ характеризуется записью, состоящей из значения целевой функции $c_i = cx_i$ и из бинарного столбца $l_i = (l_{0,i}, l_{1,i}, \ldots, l_{n,i})$. Здесь

$$l_{j,i} = \begin{pmatrix} 0, x_{j,i} = 0, \\ 1, x_{j,i} > 0. \end{pmatrix} \quad j = \overline{1,n}$$

Сначала полагаем $l_{0,i} = 0$. Если все вершины, смежные с вершиной x_i, уже получены, а их записи занесены в список, то $l_{0,i} = 1$.

Шаг 1. Двухфазным прямым симплекс-алгоритмом решаем Л-задачу $\min\{cx \mid Ax = b, x \geq 0\}$. После первой фазы из условий задачи исключаем линейно зависимые уравнения. В список под номером 1 заносим запись вершины, являющейся уже найденным оптимальным базисным решением Л-задачи. Объявляем вершину x_1 текущей, обозначаем её через x' и переходим к шагу 4.

Шаг 2. Среди записей списка, удовлетворяющих условию $c_i \geq c' \wedge l_{0,i} = 0$, выбираем запись i_0 с наименьшем значением c_{i0}. Если таких записей нет, то переходим к шагу 6.

Шаг 3. Исходя из текущей вершины x', для которой известна обратная базисная матрица, при помощи конечного числа преобразований с ведущим элементом получаем обратную базисную матрицу для вершины x_{i_0}. Объявляем вершину x_{i_0} текущей и обозначаем её через x'.

Шаг 4. Если текущая вершина x' не является вырожденной, то, последовательно вводя в базис все небазисные столбцы, находим все смежные с x' вершины x со значениями $cx \geq cx'$, заносим их записи под соответствующими номерами в список (если их там ещё не было), полагаем $l'_0 = 0$ и переходим к шагу 2.

Шаг 5. Для вырожденной текущей вершины x' сначала строим все крайние направления многогранника P, выходящие из x', затем при помощи этих направлений находим все смежные с x' вершины x со значениями $cx \geq cx'$, заносим их записи под соответствующей номерами в список (если их там еще не было). Полагаем $l'_0 = 1$ и переходим к шагу 2.

Шаг 6. Все вершины многогранника P получены. Алгоритм заканчивает свою работу.

Рассмотрим более детально отдельные шаги алгоритма. На шаге 3 переход от базиса B' текущей вершины x' к базису B вершины x осуществляется таким образом. Выделяем столбцы базиса B, которые соответствуют положительным базисным переменным и находятся вне базиса B'. Последовательно вводим эти столбцы в базис, выводя из него столбцы, соответствующие нулевым компонентам вершины x. Возможность такого перехода опирается на линейную независимость базисных векторов. Заметим, что матрица коэффициентов разложения всех векторов из B по базису B' является невырожденной.

Теперь рассмотрим шаги 4 и 5. Пусть заданы базис B, вершина x многогранника $P = \{x \mid Ax = b,\ x \geq 0\}$. Покажем, как построить все крайние направления многогранника P, выходящие из вершины x.

Разобьем матрицу A и вектор x на две части $A = (BN)$, $x = (x_B, x_N)$. Рассмотрим многогранник $P' = \{x_N \mid B^{-1}b - B^{-1}Nx_N \geq 0,\ x_N \geq 0\}$. Вершине $x = (B^{-1}b, 0)$ многогранника P соответствует вершина $x_N = 0$ многогранника P', и наоборот. Поэтому достаточно найти все крайние направления, выходящие из вершины $x_N = 0$ многогранника P'.

Если вершина $x = (B^{-1}b, 0)$ не является вырожденной, т. е. $B^{-1}b > 0$, то все крайние направления многогранника P' будут лежать на единичных векторах, соответствующих небазисным переменным x_N.

Теперь рассмотрим случай, когда вершина $x = (B^{-1}b, 0)$ является вырожденной, т. е. $I = \{i \mid (B^{-1}b)_i = 0\} \neq \infty$. В этом случае крайние направления, выходящие из вершины $x_N = 0$ многогранника P', будут лежать на крайних лучах конуса $C = \{x_N \mid x_N \geq 0, (-B^{-1}Nx_N)_i \geq 0, i \in I\}$. Для отыскания этих крайних направлений применяется описанный выше алгоритм нахождения всех крайних лучей заострённого конуса (см. § 6.4).

Описанный алгоритм отличается от других алгоритмов для нахождения всех вершин многогранника тем, Что перебор осуществляется по вершинам, а не по их базисам. Для вырожденной вершины может существовать огромное количество представляющих её базисов. Приведем пример, иллюстрирующий такую ситуацию.

Пример 6.2. Рассмотрим Л-задачу

$$\min\Big\{\sum_{j=1}^{n} c_j x_j \mid x_{j_1} + x_{j_2} \leq 1,\ 1 \leq j_1 < j_2 \leq n,\ x_j \geq 0,\ j = \overline{1,n}\Big\}.$$

Эта задача содержит $n(n-1)/2$ основных ограничения. Для нечётного целого числа n вершина $x = (1/2, 1/2, \ldots, 1/2)$ многогранника условий задачи имеет не менее чем $(n-1)!/2$ представляющих её базисов.

Действительно, определитель матрицы коэффициентов системы неравенству $x_{j_r} + x_{j_{r+1}} \leq 1, r = \overline{1,n}$ ($j_{n+1} = j_1$), для нечётного n равен 2. Здесь $(j_1, j_2, \ldots, j_n)$ — перестановка чисел $1, 2, \ldots, n$. Заметим, что $2n$ перестановок

$$(j_1, j_2, \ldots, j_n), (j_2, j_2, \ldots, j_n, j_1), \ldots, (j_n, j_1, j_2, \ldots, j_{n-1}),$$

$$(j_n, j_{n-1}, \ldots, j_1), (j_{n-1}, j_{n-2}, \ldots, j_1, j_n), \ldots, (j_1, j_n, j_{n-1}, \ldots, j_2),$$

задают одну и ту же систему неравенств. Таким образом, число представлений рассматриваемого вида для: вершины $x = (1/2, 1/2, \ldots, 1/2)$ будет равно $n!/(2n) = (n-1)!/2$.

Описанный здесь алгоритм для нахождения всех вершин многогранника был реализован на языке Алгол-60. Практически находились все вершины из некоторой окрестности оптимального базисного решения линейной задачи. Этот подход позволил решить ряд конкретных задач, которые не поддавались решению другим способом (см., например, [52]).

6.6 ПРЯМОЙ АЛГОРИТМ РЕШЕНИЯ ЦЕЛОЧИСЛЕННОЙ ЛИНЕЙНОЙ ЗАДАЧИ ОПТИМИЗАЦИИ

Этот алгоритм будет изложен на геометрическом языке. Некоторые детали останутся вне рассмотрений. Возможность распараллеливания алгоритма обусловлена распараллеливаемостью его составных частей.

Рассмотрим ЦЛ-задачу $\min\{cx \mid A'x \geq b',\ x \geq 0,\ x \equiv 0\}$. Здесь $m \times n$-матрица A', m-столбец b' и n-строка c состоят из заданных целых чисел. Считаем, что $b' \leq 0$.

Через P обозначим выпуклую оболочку всех допустимых решений этой ЦЛ-задачи, т. е. выпуклую оболочку множества точек $\{x \mid Ax \geq b,\ x \equiv 0\}$, где $A = \begin{pmatrix} I_n \\ A' \end{pmatrix}$, $b = \begin{pmatrix} 0 \\ b' \end{pmatrix}$. Прямой алгоритм состоит в построении конечной последовательности целочисленных вершин многогранника P, которая приводит к минимуму.

Предположим, что на некотором шаге алгоритма известны целочисленная вершина x' многогранника P и конус $C' = \{x \mid D'x \geq f',\ |D'| \neq 0\}$ с вершиной в точке x'. При этом $P \subseteq C'$. В систему $D'x \geq f'$ входят некоторые неравенства из условий задачи $Ax \geq b$, а также, быть может, неравенства, построенные на предыдущих шагах алгоритма. Матрица D' и вектор f' целочисленные.

Пусть система целочисленных векторов $u_1, u_2, \ldots, u_n$ задаёт все крайние направления конуса C'.

Достаточное условие оптимальности. Если $cu_k \geq 0$ для всех $k = \overline{l,n}$, то целочисленная точка x' является оптимальным решением ЦЛ-задачи.

Необходимое и достаточное условие оптимальности. Для каждого крайнего направления u'_k, выходящего из вершины x' многогранника P, выполняется неравенство $cu'_k \geq 0 (k = \overline{1,n'})$.

Таким образом, поиск направления, сдвиг вдоль которого приводит к лучшему решению (такое направление будем называть улучшающим), можно организовать, например, таким образом.

Среди векторов $u_1, u_2, \ldots, u_n$ ищем направление u_k, удовлетворяющее условиям: а) $cu_k < 0$; б) $a_i u_k \geq 0$, $i \in I$, где $I = \{i \mid a_i x' = b_i\}$ (a_i — i-я строка матрицы A); в) на луче $x(\theta) = x' + \theta u_k, \theta > 0$, существует по крайней мере одно допустимое решение ЦЛ-задачи.

Если улучшающее направление не найдено, то нужно перейти к конусу $\widetilde{C} = \{x \mid D'x \geq f',\ a_i x \geq b_i,\ i \in I\}$ и для всех его крайних направлений последовательно проверить условия а), в).

Если же и в этом случае улучшающее направление не найдено, то рассмотрим конус $\widetilde{\widetilde{C}} = \{x \mid D'x \geq f',\ a_i x \geq b_i,\ i \in I,\ \pi_i x \geq \pi_{i,0},\ i \in I'\}$ с вершиной в точке x'. Неравенства $\pi_i x \geq \pi_{i,0}$, $i \in I'$ задают грани соответствующего углового многогранника, которые проходят через точку x'. Напомним, что под угловым многогранником конуса C понимаем выпуклую оболочку всех целочисленных точек этого конуса. В примере 6.4 показано, как можно строить нужную грань (полное алгебраическое обоснование рассматриваемого алгоритма выходит за рамки данного изложения).

При переходе от C' к $\widetilde{C}$ и от $\widetilde{C}$ к $\widetilde{\widetilde{C}}$ изменяется система всех крайних направлений соответствующего конуса. Если для всех крайних направлений некоторого конуса $\widetilde{\widetilde{C}}$ не выполняется условие а), то точка x' является оптимальным решением. Неравенство $cu_k < 0$ приводит либо к новой целочисленной точке, либо к исключению направления u_k. Этот процесс конечен.

Остановимся более детально на условии в). Пусть для некоторого целочисленного вектора u_k из системы $u_1, u_2, \ldots, u_n$ выполняются условия а) и б). Рассмотрим точки $x(\theta) = x' + \theta u_k, \theta \geq 0$. Если точки $x(\theta)$ для любого $\theta \geq 0$ удовлетворяют системе $Ax \geq b$, то значения целевой функции не ограничены снизу на множестве целочисленных точек $\{x(k),\ k = \overline{1,+\infty}\}$. Определим величину θ_0 из соотношения

$$\theta_0 = \min_{a_i u_k < 0} \{(a_i x' - b_i)/(-a_i u_k)\}.$$

Через l обозначим номер неравенства, где достигается минимум ($1 \leq l \leq n + m$). Таких l может быть несколько.

Пусть θ' — наибольшее значение θ из интервала $[0, \theta_0]$, для которого точка $x(\theta')$ целочисленная. Если $\theta' = 0$, то такое направление не является крайним направлением многогранника P и не может быть улучшающим направлением.

Пусть $x(\theta') = x''$ — новая вершина многогранника P с меньшим значением целевой функции. Рассмотрим конус $C'' = \{x \mid D''x \geq f'',\ |D''| \neq 0\}$ с вершиной в точке x''. Этот конус задаётся $n - 1$

неравенствами, левые части которых линейно независимы и обращаются вектором u_k в нуль, а также гранью углового многогранника соответствующего конуса с вершиной в точке $x(\theta_0)$.

Пример 6.3. Минимизировать $-x_1-2x_2$ при условиях $x_1+x_2 \le 7, 2x_1 \le 11, 2x_2 \le 7, x_j \ge 0 \wedge x_j \in Z$, $j = \overline{1,2}$. На рис. 6.4 область, обведенная волнистой линией, изображает выпуклую оболочку P всех допустимых решений этой конкретной ЦЛ-задачи.

Начинаем с вершины $M_1(0,0)$ многогранника P и конуса $x_1 \ge 0, x_2 \ge 0$. Среди двух улучшающих направлений $(1,0)$ и $(0,1)$, выходящих из точки M_1, выбираем направление $(1,0)$. Угловым многогранником для конуса $2x_1 \le 11, 2x_2 \ge 0$ служит многогранник $x_1 \le 5, x_2 ge 0$. Таким образом, сдвиг вдоль направления $(1,0)$ приводит в точку $M_2(5,0)$ и к конусу $x_1 \le 5, x_2 \ge 0$.

Среди направлений $(-1,0)$ и $(0,1)$, выходящих из точки M_3 улучшающим является направление $(0,1)$. Сдвиг вдоль этого направления приводит в точку $M_3(5,2)$ и к конусу $x_1 \le 5$, $x_1 + x_2 \le 7$.

Среди направлений $(0,-1)$ и $(-1,1)$, выходящих из точки M_3, улучшающим является направление $(-1,1)$. Угловым многогранником для конуса $x_1 + x_2 \le 7$, $2x_2 \le 7$ служит многогранник $x_1 + x_2 \le 7$, $x_2 \le 3$. Сдвиг вдоль направления $(-1,1)$ приводит в точку $M_4(4,3)$ и к конусу $x_1 + x_2 \le 7$, $x_2 \le 3$.

Направления $(1,-1)$ и $(-1,0)$, выходящие из точки M_4, не являются улучшающими, поскольку целевая функция вдоль этих направлений возрастает. Следовательно, точка $M_4(4,3)$ — оптимальное решение с оптимальным значением -10.

Все рассуждения и построения опираются на рис. 6.4.

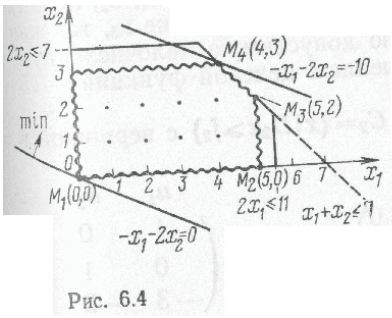


Рис. 6.4

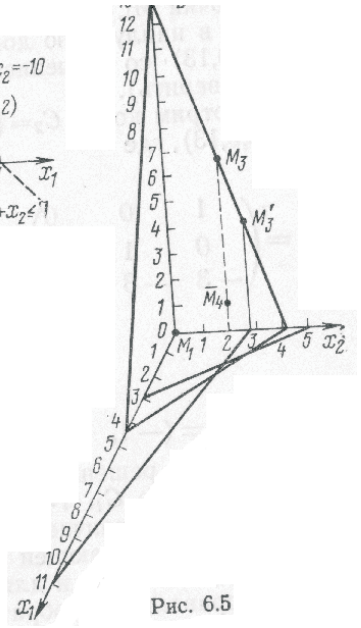


Рис. 6.5

Пример 6.4. Минимизировать

$$-4x_1 - 5x_2 - 1x_3 \text{ при условиях,}$$
$$-3x_1 - 2x_2 - 0x_3 \ge -10,$$
$$-1x_1 - 4x_2 - 0x_3 \ge -11,$$
$$-3x_1 - 3x_2 - 1x_3 \ge -13,$$
$$x_j \ge 0 \wedge x_j \in Z,\ j = \overline{1,3}.$$

На рис. 6.5 изображен процесс решения этой конкретной ЦЛ-задачи описанным выше прямым алгоритмом.

Начинаем с конуса $C_1 = \{x \mid D_1 x \ge f_1\}$, имеющего вершину в точке $M_1(0,0,0)$, где

$$D_1 = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}, \quad D_1^{-1} = \begin{matrix} u_1 & u_2 & u_2 \\ \end{matrix} \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}, \quad f_1 = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix},$$

$$cD_1^{-1} = (-4, -5, -1), \quad cD_1^{-1} f_1 = 0.$$

Среди трёх улучшающих направлений u_1, u_2, u_3, выходящих из точки M_1, выбираем направление u_3, так как оно приводит в наилучшую допустимую целочисленную точку $M_2(0, 0, 13)$ со значением целевой функции -13 (это легко проверить).

Рассмотрим конус $C_2 = \{x \mid D_2 x \geq f_2\}$ с вершиной в точке $M_2(0, 0, 13)$, где

$$D_2 = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ -3 & -3 & -1 \end{pmatrix}, \quad D_2^{-1} = \begin{array}{c} \begin{matrix} u_1 & u_2 & u_2 \end{matrix} \\ \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ -3 & -3 & -1 \end{pmatrix} \end{array}, \quad f_2 = \begin{pmatrix} 0 \\ 0 \\ -13 \end{pmatrix},$$

$$cD_2^{-1} = (-1, -2, 1), \quad cD_2^{-1} f_2 = -13.$$

В процессе решения строится конечная последовательность конусов C_i, $i = l, 2, 3, \ldots$. Матрица D_i задающая конус C_i отличается только одной строкой от матрицы D_{i-1}, задающей конус C_{i-1} ($i = 2, 3, \ldots$). В частности, это верно и для $i = 2$. Поэтому нужны формулы, подправляющие обратную матрицу. Укажем их.

Пусть невырожденные матрицы $D = (d_{i,j})$ и $D' = (d'_{i,j})$ отличаются строкой с номером k ($d'_i = d_i$, $i \neq k$). Тогда столбцы обратной матрицы $(D')^{-l} = (q'_{i,j})$ выражаются через столбцы обратной матрицы $D^{-1} = (q'_{i,j})$ по формулам:

$$q'_k = (1/z_k) q_k, \; q'_j = q_j - z_j q'_k, \; j \neq k$$

Здесь $z = (z_1, z_2, \ldots, z_n) = d'_k D^{-1}$, $z_k \neq 0$.

Среди двух улучшающих направлений u_1 и u_2, выходящих из точки M_2, выбираем направление u_2, так как оно приводит в лучшую допустимую целочисленную точку $M_3(0, 2, 7)$ со значением целевой функции -17 (это легко проверить).

Рассмотрим конус $C'_3 = \{x \mid D'_3 x \geq f'_3\}$ с вершиной в нецелочисленной точке $M_3(0, 11/4, 19/4)$, где

$$D'_3 = \begin{pmatrix} 1 & 0 & 0 \\ -1 & -4 & 0 \\ -3 & -3 & -1 \end{pmatrix}, \quad (D'_3)^{-1} = \begin{pmatrix} 1 & 0 & 0 \\ -1/4 & -1/4 & 0 \\ -9/4 & 3/4 & -1 \end{pmatrix}, \quad f'_3 = \begin{pmatrix} 0 \\ -11 \\ -13 \end{pmatrix},$$

$$c(D'_3)^{-1} = (-2/4, -2/4, 1), \quad c(D'_3)^{-1} f'_3 = -74/4.$$

Теперь нужно построить грань углового многогранника конуса C'_3, проходящую через целочисленную точку M_3. Покажем, как это можно сделать.

Целочисленные точки конуса $C = \{x \mid Dx \geq f, \; |D| \neq 0\}$ однозначно определяются неотрицательными целочисленными решениями $t = (t_1, t_2, \ldots, t_n)$ группового уравнения $\sum_{j=1}^{n} g_j t_j = g_0$, где $g_j, j = \overline{0, n}$, — элементы фактор-группы $G = Z_n / \{D\}(Dx - It = f)$. Чтобы представить фактор-группу G в виде прямой суммы циклических групп и найти её элементы g_i следует привести матрицу D к нормальному виду. Через Q обозначим выпуклую оболочку всех неотрицательных целочисленных решений t группового уравнения. Предположим, что нужно найти грань многогранника Q, проходящую через его вершину $(t'_1, 0, \ldots, 0)$, $t'_i \neq 0$. Будем искать неравенство задающее эту грань, в виде

$$\sum_{j=1}^{n} \pi_j t_j \geq 1, \pi_j \geq 9, \; j = \overline{1, n}.$$

Для всех $g \in G$ определяем функции

$$\psi_s(g) = \min\Big\{\sum_{j=1}^{s} \pi_j t_j \mid \sum_{j=1}^{s} g_j t_j = g, \; t_j \geq 0 \wedge t_j \in Z, \; j = \overline{1, s}\Big\},$$

$$s = \overline{1, n}.$$

При этом $\psi_0(g) = +\infty$, $g \in (G/0)$, а $\psi_s(0) = 0$, $s = \overline{0,n}$. Функции $\psi_s(g)$ удовлетворяют рекуррентным соотношениям вида $\psi_s(g) = \min\{\psi_{s-1}(g),\ \psi_s(g - g_s) + \pi_s\}$, $s = \overline{1,n}$. Сначала неизвестны все коэффициенты π_j, $j = \overline{1,n}$. Предположим, что уже найдены первые s коэффициентов. Тогда

$$\pi_{s+1} = \max_i\{(1 - \psi_s(g_0 - m_i g_{s+1}))/m_i \mid \psi_s(g_0 - m_i g_{s+1}) < 1,\ i = 1, 2, 3, \dots\}.$$

Если не существует значений m_i, удовлетворяющих указанному условию, то $\pi_{s+1} = 0$. Теперь уже можно вычислить и функцию $\psi_{s+1}(s)$.

Элементарные преобразования $V_{3,1}(-3), V_{3,2}(-3), U_{2,1}(1), V_2(-1), V_3(-1), V_{2,3}, U_{2,3}$ приводят матрицу D_3 к нормальному виду

$$\begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 4 \end{pmatrix},$$

а уравнение $D_3'x - It = f_3'$ даёт групповое уравнение $g_3t_1 + g_3t_2 = g_1$(см. § 3.7, 7.3, 7.5). Для точки $M_3(0, 2, 7)$ имеем $t_2 = 3$. Выполняя описанные выше вычисления, получаем неравенство $1/3t_1 + 1/3t_2 \geq 1$. Это неравенство в исходных переменных x_j принимает вид $1/3x_1 + 1/3(-x_1 - 4x_2 + 11) \geq 1$, т. е. $-x_2 \geq -2$.

Рассмотрим конус $C_3 = \{x \mid D_3x \geq f_3\}$ с вершиной в точке $M_3(0, 2, 7)$, где

$$D_3 = \begin{pmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ -3 & -3 & -1 \end{pmatrix}, \quad D_3^{-1} = \begin{matrix} u_1 & u_2 & u_2 \\ \end{matrix}\begin{pmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ -3 & -3 & -1 \end{pmatrix}, \quad f_3 = \begin{pmatrix} 0 \\ -2 \\ -13 \end{pmatrix},$$

$$cD_3^{-1} = (-1, 2, 1), \quad cD_3^{-1}f_3 = -17.$$

Существует только одно улучшающее направление u_1, выходящее из точки M_3. Оно приводит в допустимую целочисленную точку $M_4(2, 2, 1)$ со значением целевой функции -19 (на рис. 6.5 показана её проекция — точка $\overline{M}_4$).

Наконец, рассмотрим конус $C_4 = \{x \mid D_2x \geq f_4\}$ с вершиной в точке $M_4(2, 2, 1)$, где

$$D_4 = \begin{pmatrix} -3 & -2 & 0 \\ 0 & -1 & 0 \\ -3 & -3 & -1 \end{pmatrix}, \quad D_4^{-1} = \begin{matrix} u_1 & u_2 & u_2 \\ \end{matrix}\begin{pmatrix} -1/3 & 2/3 & 0 \\ 0 & -1 & 0 \\ 1 & 1 & -1 \end{pmatrix}, \quad f_4 = \begin{pmatrix} -10 \\ -2 \\ -13 \end{pmatrix},$$

$$cD_4^{-1} = (1/3, 4/3, 1), \quad cD_4^{-1}f_4 = -19.$$

Направления u_1, u_2, u_3, выходящие из точки M_4, не являются улучшающими. Выполняется критерий оптимальности. Следовательно, целочисленная точка $M_4(2, 2, 1)$ является оптимальным решением рассматриваемой конкретной ЦЛ-задачи.

Замечание 5. Изложенная теория и конкретные примеры показывают, что легко построить все грани выпуклой оболочки допустимого множества, скажем, ЦЛ-задачи.

6.7 ОБОСНОВАНИЕ АЛГОРИТМОВ

Сформулируем и докажем две теоремы, на которые опирается обоснование алгоритма нахождения всех крайних лучей заострённого конуса. Тем самым алгоритмы из § 6.4 и 6.5 будут обоснованы полностью, а алгоритм из § 6.6 — частично.

В этом параграфе будем рассматривать заострённый конус C, задаваемый конечной системой линейных неравенств $C = \{x \mid Ax \geq 0,\ r(A) = n\}$ или системой всех его крайних направлений

$$C = \Big\{x \mid x = \sum_{j=1}^{r} \alpha_j x_j,\ \alpha_j \geq 0,\ j = \overline{1,r}\Big\}.$$

Линейное подпространство максимальной размерности, содержащееся в заострённом конусе C, состоит только из нулевого вектора. Это свойство, определяющее понятие заострённого конуса, часто используется в приводимых ниже рассуждениях.

Множество векторов $\{x\} = \{y \mid y = \alpha x,\ \alpha \geq 0,\ x \neq 0\}$ называется лучом, а сам вектор x — направляющим вектором этого луча. Также говорят, что вектор x порождает луч $\{x\}$. Под крайним направлением конуса C понимают такой вектор x из $C, x \neq 0$, который непредставим в виде $x = x_1 + x_2$, где $x_1 \in C, x_2 \in C, \{x_1\} \neq \{x\}, \{x_2\} \neq \{x\}$. Если x — крайнее направление конуса C, то для любого положительного скаляра a вектор сея также является его крайним направлением.

Луч, порождаемый крайним направлением конуса C, называется крайним лучом этого конуса. Свяжем с каждым крайним лучом конуса C только один вектор, порождающий его. Полученную таким образом систему векторов будем называть системой всех крайних направлений конуса C. Если $x_1, x_2, \ldots, x_r$ — система всех крайних направлений конуса C, то система векторов $\alpha_1 x_1, \alpha_2 x_2, \ldots, \alpha_r x_r$, где $\alpha_j \geq 0$ для всех $j = \overline{1,r}$, будет такой же.

Итак, задача нахождения всех крайних лучей конуса состоит в отыскании какой-нибудь системы всех его крайних направлений.

Если скалярное произведение ax строки a и столбца x равно нулю, то говорят, что вектор x аннулирует или обращает в нуль строку a.

Теорема 6.1. Пусть $C = \{x \mid Ax \geq 0\}$ и $C' = \{x \mid Ax \geq 0, ax \geq 0\}$ — два заданных конуса, где A — n-столбцовая действительная матрица ранга n. Для конуса C известна система всех его крайних направлений $x_1, x_2, \ldots, x_r$.

Тогда система всех крайних направлений $x'_1, x'_2, \ldots, x'_{r'}$, конуса C получается из системы $x_1, x_2, \ldots, x_r$ следующим образом:

а) $x'_i = x_i$, если $ax_i \geq 0 (i = \overline{1,r})$;

б) $x'_{k,t} = |ax_l| x_k + |ax_k| x_l$, где x_k, x_l — пара векторов, для которых скалярные произведения ax_k, ax_l имеют противоположные знаки и для которых существует $n-2$ линейно независимых строк матрицы A, аннулируемых как x_k, так и x_l $(k = \overline{1,r},\ l = \overline{1,r},\ k \neq l)$.

Прежде чем перейти к доказательству теоремы 6.1, докажем две леммы.

Лемма 6.1. Вектор $x, x \neq 0$, из конуса $C = \{x \mid Ax \geq 0\}$ является крайним направлением этого конуса тогда и только тогда, когда он аннулирует $n-1$ линейно независимых строк матрицы A. Здесь A — n-столбцовая действительная матрица ранга n.

Достаточность. Пусть вектор $x, x \neq 0$, аннулирует $n-1$ линейно независимых строк матрицы A и не является крайним направлением конуса C, т. е. $x = x_1 + x_2, x1 \in C, x_2 \in C, \{x_1\} \neq \{x\}, \{x_2\} \neq \{x\}$. Тогда вектор x_1 аннулирует эти же строки. Следовательно, $x_1 = \alpha x$, а из $x_1 \in C, x \in C$ следует $\alpha > 0$. Таким образом, $\{x_1\} = \{x\}$, что приводит к противоречию.

Необходимость. Пусть x — крайнее направление конуса C, а аннулируемые им строки матрицы A имеют ранг меньший, чем $n-1$. Тогда существует вектор x', аннулирующий эти же строки и линейно независимый с x. Поскольку $x \neq 0$ и $r(A) = n$, то существует такое $\varepsilon > 0$, что $x_1 = (1/2x - \varepsilon x') \in C$, $x_2 = (1/2x + \varepsilon x') \in C$. Например,

$$\varepsilon < \frac{1}{2} \min_{a_i x > 0 \wedge |a_i x'| \neq 0} \frac{a_i x}{|a_i x'|}.$$

Тогда $x = x_1 + x_2$. Это противоречит тому, что x — крайнее направление конуса.

Лемма 6.2. Для системы всех крайних направлений $x_1, x_2, \ldots, x_r$ конуса C имеют место следующие утверждения:

а) любые два вектора x_k и x_l из системы линейно независимы $(k \neq l)$;

б) если вектор x_i обращает в нуль $n-2$ линейно независимых строк матрицы A, аннулируемых как x_k, так и x_l, то либо $x_i = x_k$, либо $x_i = x_l$ $(i = \overline{1,r},\ k = \overline{1,r},\ l = \overline{1,r},\ k \neq l)$.

Доказательство. Пусть $\alpha_1 x_k + \alpha_2 x_l = 0$. Если $\alpha_1 > 0, \alpha_2 < 0$, то $\alpha_1 x_1 = -\alpha_2 x_l$ и $\{x_k\} = \{x_l\}$ (это противоречит тому, что векторы x_k и x_l порождают различные крайние лучи). Значит, коэффициенты α_1, α_2 должны быть одинакового знака. Для определённости считаем, что они неотрицательные. Тогда $-\alpha_1 x_k = -\alpha_2 x_l \in C$. Но и $\alpha_1 x_k \in C$. Следовательно, $\alpha_1 x_k = 0, \alpha_1 = 0$, так как конус заострённый. Аналогично доказывается, $\alpha_2 = 0$.

Теперь докажем утверждение б). Рассмотрим подпространство, ортогональное к $n-2$ линейно независимым строкам матрицы A, которые аннулируются как x_k, так и x_l. Это подпространство имеет

размерность 2, содержит векторы x_k, x_l и натянуто на них, так как эти векторы линейно независимы (утверждение а) леммы 6.2). Следовательно, $x_i = \alpha_1 x_k + \alpha_2 x_l$, где α_1, α_2 — действительные числа.

Случай $\alpha_1 > 0, \alpha_2 > 0$ невозможен, так как каждый из векторов x_k, x_l обращает в нуль те строки матрицы A, которые аннулируются x_i. Среди этих строк существует $n-1$ линейно независимых. Подпространство, ортогональное к ним, имеет размерность 1 и натянуто, например, на вектор x_l. Тогда $x_k = \beta x_l$, где β — действительное число. Это противоречит линейной независимости векторов x_k, x_l, установленной в утверждении а) леммы 6.2.

Теперь считаем, что в равенстве $x_i = \alpha_1 x_k + \alpha_2 x_l$, например, $\alpha_1 < 0$. Тогда из $x_i - \alpha_1 x_k = \alpha_2 x_l$ следует, что вектор x_i обращает в нуль $n-1$ линейно независимых строк, аннулируемых x_l. Тогда $x_i = \gamma x_l$. Отсюда согласно утверждению а) леммы 6.2 имеем $x_i = x_l$.

Доказательство теоремы 6.1. Все векторы, перечисленные в формулировке теоремы 6.1, принадлежат конусу C' и не равны нулю. Каждый из них аннулирует $n-1$ линейно независимых строк матрицы $A' = \begin{pmatrix} A \\ a \end{pmatrix}$. Это очевидно для векторов x'_i, а для векторов $x'_{k,l}$ следует из того, что строка a линейно независима с $n-2$ строками матрицы A, которые аннулируются как x_k, так и x_l.

Покажем, что перечисленные векторы порождают различные лучи конуса C'. Это так для пары вектора $x'_i, x'_j, i \neq j$.

Предположим, что $x'_i = \alpha x'_{k,l}, \alpha > 0$, т. е. эти векторы порождают один и тот же луч. Тогда каждый из векторов x_k, x_l аннулирует те же $n-1$ линейно независимых строк матрицы A, что и x'_i. Отсюда $x'_i = \beta x_k, \beta > 0, x'_i = \gamma x_l, \gamma > 0$ (например, $\gamma < 0$ и заострённость конуса C влекут либо $\beta = 0$, либо $x_k = 0$). Это даёт $x_l = (\beta/\gamma) x_k$, что противоречит утверждению а) леммы 6.2.

Теперь предположим, что один и тот же луч конуса C' порождается векторами $x'_{i,j}$ и $x'_{k,l}$, где пары x_i, x_j и x_k, x_l различны. Тогда $x'_{i,j} = \alpha x'_{k,l}, \alpha > 0$. Равенство $|ax_j|x_i + |ax_i|x_j = \alpha(|ax_l|x_k + |ax_k|x_l), \alpha > 0$, показывает, что вектор x_i обращает в нуль $n-2$ линейно независимых строк матрицы A, аннулируемых как x_k, так и x_l. Согласно утверждению б) леммы 6.2 либо $x_i = x_k$, либо $x_i = x_l$. Аналогично либо $x_j = x_k$, либо $x_j = x_l$. Для определённости считаем, что $x_i = x_k$. Тогда $x_j = x_l$. Таким образом, пары x_i, x_j и x_k, x_l совпадают. Это приводит к противоречию.

Остается показать, что любой крайний луч $\{x'\}$ конуса C' порождается одним из векторов, указанных в теореме 6.1. По лемме 6.1 ненулевой вектор x' аннулирует $n-1$ линейно независимых строк матрицы $A = \begin{pmatrix} A \\ a \end{pmatrix}$.

Сначала рассмотрим случай, когда $n-1$ линейно независимых строк, аннулируемых x', находятся в матрице A. Тогда вектор x' порождает крайний луч конуса C, т. е. $\{x'\} = \{x_i\}$ для некоторого i, и $x' = \alpha x_i, \alpha \geq 0$. Из $ax' > 0$ следует $ax_i \geq 0$. Это означает, что x_i — именно тот вектор, который указан в части а) теоремы 6.1.

Пусть вектор x' не аннулирует $n-1$ линейно независимых строк из матрицы A, т. е. он аннулирует $n-2$ линейно независимых строк из A и линейно независимую с ними строку a. Для вектора x' имеет место представление

$$x' = \sum_{j=1}^{r} \alpha_j x_j, \ \alpha_j \geq 0, \ j = \overline{1, r},$$

поскольку $x' \in C$. В этой сумме должно быть не меньше двух положительных коэффициентов α_j, так как в противном случае либо $x' = 0$, либо вектор x' аннулирует $n-1$ линейно независимых строк из A.

Пусть $\alpha_k > 0, \alpha_l > 0$. Тогда векторы x_k, x_l аннулируют те же $n-2$ линейно независимых строк матрицы A, что и x'. Подпространство, ортогональное к этим строкам, имеет размерность 2, а векторы x_k, x_l составляют его базис. Поэтому можно записать $x' = \beta_1 x_k + \beta_2 x_l$. Если $\beta_1 \leq 0$, то из $x' - \beta_1 x_k = \beta_2 x_l$ следует, что вектор x' аннулирует $n-1$ линейно независимых строк матрицы A. А это не так по предположению. Следовательно, $\beta_1 > 0$. Аналогично получаем неравенство $\beta_2 > 0$.

Теперь покажем, что $ax_k \neq 0, ax_l \neq 0$. Если $ax_k = 0$, то вектор x_k обращает в нуль $n-1$ линейно независимых строк матрицы $A' = \begin{pmatrix} A \\ a \end{pmatrix}$, аннулируемых x'. Следовательно, $x' = \gamma x_k$, и вектор x' аннулирует $n-1$ линейно независимых строк матрицы A. Последнее противоречит предположению. Аналогичное противоречие получается и для $ax_l = 0$.

Из $ax' = 0$ следует, что скалярные произведения ax_k, ax_l имеют противоположные знаки, и можно записать равенство $\beta_1|ax_k| - \beta_2|ax_l| = 0$. Следовательно, векторы x_k, x_l составляют именно ту пару,

которая указана в части б) теоремы 6.1. Легко проверить, что векторы $x' = \beta_1 x_k + \beta_2 x_l$ и $|ax_l|x_k + |ax_k|x_l$ порождают один и тот же луч конуса C'.

Теорема 6.2. Среди строк матрицы A, аннулируемых двумя различными векторами x_k, x_l из системы всех крайних направлений $x_1, x_2, \ldots, x_r$ конуса $C = \{x \mid Ax \geq 0, r(A) = n\}$, находятся $n-2$ линейно независимых строк тогда и только тогда, когда не существует вектора x_i $(i = \overline{1,r},\ i \neq k,\ i \neq l)$, который обращает в нуль строки матрицы A, аннулируемые как x_k, так и x_l.

Доказательство. Необходимость условия этой теоремы следует из утверждения б) леммы 6.2.

Достаточность. Через a_{i_s}, $s = \overline{1,t}$, обозначим строки матрицы A, которые аннулируются как x_k, так и x_l. При этом никакой третий вектор x_i, $i \neq k \wedge i \neq l$ из системы $x_1, x_2, \ldots, x_r$ не обращает в нуль эти строки.

Рассмотрим линейное подпространство L пространства R_n, натянутое на строки a_{i_s}, $s = \overline{1,t}$, подпространство $\bar{L}$, ортогональное к L, и конус $K = C \bigcap \bar{L}$. Конус K может быть представлен в виде $K = \{x \mid Ax \geq 0,\ a_{i_s}x \leq 0,\ s = \overline{1,t}\}$. По теореме 6.1 система всех крайних направлений конуса K состоит из тех векторов системы $x_1, x_2, \ldots, x_r$, которые аннулируют все строки s_{i_s}, $s = \overline{1,t}$, т. е. только из двух векторов x_k, x_l (по условию теоремы). Таким образом, $K = \{x_k\} + \{x_l\}$.

Через z обозначим произвольный вектор, ортогональный к векторам x_k, x_l. Следовательно, $xz = 0$, $x \in K$. По теореме Минковского — Фаркаша z есть линейная комбинация строк матрицы A и строк $(-a_{i_s},\ s = \overline{1,t})$, с неотрицательными коэффициентами. Из $zx_k = zx_l = 0$ следует, что в этой линейной комбинации коэффициенты при всех строках, не аннулируемых x_k или x_l равны нулю. Следовательно, $z \in L$. Таким образом, подпространство, ортогональное к подпространству, натянутому на векторы x_k, x_l содержится в L. По утверждению а) леммы 6.2 размерность подпространства, натянутого на векторы x_k, x_l равна 2. Тогда размерность ортогонального подпространства равна $n-2$, а размерность подпространства L не меньше $n-2$. Это означает, что среди строк a_{i_s}, $s = \overline{1,t}$, существует по крайней мере $n-2$ линейно независимых строк [59].

УПРАЖНЕНИЯ

6.1. Составить схему l-й ветви параллельного алгоритма умножения двух матриц для случая, когда между процессорами происходит обмен столбцами. Исключить предположение, что каждое из чисел m и n делится на p, и внести в схему вызванные ими изменения.

6.2. В последовательности элементов $a_1, a_2, \ldots, a_n$ существует только один элемент a_k со свойством E для некоторого k из множества $\{1, 2, \ldots, n\}$. Поиск элемента a_k со свойством E осуществляется путём проверки всех элементов a_i в порядке $i = 1, 2, \ldots, k$. Считаем, что все n случаев равновозможны.

Показать, что имеет место неравенство $t_1/t_p < p$. Здесь t_1 и t_p — средние значения времени поиска элемента a_k со свойством E соответственно на одном и p процессорах.

6.3. Показать, что неравенства

$$\sum_{j \in J} x_j \geq 1, \quad \sum_{j \in J} (\lceil y_{i,j} \rceil - \lfloor y_{i,j} \rfloor) x_j \geq \lceil y_{i,0} \rceil - \lfloor y_{i,0} \rfloor$$

являются сечениями для ЦЛ-задачи, где J — множество индексов небазисных переменных, а базисное решение является нецелочисленным (см. § 6.2).

6.4. Связь между симплекс-таблицами. Рассмотрим прямую и двойственную к ней задачи

$$\max\{cx \mid Ax = b,\ x \geq 0\}, \quad \min\{yb \mid yA \geq c\},$$

где $m \times n$-матрица A имеет полный строковый ранг $(m \leq n)$.

Показать, что при разбиении $A = (BN)$ прямо (двойственно) допустимой симплекс-таблице прямой задачи соответствует двойственно (прямо) допустимая симплекс-таблица двойственной задачи.

6.5. Рассмотрим базисное представление Л-задачи

$$x_{B_i} = y_{i,0} - \sum_{j \in J} y_{i,j} x_j,\ i = \overline{0,m}.$$

Указать формулы преобразования этого представления для прямого и двойственного симплекс-алгоритмов, а так же для лексикографического варианта этих алгоритмов.

6.6. Попытайтесь восстановить отдельные вычислительные детали алгоритмов, описанных в § 6.3, без обращения к другим источникам.

6.7. Решить алгоритмом секущих плоскостей следующую конкретную СЦЛ-задачу. Максимизировать $-x_1-4x_2-2y_1-3y_2$ при условиях $x_1-3x_2+y_1-2y_2 \le -2$, $-x_1-3x_2-y_1-y_2 \le -3$, $x_j \ge 0 \wedge x_j \in Z$, $j=\overline{1,2}$, $y_j \ge 0$, $j=\overline{1,2}$.

6.8. Для конуса, заданного неравенствами $x_1 \ge 0, x_2 \ge 0, x_3 \ge 0, x_1 - x_2 \ge 0, -x_2 + x_3 \ge 0$, алгоритмом из § 6.4 найти все его крайние направления, исключив нормировку векторов. Геометрически истолковать проделанные вычисления. Изобразить процесс поиска на рисунке.

6.9. Используя алгоритм из § 6.5, найти все вершины множества допустимых решения Л-задач. Изобразить множество допустимых решений каждой Л-задачи на рисунке.

1. Минимизировать $-x_1 - x_2$ при условиях $x_1 \le 1$, $x_2 \le 1$, $x_j \ge 0$, $j=\overline{1,2}$.
2. Минимизировать $-x_3$ при условиях $x_1 + x_2 + x_3 \le 1$, $x_j \ge 0$, $j=\overline{1,3}$.
3. Минимизировать $-x_1 - x_2 - x_3$ при условиях $x_1 + x_2 - x_3 \le 0$, $x_1 + x_2 + x_3 \le 2$, $x_j \ge 0$, $j=\overline{1,3}$.
4. Минимизировать $-x_3$ при условиях $x_1 + x_3 \le 1$, $x_2 + x_3 \le 1$, $x_j \ge 0$, $j=\overline{1,3}$.
5. Минимизировать $-x_1 - x_2 - x_3$ при условиях $x_1 + x_3 \le 2$, $x_2 + x_3 \le 2$, $x_1 - x_3 \le 0$, $x_2 - x_3 \le 0$, $x_j \ge 0$, $j=\overline{1,3}$.
6. Минимизировать $-x_3$ при условиях $x_1 - x_2 + x_3 \le 1$, $x_1 + x_2 + 3x_3 \le 3$, $-x_1 + x_2 + x_3 \le 1$, $x_j \ge 0$, $j=\overline{1,3}$.
7. Минимизировать $-x_1 - x_2 - x_3$ при условиях $x_1 - x_2 + x_3 \le 2$, $x_1 + x_2 + 3x_3 \le 6$, $-x_1 + x_2 + x_3 \le 2$, $-x_1 + x_2 + x_3 \ge 0$, $x_1 + x_2 - 3x_3 \le 0, x_1 - x_2 + x_3 \ge 0$, $x_j \ge 0$, $j=\overline{1,3}$.
8. Минимизировать $x_1 + x_2 - 2x_3$ при условиях $x_1 + x_2 + x_3 \le 2$, $x_1 \le 1$, $x_2 \le 1$, $x_3 \le 1$, $x_j \ge 0$, $j=\overline{1,3}$.

6.10. Решить две конкретные ЦЛ-задачи прямым алгоритмом из § 6.6.

А. Минимизировать $7x_1+3x_2+4x_3$ при условиях $x_1+2x_2+3x_3 \ge 8$, $3x_1+x_2+x_3 \ge 5$, $x_j \ge 0 \wedge x_j \in Z$, $j=\overline{1,3}$.

Б. Минимизировать $2x_1+x_2$ при условиях $x_1+x_2 \le 5$, $-x_1+x_2 \le 0$, $9x_1+2x_2 \le 21$, $x_j \ge 0 \wedge x_j \in Z$, $j=\overline{1,2}$.

Решить эту задачу двумя способами: сначала перейти к задаче минимизации, а затем применить алгоритм из § 6.6; применить алгоритм, который аналогичен алгоритму из § 6.6, но находит максимум целевой функции.

6.11. Составьте параллельные вычислительные схемы, реализующие отдельные части прямого алгоритма из § 6.6.

6.12. Попытайтесь найти другое доказательство фактов, изложенных в этой главе.

Решение некоторых упражнений

6.4. Под симплекс-таблицей понимаем $(r+1)\times(s+1)$-матрицу $S=(s_{i,j})$, $i=\overline{0,r}$, $j=\overline{0,s}$. Симплекс-таблица служит для выражения r базисных переменных через s небазисных.

Построим симплекс-таблицу для прямой задачи. Из уравнения $Bx_B+Nx_N=b$ выражаем базисные переменные x_b через небазисные x_N, т.е. $x_B = B^{-1}b - B^{-1}Nx_N$. Тогда

$$x_0 = c_B x_B + c_N x_N = c_B(B^{-1}b - B^{-1}Nx_N) + c_N x_N =$$

$$= c_B B^{-1} b - (c_B B{-}1N - c_N)x_N.$$

Симплекс-таблица S имеет вид

$$S = \begin{pmatrix} c_B B^{-1} b & c_B B^{-1} N - c_N \\ B^{-1} b & B^{-1} N \end{pmatrix}, \text{ а } (x_0, x_B) = S(1, -x_N).$$

Теперь преобразуем двойственную задачу в задачу с неотрицательными переменными. Система неравенств $yA \ge c$ эквивалентна условиям $yA - z = c, z \ge 0$. Из подсистемы $yB - z_B = c_B$ находим вектор $y = c_B B^{-1} + z_B B^{-1}$, а затем подставляем его в целевую функцию и в оставшуюся подсистему. Тогда получаем

$$z_0 = yb = (c_B B^{-1} + z_B B^{-1})b = c_B B^{-1} b - z_B(-B^{-1}b),$$

$$z_N = yN - c_N = (c_B B^{-1} + z_B B^{-1})N - c_N =$$

$$= c_B^{-1} N - c_N - z_B(-B^{-1}N).$$

Симплекс-таблица S' имеет вид

$$S' = \begin{pmatrix} c_B B^{-1} b & c_B B^{-1} N - c_N \\ B^{-1} b & B^{-1} N \end{pmatrix}, \text{ а } (z_0, z_N) = (1, -z_B) S'.$$

Для прямо допустимой симплекс-таблицы прямой метод задачи выполняется неравенство $B^{-1}b \geq 0$, а для двойственно допустимой — $c_B B^{-1} N - c_N \geq 0$. Из сравнения симплекс-таблиц S и S' следует требуемое соответствие.

6.8. Векторы $(1,0,0), (0,0,1), (1,1,1)$ задают систему всех крайних направлений рассматриваемого конуса.

Глава 7

ПРИВЕДЕНИЕ ЦЕЛОЧИСЛЕННОЙ МАТРИЦЫ К СПЕЦИАЛЬНОМУ ВИДУ

Для алгоритмов Евклида с положительным остатком и с наименьшим остатком устанавливаются оценки числа их итераций. Описан способ нахождения малых по абсолютному значению множителей в представлении наибольшего общего делителя.

Использование множителей из представления наибольшего общего делителя в алгоритмах приведения целочисленной матрицы к специальному виду (в алгоритмах с номером 2) уменьшает число выполняемых арифметических операций. Приведение целочисленной матрицы осуществляется, не выходя из множества всех целых чисел.

Все рассматриваемые алгоритмы детально описаны и полностью обоснованы. Для отдельных алгоритмов указываются число ячеек памяти и число арифметических операций, необходимые для их выполнения. По этим двум параметрам сравниваются различные алгоритмы, дающие один и тот же результат. Наконец, обсуждаются особенности машинной реализации и схемы распараллеливания.

7.1 ОЦЕНКА ЧИСЛА ИТЕРАЦИЙ АЛГОРИТМОВ ЕВКЛИДА

Говоря о делителях целого числа, будем рассматривать только положительные делители этого числа. Всякое положительное целое, делящее одновременно целые числа $a_1, a_2, \ldots, a_n$, называется их общим делителем. Наибольший из общих делителей называется наибольшим общим делителем и обозначается через $\gcd(a_1, a_2, \ldots, a_n)$. Ввиду конечности числа общих делителей существование наибольшего общего делителя очевидно.

Если $\gcd(a_1, a_2, \ldots, a_n) = 1$, то числа $a_1, a_2, \ldots, a_n$ называются взаимно простыми. Если каждое из чисел $a_1, a_2, \ldots, a_n$ взаимно простое с каждым другим из них, то числа $a_1, a_2, \ldots, a_n$ называются попарно простыми. Очевидно, что попарно простые числа всегда и взаимно простые. В случае же двух чисел понятия "попарно простые" и "взаимно простые" совпадают.

Сначала рассмотрим наибольший общий делитель двух целых чисел. Для нахождения наибольшего общего делителя, а также для вывода его важнейших свойств применяется алгоритм Евклида.

Алгоритм Евклида с положительным остатком. Пусть a и b — положительные целые числа.

Находим последовательность равенств

$$\begin{array}{ll} a = q_1 b + r_1, & 0 < r_1 < b, \\ b = q_2 r_1 + r_2, & 0 < r_2 < r_1, \\ r_1 = q_3 r_2 + r_3, & 0 < r_3 < r_2, \\ r_2 = q_4 r_3 + r_4, & 0 < r_4 < r_3, \\ \dots & \dots \\ r_{k-2} = q_k r_{k-1} + r_k, & 0 < r_k < r_{k-1}, \\ r_{k-1} = q_{k+1} r_k + 0, & 0 = r_{k+1} < r_k. \end{array} \tag{7.1}$$

Эта последовательность заканчивается, когда остаток r_{k+1} обращается в нуль. Последнее обязательно происходит, поскольку убывающая последовательность положительных целых чисел $b, r_1, r_2, \dots, r_k$ не может содержать больше b членов.

Наибольший общий делитель чисел a и b равен r_k, т, е. последнему не равному нулю остатку алгоритма Евклида. Покажем это.

Если $a = qb + c, b > 0$, то совокупность общих делителей чисел a и b совпадает с совокупностью общих делителей чисел b и c, в частности, $\gcd(a, b) = \gcd(b, c)$ (для $c = 0$ имеем $\gcd(b, 0) = b$). Используя это утверждение $k + 1$ раз для последовательности равенств (7.1), получаем, что совокупность общих делителей чисел a и b совпадает с совокупностью делителей остатка r_k и, в частности, $\gcd(a, b) = r_k$.

Для a и b можно указать такие целые числа m и n, что $\gcd(a, b) = ma + nb$. Числа m и n называются множителями в представлении наибольшего общего делителя $\gcd(a, b)$. Их можно найти, например, последовательно выразив остатки $r_1, r_2, \dots, r_k$ через a и b. Конечно, множители m и n определяются неоднозначно.

Под итерацией алгоритма Евклида понимаем представление числа a в виде $a = qb + c$. Для реализации одной итерации нужно выполнить одно деление, одну операцию нахождения целой части, одно умножение и одно вычитание ($c = a - \lfloor a/b \rfloor b$). Число итераций алгоритма Евклида с положительным остатком для нахождения $\gcd(a, b)$ равно $k + 1$, т. е. числу равенств в (7.1).

Теорема 7.1 (Ламе). Если для нахождения наибольшего общего делителя $\gcd(a, b)$ положительных целых чисел a и $b, b < a$, используется алгоритм Евклида с положительным остатком, то число его итераций не превышает $5p$, где p — число десятичных разрядов в b. Эта оценка достижима.

Доказательство. Для чисел a и b рассмотрим последовательность равенств (7.1). В этих равенствах $q_1 \geq 1, q_2 \geq 2, q_3 \geq 1, q_4 \geq 1, \dots, q_k \geq 1$, но $q_{k+1} \geq 2$, так как $r_k < r_{k-1}$.

Рассмотрим последовательность чисел Фибоначчи, $u_1 = 1, u_2 = 2, u_i = u_{i-1} + u_{i-2}, i = 3, 4, 5, \dots$. Просматривая равенства (7.1) снизу вверх, последовательно получаем неравенства

$$\begin{array}{c} r_k \geq 1 = u_1 \to r_k \geq u_1, \\ r_{k-1} \geq 2 \cdot r_k + 0 \geq 2 \cdot 1 + 0 = u_2 \to r_{k-1} \geq u_2, \\ r_{k-2} \geq 1 \cdot r_{k-1} + r_k \geq u_2 + u_1 = u_3 \to r_{k-2} \geq u_3, \\ \dots \\ r_2 \geq 1 \cdot r_3 + r_4 \geq u_{k-2} + u_{k-3} = u_{k-1} \to r_2 \geq u_{k-1}, \\ r_1 \geq 1 \cdot r_2 + r_3 \geq u_{k-1} + u_{k-2} = u_k \to r_1 \geq u_k, \\ b \geq 1 \cdot r_1 + r_2 \geq u_k + u_{k-1} = u_{k+1} \to b \geq u_{k+1}. \end{array} \tag{7.2}$$

Чтобы определить верхнюю границу для величины $k + 1$, рассмотрим неравенство $u_{k+1} \leq b$.

Числа Фибоначчи могут быть ограничены снизу степенями числа g, т. е. $u_i > g^{i-1}$, $i = 2, 3, 4, \dots$. Величина $g = (1 + \sqrt{5})/2$ является положительным корнем квадратного уравнения $y^2 = y + 1$. Можно доказать существование нижней оценки по индукции, предварительно убедившись, что она верна для $i = 2, 3$.

Неравенство $u_{k+1} \leq b$ даёт $g^k < b, k < \log_{10} b / \log_{10} g$, Если p — число десятичных разрядов в b, то $b < 10^p$ и $\log_{10} b < p$. Учитывая, что $\log_{10} g > 1/5$, получаем $k < 5p$, т. е. $k + 1 \leq 5p$.

Оценка $k + 1 \leq 5p$ достигается, например, для чисел $a = 13, b = 8$.

Замечание 1. Хотя оценка $k + 1 \leq 5(\lfloor \log_{10} b \rfloor + 1)$ является неулучшаемой, вычислительный эксперимент показывает, что среднее число итераций алгоритма Евклида с положительным остатком для нахождения $\gcd(a, b)$ равно $1.9405(\lfloor \log_{10} b \rfloor + 1)$.

Теорема 7.1 обобщается на случай более двух чисел.

Теорема 7.2. Если для нахождения наибольшего общего делителя $\gcd(a_1, a_2, \ldots, a_n)$, положительных целых чисел $a_1, a_2, \ldots, a_n$ используется алгоритм Евклида с положительным остатком по схеме

$$d_2 = \gcd(a_2, a_1), \quad d_i = \gcd(a_i, d_{i-1}),\ i = \overline{3, n},$$

то число его итераций не превышает $n - 2 + 5p$, где p — число десятичных разрядов в a_1, $a_1 = \min\{a_1, a_2, \ldots, a_n\}$ $(n > 2)$. Эта оценка достижима [56].

Доказательство. Для чисел a_1 и a_2 первое применение алгоритма Евклида даёт последовательность равенств (7.1), в которой $a = a_2$, $b = a_1$, $k = k_2$, $q_i \geq 1$, $i = \overline{1, k_2}$, но $q_{k_2+1} \geq 2$.

Наибольший общий делитель $d_2 = \gcd(a_2, a_1) = r_{k_2}$ используется для вычисления $\gcd(a_3, r_{k_2})$. Первая итерация второго применения алгоритма Евклида есть $a_3 = p_3 r_{k_2} + r_{k_2+1}$, $0 \leq r_{k_2+1} < r_{k_2}$. Эта итерация даёт слагаемое 1 в члене $n-2$ оценки. Если $r_{k_2+1} = 0$, то $d_3 = d_2$, $k_3 = k_2$, и второе применение алгоритма Евклида на этом заканчивается.

Предположим, что $r_{k_2+1} > 0$. Тогда перепишем равенство $r_{k_2-1} = q_{k_2+1} r_{k_2}$, полученное на последней итерации первого применения алгоритма Евклида, в виде

$$r_{k_2-1} = (q_{k_2+1} - 1) r_{k_2} + r_{k_2+1} + (r_{k_2} - r_{k_2+1}),$$

где $r_{k_2} - r_{k_2+1} > 0$. Продолжим последовательность равенств и запишем её новые члены

$$r_{k_2+1} = q_{k_2+2} r_{k_2+1} + r_{k_2+2}, \ldots, r_{k_3-2} = q_{k_3} r_{k_3-1} + r_{k_3},$$

$$r_{k_3-1} = q_{k_3+1} r_{k_3}.$$

При этом $q'_{k_2+1} = q_{k_2+1} - 1 \geq 1$, $q_{k_2+2} \geq 1, \ldots, q_{k_3} \geq 1$, но $q_{k_3+1} \geq 2$.

Аналогично поступив еще $n-3$ раз $(n-3 \neq 0)$, получим последовательность, которая заканчивается равенствами

$$r_{k_n-2} = q_{k_n} r_{k_n-1} + r_{k_n}, \quad r_{k_n-1} = q_{k_n+1} r_{k_n}.$$

При этом $q_{k_n} \geq 1$, но $q_{k_n+1} \geq 2$. Эта последовательность равенств даёт последовательность неравенств (7.2), в которой $b = a_1, k = k_n$. Чтобы определить верхнюю границу для величины $k_n + 1$, рассмотрим неравенство $u_{k_{n+1}} \leq a_1$. С этого места, повторив рассуждения теоремы 7.1, найдем, что $k_n + 1 \leq 5p$. Заметим, что член $n - 2$ оценки уже получен.

Оценка достигается, например, для чисел 8, 13, 15, а также для чисел 9, 24, 26. Теорема 7.2 доказана.

Рассмотрим два положительных целых числа a и $b, b < a$. Пусть число a не делится на b. Тогда для числа a имеет место представление $a = qb + r, 0 < r < b$. Можно выполнить деление числа a на b и с отрицательным остатком. В этом случае представление числа a имеет вид $a = (q + 1)b - (b - r)$, $0 < b - r < b$. Оба случая записываются в виде $a = qb + \epsilon r$, где $0 < r < b$, $\varepsilon = \pm 1$, а q и r зависят от ε.

Общий алгоритм Евклида. Схема общего алгоритма Евклида для чисел a и b имеет вид

$$\begin{array}{rl}
a = q_1 b + \varepsilon_1 r_1, & 0 < r_1 < b, \\
b = q_2 r_1 + \varepsilon_2 r_2, & 0 < r_2 < r_1, \\
r_1 = q_3 r_2 + \varepsilon_3 r_3, & 0 < r_3 < r_2, \\
r_2 = q_4 r_3 + \varepsilon_4 r_4, & 0 < r_3 < r_2, \\
\ldots & \ldots \\
r_{k-2} = q_k r_{k-1} + \varepsilon_k r_k, & 0 < r_k < r_{k-1}, \\
r_{k-1} = q_{k+1} r_k + 0, & 0 = r_{k+1} < r_k,
\end{array}$$

$$\varepsilon_i = \pm 1,\ i = \overline{1, k}.$$

Представим все алгоритмы Евклида, например, для чисел 5 и 4:

$$\begin{array}{llll}
5 = 1 \cdot 4 + 1, & 5 = 2 \cdot 4 - 3, & 5 = 2 \cdot 4 - 3, & 5 = 2 \cdot 4 - 3, \\
4 = 4 \cdot 1 + 0, & 4 = 1 \cdot 3 + 1, & 4 = 2 \cdot 3 - 2, & 4 = 2 \cdot 3 - 2, \\
 & 3 = 3 \cdot 1 + 0, & 3 = 1 \cdot 2 + 1, & 3 = 2 \cdot 2 - 1, \\
 & & 2 = 2 \cdot 1 + 0, & 2 = 2 \cdot 1 + 0.
\end{array}$$

Из двух величин r и $b - r$ в представлениях $a = qb + r$ и $a = (q + 1)b - (b - r)$ по крайней мере, одна не превышает $b/2$. Поэтому всегда возможно деление с остатком по модулю не большим $1/2b$. Такое

деление называется делением с наименьшим остатком. Алгоритм Евклида, на каждой итерации которого выполняется деление с наименьшим остатком, называется алгоритмом Евклида с наименьшим остатком и обозначается LRA.

Как видно, алгоритм LRA для чисел 5 и 4 по числу итераций не длиннее любого другого алгоритма Евклида. Это утверждение справедливо и для произвольных положительных целых чисел a и b. Приведем без доказательства теорему, говорящую об этом [56].

Теорема 7.3 (Кронекер). Пусть a и b — два положительных числа, $b < a$. Тогда алгоритм Евклида с наименьшим остатком для пары a и b по числу итераций не длиннее любого другого алгоритма Евклида для этой же пары.

Алгоритм Евклида с наименьшим остатком может иметь то же самое число итераций, как и некоторый другой алгоритм. Покажем это, например, для чисел 33 и 13:

$$\begin{array}{cc} & LRA \\ 33 = 3 \cdot 13 - 6, & 33 = 2 \cdot 13 + 7, \\ 13 = 2 \cdot 6 + 1, & 13 = 2 \cdot 7 - 1, \\ 6 = 6 \cdot 1 + 0, & 7 = 7 \cdot 1 + 0. \end{array}$$

Теперь оценим число итераций алгоритма Евклида с наименьшим остатком.

Теорема 7.4. Если для нахождения наибольшего общего делителя $\gcd(a, b)$ положительных целых чисел a и b, $b < a$, используется алгоритм Евклида с наименьшим остатком, то число его итераций не превышает $8/3(p + 1/2)$, где p — число десятичных разрядов в b.

Доказательство. Теорема 7.4 может быть доказана при помощи рассуждений, аналогичных рассуждениям теоремы 7.1. Поэтому ограничимся изложением лишь тех основных моментов, в которых доказательства этих теорем различаются.

Пусть $r_1, r_2, r_3, r_4, \ldots, r_k$ — величины, даваемые алгоритмом Евклида с наименьшим остатком для чисел a и b. Тогда имеют место неравенства

$$b \geq 2 \cdot r_1 + r_2, \quad r_1 \geq 2 \cdot r_2 + r_3, \quad r_2 \geq 2 \cdot r_3 + r_4, \ldots,$$

$$r_{k-2} \geq 2 \cdot r_{k-1} + r_k, \quad r_{k-1} \geq 2 \cdot r_k, \quad r_k \geq 1.$$

Рассматриваем последовательность чисел $u_1 = 1, u_2 = 2, u_i = 2u_{i-1} + u_{i-2}$, $i = 3, 4, 5, \ldots$, при этом $u_i > 2g^{i-2}$, $i = 3, 4, 5, \ldots$, $g = 1 + \sqrt{2}$ — положительный корень квадратного уравнения $y^2 = 2y + 1$. Заметим, что $\log_{10} g = 0.382\ldots, \log_{10} 2 = 0.301\ldots, 3/8 = 0.375 < \log_{10} g, 2\log_{10} g - \log_{10} 2 < 1/2$. Учитывая это, из неравенства $k + 1 < (p + 2\log_{10} g - \log_{10} 2)/\log_{10} g$ получаем искомую оценку.

Замечание 2. Число итераций алгоритма Евклида с наименьшим остатком не превосходит $\lfloor \log_2 b \rfloor + 1$.

В заключение приведем теорему о количестве алгоритмов Евклида.

Теорема 7.5. Количество алгоритмов Евклида для двух взаимно простых положительных целых чисел a и $b, b < a$, не зависит от a и равно b.

Доказательство. Обозначим количество алгоритмов Евклида для чисел a и b через $\varphi(a, b)$. Легко проверить, что $\varphi(a, b) = b$ для $b = 1, 2, 3, 4$. Рассматривая для числа a два его представления: $a = qb + r$ и $a = (q+1)b - (b-r)$, $0 < r < b$, получаем, что $\varphi(a, b) = \varphi(b, r) + \varphi(b, b-r)$. К слагаемым правой части этого равенства уже применимо индукционное предположение.

7.2 МНОЖИТЕЛИ В ПРЕДСТАВЛЕНИИ НАИБОЛЬШЕГО ОБЩЕГО ДЕЛИТЕЛЯ

Во многих случаях требуется найти не только наибольший общий делитель $\gcd(a_1, a_2, \ldots, a_n)$ целых чисел $a_1, a_2, \ldots, a_n$, но и целочисленные множители $x_1, x_2, \ldots, x_n$ в его представлении

$$\gcd(a_1, a_2, \ldots, a_n) = \sum_{i=1}^{n} x_i a_i.$$

Вычислительная схема без обратного хода. В этой схеме для сохранения информации о проделанных вычислениях к столбцу a положительных целых чисел $a_1, a_2, \ldots, a_n$ присоединяется единичная

матрица I_n, порядка n и в качестве исходной рассматривается матрица $A = (aI_n)$ (часто используемый прием).

Шаг 1. Среди $a_1, a_2, \ldots, a_n$ находим наименьшее ненулевое число. Строку матрицы A, в которой находится это наименьшее число, называем преобразующей.

Шаг 2. Выбираем любую другую строку с ненулевым первым элементом и называем её преобразуемой строкой. Если нет преобразуемой строки, то процесс заканчивается.

Шаг 3. К преобразуемой строке прибавляем преобразующую строку, умноженную на такое отрицательное целое число, что первый элемент преобразованной строки остается неотрицательным и становится строго меньше первого элемента преобразующей строки. Возвращаемся к шагу 1 [56].

После окончания процесса первый элемент преобразующей строки есть $\gcd(a_1, a_2, \ldots, a_n)$, а остальные элементы этой строки — искомые множители. Можно обосновать этот алгоритм, использовав свойства наибольшего общего делителя и рассмотрев систему уравнений $a = I_n y$.

Теперь конкретизируем этот алгоритм, точно определив на шаге 2 выбор преобразуемой строки. Пусть a_1 — наименьшее целое среди $a_1, a_2, \ldots, a_n$. Тогда алгоритм можно осуществить как последовательное вычисление $d_2 = \gcd(a_1, a_2)$, $d_i = \gcd(d_{i-1}, a_i)$, $i = \overline{3,n}$. Пусть k_i — число итераций для вычисления $\gcd(d_{i-1}, a_i)$ и $k = \sum\limits_{i=2}^{n} k_i$. Тогда для выполнения рассматриваемого алгоритма требуется k делений, $k(n+1)$ умножений, $k(n+1)$ сложений и $n(n+1)$ ячеек памяти (исходный столбец a не сохраняется). Заметим, что $k \geq n-1$.

Вычислительная схема с обратным ходом. В этой схеме многократно решается задача нахождения наибольшего общего делителя и его множителей для двух целых чисел. Укажем один из способов решения этой задачи.

Пусть требуется найти $\gcd(c_1, c_2)$ и целочисленные множители y, z в его представлении $\gcd(c_1, c_2) = yc_1 + zc_2$. Рассмотрим матрицу $\begin{pmatrix} c_1 & 1 \\ c_2 & 0 \end{pmatrix}$. Над этой матрицей выполним преобразования по слегка измененной вычислительной схеме без обратного хода ($n = 2$, присутствует один, а не два столбца). Тогда после окончания процесса ненулевой первый элемент строки даёт $\gcd(c_1, c_2)$, а второй элемент этой же строки — множитель y. Множитель z находится по формуле $z = (\gcd(c_1, c_2) - yc_1)/c_2$. Если k — число итераций алгоритма Евклида, то для решения рассматриваемой задачи требуется $k+1$ делений, $2k+1$ умножений и $2k+1$ сложений.

Алгоритм нахождения множителей в представлении наибольшего общего делителя

Шаг 1 (начало). Запомнив знаки, перейдем к абсолютным величинам заданных целых чисел $a_1, a_2, \ldots, a_n$. В дальнейшем будем считать, что все числа $a_1, a_2, \ldots, a_n$ неотрицательные. Пусть $a_k = \min\limits_{\substack{i=1 \\ a_i \neq 0}}^{n} a_i$. Меняем местами a_1 и a_k. Определяем $a_l = \min\limits_{\substack{i=2 \\ a_1 \neq a_i}}^{n} a_i$. Меняем местами a_2 и a_l.

Шаг 2 (прямой ход). Полагаем $d_1 = a_1$. Пусть $i = \overline{2,n}$. Для d_{i-1} и a_i вычисляем величины:

а) их наибольший общий делитель d_i (используется алгоритм Евклида с наименьшим остатком);

б) целочисленные множители y_i и z_i в представлении $d_i = y_i d_{i-1} + z_i a_i$;

в) величины $u_i = -a_i/d_i$ и $v_i = d_{i-1}/d_i$;

г) если $|y_i| > -u_i/2$, то $y_i' = y_i - u_i \lfloor \lfloor y_i/u_i \rfloor \rfloor$, $z_i' = (d_i - y_i' d_{i-1})/a_i$ и полагаем $y_i = y_i'$, $z_i = z_i'$.

Шаг 3 (обратный ход). Полагаем $x_n = z_n, \overline{y}_n = y_n$. Для $i = \overline{n-1, 2}$ вычисляем величины

$$x_i = z_i \overline{y}_{i+1} - v_i \left\lfloor \left\lfloor \frac{z_i \overline{y}_{i+1}}{v_i} \right\rfloor \right\rfloor, \quad \overline{y}_i = y_i \overline{y}_{i+1} - u_i \left\lfloor \left\lfloor \frac{z_i \overline{y}_{i+1}}{v_i} \right\rfloor \right\rfloor$$

Полагаем $x_1 = \overline{y_2}$.

Шаг 4 (окончание). Ставим на свои места перемещенные числа. Учитываем знаки заданных чисел a_i. Выполняем контроль вида $d_n = \sum\limits_{i=1}^{n} x_i a_i$.

Обоснование алгоритма. Целочисленные величины u_i, v_i удовлетворяют однородному уравнению $d_{i-1}y + a_i z = 0$. Легко проверить, что $z_i' = z_i - v_i \lfloor \lfloor y_/u_i \rfloor \rfloor$. В силу выбора u_i, v_i величины y_i', z_i' также являются целочисленными множителями в представлении d_i.

Приведем оценки для ненулевых множителей

$$|x_n| \le \frac{d_{n-1}}{2d_n}, \quad |x_{n-1}| \le \frac{1}{2}v_{n-1} = \frac{d_{n-2}}{2d_{n-1}}, \ldots,$$

$$|x_2| \le \frac{1}{2}v_2 = \frac{d_1}{2d_2} \quad (d_1 = a_1 > 0,\ k = 1,\ l = 2).$$

Эти соотношения следуют из формул, приведённых в алгоритме.

Для двух положительных целых a_1, a_2 описанный алгоритм даёт множители y_2 и z_2, удовлетворяющие условиям $|a_2| \le a_2/(2d_2)$, $z_2 \le a_1/(2d_2)$. Эти множители являются наименьшими в том смысле, что для любых других множителей x_1, x_2 имеет место неравенство $|x_1| \ge |y_2|$.

Для $n = 3$ рассматриваемый алгоритм уже не даёт наименьших в этом смысле множителей. Приведем пример, иллюстрирующий это. Для чисел $178, 288, 501$ рассматриваемый алгоритм даёт множители $-74, 44, 1$. Однако для наибольшего общего делителя 1 этих же чисел существуют и другие множители $19, -10, -1$, которые меньше первых. Заметим, что в двух местах алгоритма поступаем наилучшим образом: для двух чисел ищутся наименьшие множители (шаг 2, г); решение однородного уравнения используется для уменьшения множителей (шаг 3).

Для чисел $424, 444, 932, 22347$ укажем множители из двух представлений: 37, 15, 0, -1 и 122914, -117327, 0, -1. Второе из них получено без поправок, т. е. по формулам $x_n = z_n$, $\overline{y}_n = y_n$, $x_i = z_i\overline{y}_{i+1}$, $\overline{y}_i = y_i\overline{y}_{i+1}$ $(i = \overline{n-1, 2})$, $x_1 = \overline{y}_2$.

Сведения о процедуре. Здесь ограничимся лишь краткими сведениями о процедуре eucl 1. Эта процедура реализует алгоритм 1 для нахождения множителей в представлении наибольшего общего делителя и содержит ряд вычислительных деталей, опущенных при описании алгоритма.

Для заданных целых чисел $a_1, a_2, \ldots, a_n, n \ge 1$, процедура eucl 1 находит их наибольший общий делитель d_n, целочисленные множители $x_1, x_2, \ldots, x_n$ в представлении $d_n = \sum\limits_{i=1}^{n} x_i a_i$ и дополнительную информацию из семи чисел, состоящую из величин $y_2, z_2, u_2, v_2, k, l, \overline{y_3}$, для которых выполняются соотношения

$$a_k y_2 + a_l z_2 = d_2, a_k u_2 + a_l v_2 = 0, \quad \overline{y_3} d_2 + \sum_{\substack{i=1 \\ i \ne k,l}}^{n} x_i a_i = d_n.$$

В одних и тех же элементах $x[i]$ массива x сначала находятся $|a_i|$, затем z_i, и, наконец, множители x_i $(i = \overline{1, n})$. При этом $x[n+1] = y_2$, $x[n+2] = z_2$, $x[n+3] = u_2$, $x[n+4] = v_2$, $x[n+5] = k$, $x[n+6] = l$, $x[n+7] = \overline{y_3}$.

Дополнительная информация из семи чисел во многих случаях позволяет исключить повторное применение процедуры и расширяет ее использование. Алгоритм, реализованный процедурой eucl 1, требует выполнения небольшого числа арифметических операций и даёт, как правило, малые по абсолютному значению множители в представлении наибольшего общего делителя.

7.3 ЭЛЕМЕНТАРНЫЕ ПРЕОБРАЗОВАНИЯ ЦЕЛОЧИСЛЕННОЙ МАТРИЦЫ

Рассмотрим целочисленную $m \times n$-матрицу A. Под элементарными преобразованиями матрицы A понимаем следующие операции: умножение строки (столбца) матрицы A на -1, перестановку двух строк (столбцов) матрицы A и сложение двух строк (столбцов) матрицы A, одна (один) из которых умножается на целое число. Рассмотрим более детально элементарные преобразования матрицы A.

1. Умножение строки матрицы на -1. Строка a_k матрицы A умножается на -1, а остальные строки этой матрицы остаются без изменения. Это эквивалентно умножению слева A на $m \times m$-матрицу $U_k(-1) = (\overline{u}_{i,j})$, где

$$u_{i,j} = \begin{cases} -1, & i = k \wedge j = k, \\ 0, & i \ne j, \\ 1, & i = j \wedge i \ne k. \end{cases} \qquad i = \overline{1, m},\ j = \overline{1, m},$$

Рассматриваемая операция соответствует, например умножению k-го уравнения системы $Ax = b$ на -1.

2. Перестановка двух строк матрицы. Две строки a_k и a_l матрицы A меняются местами ($k \neq l$), а остальные строки этой матрицы остаются без изменения. Это эквивалентно умножению слева A на перестановочную $m \times m$-матрицу $U_{k,l}$, которая получается из единичной матрицы I_m перестановкой её строк с номерами k и l. Рассматриваемая операция соответствует, например, изменению нумерации уравнений системы $Ax = b$.

3. Сложение двух строк матрицы, одна из которых умножается на целое число. К строке a_k матрицы A прибавляется её строка a_l ($k \neq l$), умноженная на целое число α. Это эквивалентно умножению слева A на $m \times m$-матрицу $U_{k,l}(\alpha) = (u_{i,j})$, где

$$u_{i,j} = \begin{cases} \alpha, & i = k \wedge j = l \wedge k \neq l, \\ 0, & \neg((i = k \wedge j = l \wedge k \neq l) \vee i = j), \\ 1, & i = j. \end{cases} \quad i = \overline{1,m},\ j = \overline{1,m},$$

Рассматриваемая операция соответствует, например, прибавлению, к k-му уравнению системы $Ax = b$ её l-го уравнения, умноженного на α.

4. Умножение столбца матрицы на -1. Столбец a_l матрицы A умножается на -1, а остальные столбцы этой матрицы остаются без изменения. Это эквивалентно умножению справа A на $n \times n$-матрицу $V_l(-1) = (v_{i,j})$, где

$$v_{i,j} = \begin{cases} -1, & i = l \wedge j = l, \\ 0, & i \neq j, \\ 1, & i = j \wedge j \neq l. \end{cases} \quad i = \overline{1,n},\ j = \overline{1,n},$$

Рассматриваемая операция соответствует, например, переходу от неизвестных $x_1, x_2, \ldots, x_n$ системы $Ax = b$ к неизвестным $y_1, y_2, \ldots, y_n$, где $y_l = -x_l$, $y_j = x_j$, $j \neq l$, $j = \overline{1,n}$.

5. Перестановка двух столбцов матрицы. Два столбца a_k и a_l матрицы A меняются местами, а остальные столбцы этой матрицы остаются без изменения. Это эквивалентно умножению справа A на перестановочную $n \times n$-матрицу $V_{k,l}$, которая получается из единичной матрицы I_n перестановкой её столбцов с номерами k и l. Рассматриваемая операция соответствует, например, изменению нумерации неизвестных системы $Ax = b$, т. е. переходу от неизвестных $x_1, x_2, \ldots, x_n$ к неизвестным $y_1, y_2, \ldots, y_n$, где $y_k = x_l$, $y_l = x_k$, $y_j = x_j$, $j \neq k$, $j \neq l$, $j = \overline{1,n}$.

6. Сложение двух столбцов матрицы, один из которых умножается на целое число. К столбцу a_k матрицы A прибавляется её столбец $a_l (k \neq l)$, умноженный на целое число α. Это эквивалентно умножению справа A на $n \times n$-матрицу $V_{l,k}(\alpha) = (v_{i,j})$, где

$$v_{i,j} = \begin{cases} \alpha, & i = l \wedge j = k \wedge k \neq l, \\ 0, & \neg((i = l \wedge j = k \wedge k \neq l) \vee i = j), \\ 1, & i = j. \end{cases} \quad i = \overline{1,n},\ j = \overline{1,n},$$

Рассматриваемая операция соответствует, например, переходу от неизвестных $x_1, x_2, \ldots, x_n$ к неизвестным $y_1, y_2, \ldots, y_n$, т. е. преобразованию $x = V_{l,k}(\alpha)y$:

$$x_l = \alpha y_k + y_l,\ x_j = y_j,\ j \neq l,\ j = \overline{1,n}.$$

Преобразования 1, 2, 3 называются элементарными строковыми преобразованиями, а преобразования 4, 5, 6 — элементарными столбцовыми преобразованиями. Для каждого из элементарных преобразований 1–6 матрицы A существует обратное преобразование, которое также является элементарным. Матрицы $U_k(-1), U_{k,l}, U_{k,l}(\alpha), V_l(-1), V_{k,l}, V_{l,k}(\alpha)$ называются элементарными. Матрица, определитель которой равен либо -1, либо $+1$, называется унимодулярной. Любая элементарная матрица и произведение конечного числа элементарных матриц являются унимодулярными целочисленными матрицами. Верно и обратное утверждение. Всякая унимодулярная целочисленная матрица представима в виде произведения конечного числа элементарных матриц.

Можно выполнить перестановку двух строк (столбцов) матрицы A при помощи четырёх преобразований вида 1 и 3 (вида 4 и 6). Для строк это схематически представим следующим образом:

$$\begin{pmatrix} k \\ l \end{pmatrix} \to \begin{pmatrix} k \\ l-k \end{pmatrix} \to \begin{pmatrix} l \\ l-k \end{pmatrix} \to \begin{pmatrix} l \\ -k \end{pmatrix} \to \begin{pmatrix} l \\ k \end{pmatrix}$$

Предположим, что $a_{1,1} = 1$. Прибавим к строке a_i матрицы A её первую строку, умноженную на $-a_{i,1}$, $i = \overline{2,m}$. В результате выполнения этого преобразования переменная x_1 системы $Ax = b$ исключается из всех уравнений, кроме первого. Матрица U, задающая рассматриваемое преобразование, есть

$$U = \begin{pmatrix} 1 & 0 & 0 & \dots & 0 \\ -a_{2,1} & 1 & 0 & \dots & 0 \\ -a_{3,1} & 0 & 1 & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ -a_{m,1} & 0 & 0 & \dots & 1 \end{pmatrix}$$

Это пример неэлементарного преобразования матрицы. В общем случае (например, когда $a_{1,1} \neq 1$) подобное неэлементарное преобразование осуществляется с мощью множителей из представления наибольшего общего делителя.

Будем говорить, что матрицы A и B эквивалентны, и записывать это символом $A \sim B$, если от A можно перейти к B с помощью конечного числа элементарных преобразований. Легко показать, что так определённое бинарное отношение матриц является рефлексивным, симметричным и транзитивным. Множество всех целочисленных $m \times n$-матриц распадается на непересекающиеся классы эквивалентных матриц. Среди всех матриц, эквивалентных данной матрице A, находятся матрицы простого вида, такие как диагональные, треугольные, трапецеидальные и др.

7.4 АЛГОРИТМЫ ПРИВЕДЕНИЯ ЦЕЛОЧИСЛЕННОЙ МАТРИЦЫ К ТРАПЕЦЕИДАЛЬНОМУ ВИДУ

Теорема 7.6. Для заданной целочисленной $m \times n$-матрицы A существуют такие две унимодулярные целочисленные матрицы P и K, что матрица $T = PAK$ имеет трапецеидальный вид (рис. 7.1). Матрица P представляет собой произведение выполненных перестановок строк и задаётся m числами (перестановкой $p = (i_1, i_2, \dots, i_m)$).

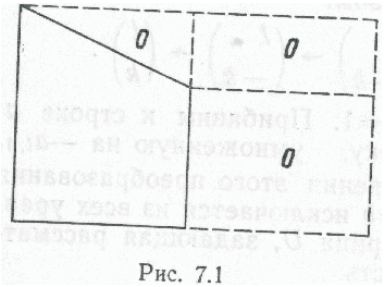


Рис. 7.1

Алгоритмы для приведения целочисленной матрицы к трапецеидальному виду в кольце целых чисел, рассмотренные ниже, доказывают теорему 7.6, строят матрицы P и K.

Алгоритм 1

Шаг 1 (начало). Выполняем операции $i := 1$, $s := m$, $p = (1, 2, \dots, m)$ (p — перестановка строк), $T := A$ (A — заданная $m \times n$-матрица), $K := I_n$ (I_n — единичная матрица n-го порядка).

Шаг 2. Если в строке i существует элемент $t_{i,j} \neq 0$, $i \leq j \leq n$, то переходим к шагу 3. Среди строк $i+1, i+2, \dots, s$ находим первую снизу строку h с элементом $t_{h,j} \neq 0$, $i \leq j \leq n$. Если такой строки h не существует, то переходим к шагу 7. Меняем местами строки i и h матрицы T. Меняем местами элементы i и h перестановки p. Полагаем $s = h - 1$.

Шаг 3. Определяем элемент $t_{i,q}$ из условия $|t_{i,q}| = \min\limits_{\substack{j=i, \\ t_{i,j} \neq 0}}^{n} |t_{i,j}|$. Умножаем столбец q матриц T и K на знак элемента $t_{i,q}$. Меняем местами столбцы i и q матриц T и K.

Шаг 4. Если $t_{i,i} \mid t_{i,j}$, $j = \overline{i+1, n}$, то переходим к шагу 5. Пусть $t_{i,q}$ — первый элемент, не делящийся на $t_{i,i}$ ($i + 1 \leq q \leq n$). Тогда, пока имеет место $t_{i,i} \nmid t_{i,q}$ над столбцами i и q матриц T и K выполняем следующие два элементарных преобразования:

$$t_q := t_q - \lfloor t_{i,q}/t_{i,i} \rfloor t_i, \quad k_q := k_q - \lfloor t_{i,q}/t_{i,i} \rfloor k_i;$$

меняем местами столбцы i и q матриц T и K. Переходим к началу шага 4.

Шаг 5. Над столбцами j матриц T и K выполняем элементарные преобразования

$$t_j := t_j - (t_{i,j}/t_{i,i})t_i,\ k_j := k_j - (t_{i,j}/t_{i,i})k_i \quad (j = \overline{i+1,n}).$$

Шаг 6. Если $i < m \wedge i < n$, то $i := i + l$ и переходим к шагу 2.

Шаг 7 (окончание).

Шаг 7.1. Для невырожденной квадратной матрицы A приводим получившуюся нижнюю треугольную матрицу к эрмитовому виду: $t_{i,i} > 0$, $t_{i,j} \le 0 \wedge |t_{i,j}| < t_{i,i}$ для $j = \overline{1,i-1}$, $t_{i,j} = 0$ для $j = \overline{i+1,n}$ $(i = \overline{1,n})$.

Шаг 7.2. Выполняем контроль вида $T = PAK$.

Алгоритм 2

Этот алгоритм отличается от предыдущего шагами $3', 4'$, в которых используются множители из представления наибольшего общего делителя.

Шаг $3'$. Для целых чисел $t_{i,i}, t_{i,i+1}, \ldots, t_{i,n}$ находим их наибольший общий делитель d_{n-i+1}, целочисленные множители $x_i, x_{i+1}, \ldots, x_n$ в его представлении $d_{n-i+1} = \sum\limits_{j=i}^{n} x_j t_{i,j}$ и семь величин y_2, z_2, u_2, v_2, k, l, $\overline{y_3}$, для которых выполняются соотношения

$$t_{i,k}y_2 + t_{i,l}z_2 = d_2, \quad t_{i,k}u_2 + t_{i,l}v_2 = 0,$$

$$\overline{y_3}d_2 + \sum_{\substack{j=1\\ j\ne k,l}}^{n} x_j t_{i,j} = d_{n-i+1} \quad (k \ne l,\ i \le k \le n,\ i \le l \le n).$$

Шаг $4'$.1. Последовательно рассматриваем четыре случая и находим номер q $(i \le q \le n)$. Если для некоторого q имеет место $x_q = 1 \vee t_{i,q} = 0$, то переходим к шагу $4'$.2. Случай $x_q = -1$ сводится к предыдущему умножением на -1 столбца q матриц T и K. Если для некоторого $q \ne k$ элемент $t_{i,q}$ делится на $t_{i,k}$, то $x_k := x_k - t_{i,q}/t_{i,k}$, и переходим к шагу $4'$.2.

Наконец, если не имеет места ни один из трёх вышеуказанных случаев, то над столбцами k и l матриц T и K выполняем операции $t_k := y_2 t_k + z_2 t_l$, $t_l := u_2 t_k + v_2 t_l$, $k_k := y_2 k_k + z_2 k_l$, $k_l := u_2 k_k + v_2 k_l$ (здесь k_k — k-й столбец матрицы K). Полагаем $q = l$, $x_k = \overline{y_3}$ и переходим к шагу $4'$.2.

Шаг $4'$.2. Над столбцами матриц T и K выполняем преобразования

$$t_q := t_q + \sum_{\substack{j=i\\ j\ne q, x_j\ne 0}}^{n} x_j t_j, k_q := k_q + \sum_{\substack{j=i\\ j\ne q, x_j\ne 0}}^{n} x_j k_j.$$

Меняем местами столбцы i и q матриц T и K.

Обоснование алгоритмов. Матрица P состоит не более чем из $\lfloor m/2 \rfloor$ множителей P_i и не зависит от их порядка. При этом $P_i = P_i^{-1}$ и, следовательно, $P = P^{-1}$. Это даёт, что $T = PAK$ эквивалентно $PT = AK$. Соотношение $PT = AK$ используется для контроля правильной работы процедур trap 1 и trap 2 [56].

Элемент $t_{i,i}$ на шаге 4 алгоритма 1 является строго убывающей положительной целочисленной величиной, ограниченной снизу числом 1. Это гарантирует конечность шага 4, а тем самым и конечность всего алгоритма.

После выполнения шага $4'$ алгоритма 2 получаем, что $t_{i,i} \mid t_{i,j}$, $j = \overline{i+1,n}$.

Первые два случая шага $4'$.1 очевидны. В случае, когда элемент $t_{i,q}$ делится на $t_{i,k}$, используемый вариант алгоритма Евклида даёт множитель $x_q = 0$. Величины $x'_j = x_j$ $(j \ne k, q)$, $x'_k = x_k - t_{i,q}/t_{i,k}$, $x'_q = 1$ также являются множителями.

Учитывая определение величин y_2, z_2, u_2, v_2 в четвёртом случае получаем $t'_{i,k} = d_2$, $t'_{i,l} = 0$. Наличие нулевых множителей сводит этот случай к первому.

Оценка числа операций. Пусть A — невырожденная квадратная матрица с целочисленными элементами. Тогда алгоритм 1 приводит её к эрмитовому виду. Оценим число умножений, которое выполняется алгоритмом 1 в этом случае.

Шаг 4 алгоритма 1 похож на вычислительную схему без обратного хода для нахождения наибольшего общего делителя d_{n-i+1} целых чисел $t_{i,i}, t_{i,i+1}, \ldots, t_{i,n}$ алгоритмом Евклида с положительным остатком. Через k_i обозначим число итераций алгоритма Евклида, в результате выполнения которых находится d_{n-i+1}. Оценим сверху число умножений на отдельных шагах алгоритма 1:

$$k_i(2n-i+1), \quad i = \overline{1, n-1} \quad \text{(шаг 4)};$$

$$(n-i)(2n-i+1), \quad i = \overline{1, n-1} \quad \text{(шаг 5)};$$

$$(i-1)(2n-i+1), \quad i = \overline{2, n} \quad \text{(шаг 7.1)};$$

$$n^2 \quad \text{(шаг 7.2)}.$$

Таким образом, число умножений не превышает

$$\sum_{i=1}^{n-1} k_i(2n-i+1) + \sum_{i=1}^{n-1}(n-i)(2n-i+1) + \sum_{i=1}^{n-1} i(2n-i) + n^2 =$$

$$= \sum_{i=1}^{n-1} k_i(2n-i+1) + \sum_{i=1}^{n-1}(n(2n+1) - i(n+1)) + n^2 =$$

$$= \frac{3}{2}n^3 - \frac{1}{2}n + \sum_{i=1}^{n-1} k_i(2n-i+1).$$

Заметим, что увеличение значений элементов матрицы, влечёт рост величин k_i.

Теперь рассмотрим шаг $4'.2$ алгоритма 2. Пусть p_i — число сложений двух столбцов, один из которых умножается на целое число. Величина p_i равна числу ненулевых множителей x_j Если на шаге $4'.1$ имеет место четвёртый случай, то следует учесть число ненулевых целых среди y_2, z_2, u_2, v_2. Следовательно, $p_i \le n - i + 4$. Как правило, величина p_i намного меньше k_i. Например, для строки i с двумя ненулевыми элементами $t_{i,i}$ и $t_{i,i+1}, |t_{i,i}| \le |t_{i,i+1}|$, среднее значение k_i равно $1{,}9405 \log_{10} |t_{i,i}|$, а $p_i = 2$ для $i < n-1$ и $p_i = 4$ для $i = n-1$. Для строки (13 26 47 50) имеем $p_i = 2, k_i = 5$.

Хотя оценка $n-i+4$ для p_i достигается, среднее значение p_i намного меньше этой оценки. Известен следующий результат: вероятность того, что два случайно взятых целых числа являются взаимно простыми, равна $6/\pi^2 (6/\pi^2 > 0{,}6)$ (Чезара [24]). Если $\gcd(t_{i,i}, t_{i,i+1}) = 1$, то $p_i = 2$. Считая значения элементов $t_{i,i}$ и $t_{i,i+!}$ случайно выбранными, получаем, что среднее значение p_i меньше $0{,}6 \cdot 2 + 0{,}4(n - i + 4) = 2{,}8 + 0{,}4(n-1)$ [56].

7.5 АЛГОРИТМЫ ПРИВЕДЕНИЯ ЦЕЛОЧИСЛЕННОЙ МАТРИЦЫ К НОРМАЛЬНОМУ ВИДУ

Теорема 7.7. Для заданной целочисленной $m \times n$-матрицы A существуют такие две унимодулярные целочисленные матрицы E и F, что диагональная матрица $D = EAF$ имеет нормальный вид: $d_{i,i} > 0$, $i = \overline{1, r}$, $d_{i,i} \mid d_{i+1,i+1}$, $i = \overline{1, r-1}$ (r — ранг матрицы A).

При помощи алгоритмов для приведения целочисленной матрицы к нормальному виду в кольце целых чисел, рассмотренных далее, докажем теорему 7.7 и построим матрицы E и F.

Алгоритм 1

Шаг 1 (начало). Выполняем операции $i := 1$, $s := m$, $T := A$ (A — заданная $m \times n$-матрица), $E := I_m$, $F := I_n$ (здесь I_m, I_n — единичные матрицы порядков m и n соответственно).

Шаг 2. Если в строке i существует элемент $t_{i,j} \ne 0$, $i \le j \le n$, то переходим к шагу 3. Среди строк $i+1, i+2, \ldots, s$ находим первую снизу строку с элементом $t_{h,j} \ne 0$, $i \le j \le n$. Если такой строки h не существует, то переходим к шагу 9. Меняем местами строки i и h матриц T и E. Полагаем $s = h - 1$.

Шаг 3. Над столбцами $i, i+1, \ldots, n$ матриц T и F выполняем элементарные столбцовые преобразования, чтобы получить $t_{i,i} \mid t_{i,j}$, $j = \overline{i+1, n}$.

Шаг 3.1. Определяем элемент $t_{i,q}$ из условия $|t_{i,q}| = \min\limits_{\substack{j=i \\ t_{i,j}\neq 0}}^{n} |t_{i,j}|$. Умножаем столбец q матриц T и F на знак элемента $t_{i,q}$. Меняем местами столбцы i и q матриц T и F.

Шаг 3.2. Если $t_{i,i} \mid t_{i,j}$, $j = \overline{i+1,n}$, то переходим к шагу 4. Пусть $t_{i,q}$ — первый элемент, не делящийся на $t_{j,i} (i+1 \le q \le n)$. Тогда, пока имеет место $t_{i,i} \nmid t_{i,q}$, над столбцами i и q матриц T и F выполняем следующие два элементарных преобразования:

$$t_q := t_q - \lfloor t_{i,q}/t_{i,i} \rfloor t_i, \quad f_q := f_q - \lfloor t_{i,q}/t_{i,i} \rfloor f_i;$$

меняем местами столбцы i и q матриц T и F. Переходим к началу шага 3.2.

Шаг 4. Если $t_{i,i} \mid t_{h,i}$, $h = \overline{i+1,s}$, то переходим к шагу 7.

Шаг 5. Аналогично шагу 3 над строками $i, i+1, \ldots, s$ матриц T и E выполняем элементарные строковые преобразования, чтобы получить $t_{i,i} \mid t_{h,i}$, $h = \overline{i+1,s}$.

Шаг 6. Если $t_{i,i} \mid t_{i,j}$, $j = \overline{i+1,n}$, то переходим к шагу 7, в противном случае — к шагу 3.

Шаг 7. Над столбцами j матриц T и F выполняем элементарные преобразования

$$t_j := t_j - (t_{i,j}/t_{i,i})t_i, f_j := f_j - (t_{i,j}/t_{i,i})f_i \quad (j = \overline{i+1,n}).$$

Над строками h матриц T и E выполняем элементарные преобразования

$$t_h := t_h - (t_{h,i}/t_{i,i})t_i, e_h := e_h - (t_{h,i}/t_{i,i})e_i \quad (h = \overline{i+1,s}).$$

Шаг 8. Если $i < m \wedge i < n$, то $i := i+1$, и переходим к шагу 2.

Шаг 9. На этом шаге добиваемся выполнения условия $t_{i,i} \mid t_{i+1,i+1}$, $i = \overline{1,r-1}$. Полагаем $i = 1$.

Шаг 9.1. Если $t_{i,i} \mid t_{j,j}$, $j = \overline{i+1,r}$, то переходим к шагу 9.3.

Шаг 9.2. Пусть k — первый номер, для которого $t_{i,i}$ не делит $t_{k,k}$, $k \ge i+1$. Выполняем элементарные строковые и столбцовые преобразования над матрицами E, F, T. Схематически это представим следующим образом.

Пока имеет место условие, что $t_{i,i}$ не делит $t_{k,k}$, выполняем шаги:

а) $\begin{pmatrix} t_{i,i} & 0 \\ 0 & t_{k,k} \end{pmatrix} \to \begin{pmatrix} t_{i,i} & q_1 t_{i,i} \\ 0 & t_{k,k} \end{pmatrix} \to \begin{pmatrix} t_{i,i} & q_1 t_{i,i} \\ -t_{i,i} & r_1 \end{pmatrix} \to \begin{pmatrix} -t_{i,i} & r_1 \\ t_{i,i} & q_1 t_{i,i} \end{pmatrix} \to \begin{pmatrix} r_1 & -t_{i,i} \\ q_1 t_{i,i} & t_{i,i} \end{pmatrix}$,
где $t_{k,k} = q_1 t_{i,i} + r_1$, $0 < r_1 < t_{i,i}$;

б) выполняем шаги 3–7 этого алгоритма, взяв в качестве исходной матрицу

$$\begin{pmatrix} r_1 & -t_{i,i} \\ q_1 t_{i,i} & t_{i,i} \end{pmatrix}$$

Если $t_{i.i}$ делит $t_{k,k}$, то переходим к шагу 9.1.

Шаг 9.3. Если $i < r-1$, то $i := i+1$ и переходим к шагу 9.1.

Шаг 10 (окончание). Выполняем контроль вида $T = EAF$.

Алгоритм 2

Этот алгоритм отличается от предыдущего шагами $3', 5', 9'$, в которых используются множители из представления наибольшего общего делителя. Такой алгоритм легко построить и обосновать.

Шаг $3'$. Используя множители, над столбцами $i, i+1, \ldots, n$ матриц T и F выполняем столбцовые преобразования, чтобы получить $t_{i,i} \mid t_{i,j}$, $j = \overline{i+1,n}$. Этот шаг аналогичен шагам $3'$ и $4'$ алгоритма 2 для приведения целочисленной матрицы к трапецеидальному виду в кольце целых чисел (см. § 7.4).

Шаг $9'$. Используя множители из представления наибольшего общего делителя, на этом шаге добиваемся выполнения условия $t_{i,i} \mid t_{i+1,i+1}$, $i = \overline{l,r-1}$. Полагаем $i = 1$.

Шаг $9'.1$. Если $t_{i,i} \mid t_{j,j}$, $j = \overline{i+1,r}$, то переходим к шагу $9'.3$.

Шаг $9'.2$. Пусть k — первый номер, для которого $t_{i,i}$ не делит $t_{k,k}$ $(k \ge i+1)$. Находим наибольший общий делитель $d_{i,k}$ чисел $t_{i,i}$ и $t_{k,k}$, множители x_i и x_k в представлении $d_{i,k} = x_i t_{i,i} + x_k t_{k,k}$. Используя найденные величины, выполняем столбцовые и строковые преобразования для матриц T, E, F. Схематически это представим следующим образом:

$$\begin{pmatrix} t_{i,i} & 0 \\ 0 & t_{k,k} \end{pmatrix} \to \begin{pmatrix} t_{i,i} & x_i t_{i,i} + x_k t_{k,k} \\ 0 & t_{k,k} \end{pmatrix} = \begin{pmatrix} t_{i,i} & d_{i,k} \\ 0 & t_{k,k} \end{pmatrix} \to$$

$$\to \begin{pmatrix} 0 & d_{i,k} \\ \frac{-t_{i,i}t_{k,k}}{d_{i,k}} & t_{k,k} \end{pmatrix} \to \begin{pmatrix} 0 & d_{i,k} \\ \frac{-t_{i,i}t_{k,k}}{d_{i,k}} & 0 \end{pmatrix} \to \begin{pmatrix} d_{i,k} & 0 \\ 0 & \frac{t_{i,i}t_{k,k}}{d_{i,k}} \end{pmatrix}.$$

Переходим к шагу 9′.1.

Шаг 9′.3. Если $i < r-1$, то $i := i+1$ и переходим к шагу 9′.1.

Обоснование этих алгоритмов аналогично обоснованию алгоритмов для приведения целочисленной матрицы к трапецеидальному виду.

7.6 АЛГОРИТМЫ НАХОЖДЕНИЯ ОБЩЕГО ЦЕЛОЧИСЛЕННОГО РЕШЕНИЯ СИСТЕМЫ ЛИНЕЙНЫХ УРАВНЕНИЙ

Для заданной системы линейных уравнений $Ax = b$ с целочисленными данными A, b при помощи алгоритмов 1 и 2 либо находим общее целочисленное решение $x = f + K_2 t$, либо устанавливаем, что эта система не имеет целочисленных решений. Рассматриваемые алгоритмы для нахождения общего целочисленного решения системы линейных уравнений основаны на приведении целочисленной матрицы A к трапецеидальному виду в кольце целых чисел. В них используются столбцовые преобразования и, возможно, перестановки строк матрицы.

Алгоритм 1

Шаг 1 (начало). Выполняем операции $i := 1$, $s := m$, $p := (1, 2, \ldots, m)$ (p — перестановка строк), $C := \begin{pmatrix} A & b \\ I_n & 0 \end{pmatrix}$ (A — $m \times n$-матрица коэффициентов системы уравнений, b — m-столбец свободных членов, I_n — единичная матрица n-го порядка) (рис. 7.2).

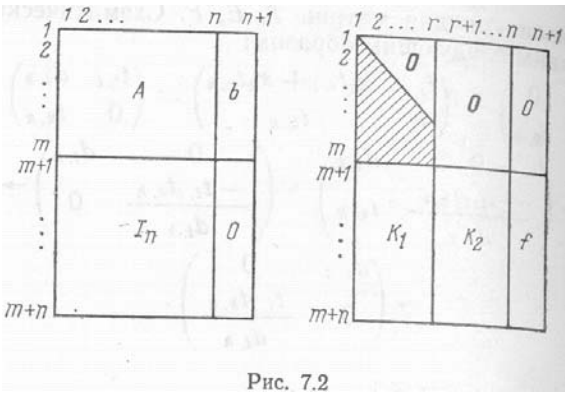


Рис. 7.2

Шаг 2. Если в строке i существует элемент $c_{i,j} \neq 0$, $i \leq j \leq n$, то переходим к шагу 3. Если $c_{i,n+1} \neq 0$, то система несовместна в поле всех рациональных чисел, и переходим к шагу 7.2. Среди строк $i+1, i+2, \ldots, s$ находим первую снизу строку h с элементом $c_{h,j} \neq 0$, $i \leq j \leq n$. Если такой строки h не существует, то переходим к шагу 7.1. Меняем местами строки i и h матрицы C. Меняем местами элементы i и h перестановки p. Полагаем $s = h - 1$.

Шаг 3. Над столбцами $i, i+1, \ldots, n$ матрицы C выполняем элементарные столбцовые преобразования, чтобы получить $c_{i,i} \mid c_{i,j}$, $j = \overline{i+1, n}$.

Шаг 3.1. Определяем элемент $c_{i,q}$ из условия $|c_{i,q}| = \min\limits_{\substack{j=i \\ c_{i,j} \neq 0}}^{n} |c_{i,j}|$. Умножаем столбец q матрицы C на знак элемента $c_{i,q}$. Меняем местами столбцы i и q матрицы C.

Шаг 3.2. Если $c_{i,i} \mid c_{i,j}$, $j = \overline{i+1, n}$, то переходим к шагу 4. Пусть $c_{i,q}$ — первый элемент, не делящийся на $c_{i,i}$ ($i+l \leq q \leq n$). Тогда, пока имеет место $c_{i,i} \nmid c_{i,q}$ над столбцами i и q матрицы C выполняем следующие два элементарных преобразования: $c_q := c_q - \lfloor c_{i,q}/c_{i,i} \rfloor c_i$; меняем местами столбцы i и q матрицы C. Переходим к началу шага 3.2.

Шаг 4. Если элемент $c_{i,n+1}$ не делится на $c_{i,i}$, то система не имеет целочисленных решений и переходим к шагу 7.2.

Шаг 5. Над столбцами j матрицы C выполняем элементарные преобразования $c_j := c_j - (c_{i,j}/c_{i,i})c_i$ $(j = \overline{l+1, n+1})$.

Шаг 6. Если $i < m \wedge i < n$, то $i := i + 1$ и переходим к шагу 2. При $i = m$ переходим к шагу 7.1. Если же $i < m \wedge i = n \wedge c_{h,n+1} \neq 0$ (для некоторого h, $n + 1 \leq h \leq m$), то система несовместна в поле всех рациональных чисел, и переходим к шагу 7.2.

Шаг 7 (окончание).

Шаг 7.1. Выполняем контроль вида

$$\sum_{j=1}^{n} a_{i,j} c_{m+j,n+1} = b_i, \quad i = \overline{1, m}.$$

Шаг 7.2. Выполняем контроль вида

$$\sum_{h=1}^{n} a_{i,h} c_{m+h+j} = c_{p_i,j}, \quad i = \overline{1, m},\ j = \overline{1, n}.$$

Алгоритм 2

Этот алгоритм отличается от предыдущего шагом $3'$, в котором используются множители из представления наибольшего общего делителя.

Обоснование алгоритмов. Предположим, что алгоритм не указывает на отсутствие целочисленных решений. Тогда после его выполнения получается матрица

$$C = \begin{pmatrix} T & 0 & 0 \\ K_1 & K_2 & f \end{pmatrix}$$ (см. рис. 7.2).

Представим себе, что преобразования над столбцом $n + 1$ матрицы C выполняются после приведения целочисленной матрицы A к трапецеидальному виду.

Рассмотрим преобразование

$$x = Ky = (K_1 K_2) \cdot \begin{pmatrix} y_1 \\ y_2 \end{pmatrix}.$$

Унимодулярная целочисленная матрица K устанавливает взаимно однозначное соответствие между целочисленными векторами x и y. Переходя к целочисленным переменным y, получаем эквивалентную систему Уравнений вида $Ty_1 + 0y_2 = b$. Нетрудно видеть, что общее целочисленное решение этой системы имеет вид $y = (\overline{y_1}, 0) + (0, y_2)$, где $\overline{y_1}$ — фиксированный целочисленный вектор, удовлетворяющий r первым уравнениям системы (r — ранг матрицы A), а y_2 — произвольный целочисленный вектор. Переходя обратно к переменным x получаем

$$x = K_1 \overline{y_1} + K_2 y_2 = f + K_2 y_2.$$

Случаи, когда система уравнений не имеет целочисленных решений, обосновываются аналогично.

После выполнения шага $3'$ алгоритма 2 получаем, что $c_{i,i} \mid c_{i,j}$, $j = \overline{i+1, n}$.

7.7 МАШИННАЯ РЕАЛИЗАЦИЯ И СХЕМЫ РАСПАРАЛЛЕЛИВАНИЯ

Для алгоритмов, рассмотренных в § 7.2, 7.4–7.6, на языке Алгол-60 были написаны процедуры eucl 1, trap 1 и trap 2, diag 1 и diag 2, diop 1 и diop 2. Счёт по этим процедурам проводился после их трансляции на трансляторах АЛГОЛ–БЭСМ-6 и АЛЬФА-6. В каждой из семи процедур предусмотрен контроль, выполнение которого гарантирует её правильную работу. Процедуры прошли тщательную проверку путём решения разнообразных тестовых задач. При счёте процедур в окончательной редакции отклонения от правильной работы не были обнаружены [56].

При решении конкретных задач получаются большие по модулю целые числа (больше, чем 10^{10}). При этом значения элементов промежуточных и конечных матриц очень быстро возрастают, когда увеличивается размер $\min\{m, n\}$ заданной матрицы. Это серьезная вычислительная трудность.

Выполнение арифметических операций над большими числами без потери младших разрядов требует, например, в рамках языка Алгол-60 создания специальных процедур или, возможно, специальных подпрограмм (написанных непосредственно в машинных командах), при помощи которых экономно осуществляется поразрядная обработка бинарных данных.

Распараллеливание. В алгоритмах приведения целочисленной матрицы к диагональному и трапецеидальному видам целесообразно представить совокупность всех целых чисел в виде однородного бинарного массива. Целые числа могут быть как малыми, так и очень большими. Следовательно, их бинарные представления являются либо короткими, либо очень длинными. Строение бинарного массива должно учитывать эту особенность. Обработку этого массива, который может быть весьма большим, рационально распределить между параллельно работающими процессорами.

Второй способ распараллеливания состоит в разбиении матрицы A на клетки, в приведении её отдельных клеток к требуемому виду и в выполнении преобразований, дающих диагональную или трапецеидальную матрицу.

УПРАЖНЕНИЯ

7.1. Указать три положительных целых числа, которые являются взаимно простыми, а любая пара из этих чисел не является таковой.

7.2. Доказать свойства наибольшего общего делителя.

А. Если m — любое положительное число, то $\gcd(ma, mb) = m\gcd(a, b)$.

Б. Если δ — любой общий делитель чисел a и b , то $\gcd(a/\delta, b/\delta) = (1/\delta)\gcd(a, b)$. В частности, $\gcd(a/\gcd(a, b), b/\gcd(a, b)) = 1$, т. е. частные от деления двух чисел на их наибольший общий делитель — взаимно простые числа.

В. Если $\gcd(a, b) = 1$, то $\gcd(ac, b) = \gcd(c, b)$.

Г. Если $\gcd(a, b) = 1$ и ac делится на b, то c делится на b.

Д. Если каждое из чисел $a_1, a_2, \ldots, a_m$ взаимно простое с каждым из чисел $b_1, b_2, \ldots, b_n$, то произведение $a_1a_2 \ldots a_m$ взаимно простое с произведением $b_1b_2 \ldots b_n$.

7.3. А. Пусть a и b — целые числа, не равные одновременно нулю, а $d = ax_0 + by_0$ — наименьшее положительное число вида $ax + by$(x и y — целые). Доказать, что $d = \gcd(a, b)$.

Б. Использовав результат, полученный в п. А этого упражнения, дать другое доказательство утверждений: совокупность общих делителей чисел a и b совпадает с совокупностью делителей их наибольшего общего делителя; имеют место предложения А и Б упражнения 7.2.

В. Обобщить выводы пунктов А и Б этого упражнения, рассмотрев числа вида $a_1x_1 + a_2x_2 + \ldots + a_nx_n$.

7.4. Применив алгоритмы Евклида с положительным остатком и с наименьшим остатком, найти $\gcd(525, 231)$, $\gcd(6188, 4709)$, $\gcd(81719, 52003, 33649, 30107)$. Сравнить фактическое число итераций алгоритма Евклида с оценкой, даваемой соответствующей теоремой.

7.5. Дать детальное доказательство теоремы 7.4, использовав рассуждения, аналогичные рассуждениям теоремы 7.1.

7.6. Указать пару положительных целых чисел a и $b, a > b > 2$, для которой число итераций алгоритма Евклида с наименьшим остатком равно $\lfloor \log_2 b \rfloor + 1$.

7.7. Установить справедливость следующей теоремы (дать доказательство, аналогичное доказательствам теорем 7.1 и 7.2).

Теорема. Если для нахождения наибольшего общего делителя $\gcd(a_1, a_2, \ldots, a_n)$ положительных целых чисел $a_1, a_2, \ldots, a_n$ используется алгоритм Евклида с наименьшим остатком по схеме
$d_2 = \gcd(a_1, a_2)$, $d_i = \gcd(a_i, d_{i-1})$, $i = \overline{3, n}$, то число его итераций не превышает $n - 2 + (\lfloor \log_2 a \rfloor + 1)$, где $a_1 = \min\{a_1, a_2, \ldots, a_n\}$ $(n \geq 2)$. Эта оценка достижима.

7.8. Для величин $u_i, i = \overline{1, +\infty}$,встречающихся в доказательствах теорем 7.1, 7.2, 7.4 и теоремы из упражнения 7.7, написать уравнение в конечных разностях, найти решение этого уравнения и доказать неравенство вида $u_i \geq ag^{i-k}$.

7.9. Проведя вычислительный эксперимент над алгоритмами Евклида с положительным остатком и с наименьшим остатком, найти коэффициенты α_1, α_2, в выражениях

$$n - 2 + \alpha_1(\lfloor \log_{10} |a_1| \rfloor + 1), \quad n - 2 + \alpha_2(\lfloor \log_2 |a_1| \rfloor + 1),$$

которые дают среднее число их итераций. Ясно, что $0 < \alpha_1 \leq 5, 0 < \alpha_2 \leq 1$.

7.10. Обосновать вычислительную схему без обратного хода для нахождения наибольшего общего делителя n положительных целых чисел и множителей в его представлении (§ 7.2).

7.11. Алгоритм 1 из § 7.2 найти наибольший общий делитель и множители в его представлении для следующих чисел:

а) $139, -86$;

б) $707, 231, 525$;

в) $424, 444, 932, 22347$;

г) $93322, 81719, 52003, 33649, 30107$.

7.12. Показать, что всякая унимодулярная целочисленная матрица представима в виде произведения конечного числа только элементарных столбцовых преобразований (только элементарных строковых преобразований).

7.13. Показать, что наибольший общий делитель d_k всех миноров k-го порядка целочисленной матрицы A не меняется при выполнении элементарных преобразований.

7.14. Алгоритмами 1 и 2 из § 7.4 привести матрицу

$$\begin{pmatrix} 7 & -1 & -1 & 0 \\ 1 & 3 & 2 & 1 \\ 10 & 20 & -15 & 0 \\ 13 & 21 & 0 & -37 \end{pmatrix}$$

к трапецеидальному виду.

7.15. Доказать, что для всякой невырожденной целочисленной матрицы A существует единственная эрмитова матрица.

7.16. Алгоритмами 1 и 2 из § 7.5 привести матрицу

$$\begin{pmatrix} 7 & -1 & -1 & 0 \\ 1 & 3 & 2 & 1 \\ 10 & 20 & -15 & 0 \\ 13 & 21 & 0 & -37 \end{pmatrix}$$

к нормальному виду.

7.17. Построить и обосновать алгоритм приведения целочисленной матрицы к нормальному виду, использовав следующую схему:

а) при помощи элементарных преобразований приводим матрицу A к виду $\begin{pmatrix} a^1_{1,1} & 0 \\ 0 & A^1 \end{pmatrix}$, где $a^1_{1,1}$ делят все элементы матрицы A^1;

б) для матрицы A^1 выполняем шаг а) и т. д.

7.18. Доказать индукцией по порядку матрицы, что для всякой квадратной целочисленной матрицы A порядка n существует такие две унимодулярные целочисленные матрицы E и F , что диагональная матрица $D = EAF$ является нормальной.

7.19. Доказать, что для всякой целочисленной матрицы A существует единственная нормальная матрица (воспользуемся результатом из упражнение 7.13).

7.20. Алгоритмами 1 и 2 из § 7.6 найти общее целочисленной решение системы линейных уравнений $Ax = b$, где

$$A = \begin{pmatrix} 1 & 1 & 1 & -4 & 1 \\ 2 & 0 & 2 & -1 & -1 \\ 1 & -1 & 1 & 3 & -2 \end{pmatrix}, \quad b = \begin{pmatrix} 0 \\ 2 \\ 2 \end{pmatrix}.$$

Решение некоторых упражнений

7.1. Например, числа 6,10,15.

7.2. А. Для чисел a и b алгоритмом Евклида с положительным остатком найти последовательность равенств. Умножить каждое равенство на m.

Б. Воспользоваться утверждением А этого упражнения.

В. Число $\gcd(ac, b)$ делит ac и b , следовательно, оно делит и $\gcd(ac, bc) = c\gcd(a, b) = c$. Но число $\gcd(ac, b)$ делит и b. Поэтому оно делит и $\gcd(c, b)$. Обратно, число $\gcd(c, b)$ делит ac и b. Поэтому оно

делит и $\gcd(ac, b)$. Таким образом, числа $\gcd(ac, b)$ и $\gcd(ac, b)$ взаимно делят друг друга и, следовательно, равны между собой.

Г. Воспользоваться утверждением В этого упражнения.

Д. Неоднократно воспользоваться утверждением В этого упражнения.

7.3.

А. Остаток от деления $ax + by$ на d имеет вид $ax' + by'$, является неотрицательным и меньше d. Поэтому он непременно равен нулю. Следовательно, d — делитель всех чисел вида $ax+by$ и, в частности, общий делитель чисел $a1 + b0 = a$ и $a0 + b1 = b$. С другой стороны, выражение для d показывает, что всякий общий делитель чисел a и b делит d. Отсюда $d = \gcd(a, b)$.

Б. Всякий общий делитель чисел a и b делит d, которое даёт $\gcd(a, b)$. Наименьшее положительное число вида $(ma)x+(mb)y$ есть $(ma)x_0+(mb)y_0$. Наименьшее положительное число вида $(a/\delta)x+(b/\delta)y$ есть $(a/\delta)x_0 + (b/\delta)y_0$.

7.4. $\gcd(525, 231) = 21$, $\gcd(6188, 4709) = 17$, $\gcd(81719, 52003, 33649, 30107) = 23$.

7.6. Например, 8 и 3.

7.7. Оценка достигается, например, для чисел 3,8, а так же для чисел 4, 10, 13.

7.11. Искомые величины:

а) $\begin{matrix} 139 & -86, d_2 = 1 \\ 13 & 21 \end{matrix}$;

б) $\begin{matrix} 707 & 231 & 525, d_3 = 7 \\ -1 & -6 & 4 \end{matrix}$;

в) $\begin{matrix} 424 & 444 & 932 & 22347, d_4 = 1 \\ 37 & 15 & 0 & -1 \end{matrix}$;

г) $\begin{matrix} 93322 & 81719 & 52003 & 33649 & 30107, d_5 = 1 \\ -2 & -3 & 5 & 6 & -1 \end{matrix}$.

7.14. Искомыми являются матрицы

$$\begin{matrix} T \\ \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 5 & 0 \\ -3726 & -37 & -2168 & 5078 \end{pmatrix} \end{matrix} \quad \begin{matrix} P \\ \begin{matrix} 1 \\ 2 \\ 3 \\ 4 \end{matrix} \end{matrix} \quad \begin{matrix} A \\ \begin{pmatrix} 7 & -1 & -1 & 0 \\ 1 & 3 & 2 & 1 \\ 10 & 20 & -15 & 0 \\ 136 & 21 & 0 & -37 \end{pmatrix} \end{matrix}$$

$$\begin{matrix} K \\ \begin{pmatrix} -5 & 0 & -3 & 7 \\ -14 & 0 & -8 & 19 \\ -22 & 0 & -13 & 30 \\ 91 & 1 & 53 & -124 \end{pmatrix} \end{matrix}.$$

Оба алгоритма дают один и тот же результат.

7.16. Искомыми являются матрицы

$$D = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 25390 \end{pmatrix}, \quad E = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 3 & 1 & 0 & 0 \\ 73779 & 30387 & -876 & -2 \\ -184490 & -75985 & 2168 & 5 \end{pmatrix},$$

$$A = \begin{pmatrix} 7 & -1 & -1 & 0 \\ 1 & 3 & 2 & 1 \\ 10 & 20 & -15 & 0 \\ 13 & 21 & 0 & -37 \end{pmatrix}, \quad F = \begin{pmatrix} 0 & 0 & -3 & -30461 \\ -1 & 1 & -8 & -81229 \\ 0 & -1 & -13 & -131998 \\ 0 & 0 & 53 & 538144 \end{pmatrix}.$$

Оба алгоритма дают один и тот же результат.

7.20. Алгоритм 1 даёт матрицы

$$\begin{array}{c|ccc|c} p & & K_2 & & f \\ \hline 1 & -1 & -1 & 2 & 2 \\ 2 & 0 & -7 & 9 & 6 \\ 3 & 1 & 0 & 0 & 0 \\ & 0 & -2 & 3 & 2 \\ & 0 & 0 & 1 & 0 \end{array}.$$

Алгоритм 2 даёт матрицы

$$\begin{array}{c|ccc|c} p & & K_2 & & f \\ \hline 1 & -1 & 4 & -1 & 0 \\ 2 & -3 & 14 & -6 & -4 \\ 3 & 2 & -7 & 3 & 2 \\ & 0 & 1 & 0 & 0 \\ & 2 & -7 & 4 & 2 \end{array}.$$

Глава 8

КРАТКИЕ СВЕДЕНИЯ О ВСПОМОГАТЕЛЬНЫХ АЛГОЛ-ПРОЦЕДУРАХ

Настоящая глава содержит описание десяти процедур для решения семи математических задач. Эти процедуры представлены в форме, удобной для распараллеливания. В свою очередь, они могут служить вспомогательными средствами при создании параллельных программ, реализующих алгоритмы целочисленной оптимизации.

Каждая процедура предназначена как для чтения человеком, так и для счёта машиной. Все процедуры записаны на языке Алгол-60.

8.1 ОБЩИЕ ЗАМЕЧАНИЯ

В работе [56] представлены тексты процедур, которые являются точными копиями машинных выдач. Для каждой процедуры, кроме ее текста, формулируется математическая задача, которую она решает, разъясняется смысл формальных параметров этой процедуры и приводятся тестовые задачи. Разумеется, чтению процедуры должно предшествовать ознакомление с алгоритмом, который она реализует. Детальное изложение алгоритмов, реализованных в виде процедур, можно найти, например, в работах [54, 56] (см. также гл. 6 и 7).

Счёт по процедурам производится после их трансляции на трансляторах АЛГОЛ–БЭСМ-6 и АЛЬФА-6. В каждой из процедур предусмотрен контроль, выполнение которого гарантирует её правильную работу. Процедуры прошли тщательную проверку путём решения разнообразных тестовых задач. При счёте по процедурам в их окончательной редакции отклонения от правильной работы не были обнаружены.

При приведении конкретных целочисленных матриц к специальному виду были получены сигналы о появлении больших целых чисел (скажем, больше чем 10^{10}), хотя элементы исходной матрицы были малы. Выполнение арифметических операций над большими целыми числами без потери младших разрядов требует создания специальных процедур в рамках языка Алгол-60 или других средств, с помощью которых экономно осуществляется поразрядная обработка бинарных данных.

При отыскании всех вершин конкретных многогранников, где использовалась процедура для нахождения всех крайних лучей заострённого конуса, а также при решении конкретных линейных задач оптимизации даже сравнительно небольшого размера приходилось учитывать влияние ошибок округления.

В этой главе ограничимся лишь основными сведениями о процедурах, описанных в [56]. Для каждой процедуры формулируется математическая задача, которую она решает, а также перечисляются ее входные и выходные параметры.

8.2 ПРОЦЕДУРА НАХОЖДЕНИЯ МНОЖИТЕЛЕЙ В ПРЕДСТАВЛЕНИИ НАИБОЛЬШЕГО ОБЩЕГО ДЕЛИТЕЛЯ

Для заданных целых чисел $a_1, a_2, \ldots, a_n$ процедура eucl 1 находит их наибольший общий делитель d_n, множители $x_1, x_2, \ldots, x_n$, в его представлении $d_n = \sum_{j=1}^{n} x_j a_j$ и дополнительную информацию из семи чисел, состоящую из величин $y_2, z_2, u_2, v_2, k, l, \overline{y_3}$, для которых выполняются соотношения

$$a_k y_2 + a_l z_2 = d_2,$$

$$a_k u_2 + a_l v_2 = 0,$$

$$\overline{y_3} d_2 + \sum_{\substack{j=1 \\ j \neq k,l}}^{n} x_j a_j = d_n.$$

Работа процедуры eucl 1 начинается с поиска наибольшего общего делителя d_2 чисел a_k и a_l, где a_k — наименьшее по абсолютной величине ненулевое число, a_l — наименьшее по абсолютной величине ненулевое число среди $a_j, j \neq k$, не делящееся на a_k.

Параметрами процедуры eucl 1 являются величины:

1) количество заданных целых чисел n;

2) заданные целые числа $a_1, a_2, \ldots, a_n$;

3) наибольший общий делитель d_n чисел $a_1, a_2, \ldots, a_n$;

4) множители $x_1, x_2, \ldots, x_n$ в представлении наибольшего общего делителя d_n и величины $y_2, z_2, u_2, v_2, k, l, \overline{y_3}$.

8.3 ПРОЦЕДУРЫ ПРИВЕДЕНИЯ ЦЕЛОЧИСЛЕННОЙ МАТРИЦЫ К ТРАПЕЦЕИДАЛЬНОМУ ВИДУ

Для заданной целочисленной матрицы A процедуры trap 1 и trap 2 находят такие две унимодулярные целочисленные матрицы P и K, что матрица $T = PAK$ имеет трапецеидальный вид, показанный на рис. 7.1 (The Trapezium — трапеция). Здесь P — перестановочная матрица, представляющая произведение выполненных перестановок строк.

В процедуре trap 1 реализован традиционный алгоритм приведения целочисленной матрицы к трапецеидальному виду в кольце целых чисел. В этом алгоритме используются элементарные столбцовые преобразования и, возможно, перестановки строк. В процедуре trap 2 в отличие от trap 1 применяется процедура eucl 1. Это позволяет уменьшить количество выполняемых арифметических операций. В обеих процедурах не выполняются арифметические операции, которые не изменяют отдельных частей матрицы.

Параметрами процедур trap 1 и trap 2 являются величины:

1) m — число строк заданной матрицы A;

2) n — число столбцов заданной матрицы A;

3) A — заданная целочисленная $m \times n$-матрица;

4) T — целочисленная $m \times n$-матрица, имеющая трапецеидальный вид;

5) K — целочисленная $m \times n$-матрица, задающая произведение выполненных элементарных столбцовых преобразований;

6) P — перестановочная $m \times n$-матрица, задающая произведение выполненных перестановок строк;

7) r — ранг заданной матрицы A.

8.4 ПРОЦЕДУРЫ ПРИВЕДЕНИЯ ЦЕЛОЧИСЛЕННОЙ МАТРИЦЫ К НОРМАЛЬНОМУ ВИДУ

Для заданной целочисленной матрицы A процедуры diag 1 и diag 2 находят такие две унимодулярные целочисленные матрицы E и F, что диагональная матрица $D = EAF$ имеет нормальный вид:

$d_{i,i} > 0, i = \overline{1,r}; d_{i,i}$ является делителем $d_{i+1,i+1}, i = \overline{1,r-1}$, где r — ранг матрицы A (The Diagonal — диагональ).

В процедуре diag 1 реализован традиционный алгоритм приведения целочисленной матрицы к диагональному виду в кольце целых чисел. В этом алгоритме пользуются элементарные столбцовые и строковые преобразования. В процедуре diag 2 в отличие от diag 1 применяется процедура eucl 1. Это позволяет уменьшит количество выполняемых арифметических операций. В обеих процедурах не выполняются арифметические операции, которые не изменяют отдельных частей матрицы.

Параметрами процедур diag 1 и diag 2 являются величины:

1) m — число строк заданной матрицы A;

2) n — число столбцов заданной матрицы A;

3) A — заданная целочисленная $m \times n$-матрица;

4) D — нормальная матрица для заданной матрицы A;

5) E — целочисленная $m \times n$-матрица, задающая произведение выполненных элементарных строковых преобразований;

6) F — целочисленная $m \times n$-матрица, задающая произведение выполненных элементарных столбцовых преобразований;

7) r — ранг заданной матрицы A.

8.5 ПРОЦЕДУРЫ НАХОЖДЕНИЯ ОБЩЕГО ЦЕЛОЧИСЛЕННОГО РЕШЕНИЯ СИСТЕМЫ ЛИНЕЙНЫХ УРАВНЕНИЙ

Для заданной системы линейных уравнений $Ax = b$ с целочисленными данными A, b процедуры diop 1 и diop 2 либо находят общее целочисленное решение $x = f + K_2 t$, либо устанавливают, что эта система не имеет целочисленных решений (Diophantos — Диофант).

В процедуре diop 1 реализован традиционный алгоритм, основанный на приведении целочисленной матрицы A к трапецеидальному виду. В этом алгоритме используются элементарные столбцовые преобразования и, возможно, перестановки строк. В процедуре diop 2 в отличие от diop 1 применяется процедура eucl I. Это позволяет уменьшить количество выполняемых арифметических операций. В обеих процедурах не выполняются арифметические операции, которые не изменяют отдельных частей матрицы.

Параметрами процедур diop 1 и diop 2 являются величины:

1) m — число строк заданной матрицы A;

2) n — число столбцов заданной матрицы A;

3) A — целочисленная $m \times n$-матрица коэффициентов заданной системы уравнений $Ax = b$;

4) b — целочисленный m-вектор свободных членов заданной системы уравнений $Ax = b$;

5) r — ранг заданной матрицы A;

6) $p = (i_1, i_2, \ldots, i_m)$ — перестановка чисел $1, 2, \ldots, m$, задающая произведение выполненных перестановок строк;

7) C — расширенная $(m + n) \times (n + 1)$-матрица (вид матрицы C перед работой процедуры и после её выполнения показан на рис. 7.2).

8.6 ПРОЦЕДУРА НАХОЖДЕНИЯ ВСЕХ КРАЙНИХ ЛУЧЕЙ ЗАОСТРЕННОГО КОНУСА

Для заданного заострённого конуса $C = \{x \mid x \geq 0,\ Ax \geq 0\}$ процедура exray 1 находит все его крайние направления (The Extreme Ray — крайний луч). Здесь A — рациональная $m \times n$-матрица. Под крайним направлением (вектором) конуса C понимаем направляющий вектор его крайнего луча. Найденные направления нормируются при помощи евклидовой нормы $|x| = \sqrt{\sum_{j=1}^{n} x_j^2}$. Если система

неравенств $x \geq 0$, $Ax \geq 0$ имеет только нулевое решение, то число (крайних лучей) считается равным нулю. Это также устанавливается процедурой.

Параметрами процедуры exray 1 являются величины:

1) m — число строк заданной матрицы A;

2) n — число столбцов заданной матрицы A;

3) A — заданная рациональная $m \times n$-матрица;

4) e — достаточно малое положительное число, точность счёта;

5) $p1$ — размер массивов, отводимых для хранения ненулевых компонент крайних векторов конуса C и номеров этих компонент;

6) $p2$ — размер массива, отводимого для хранения бинарных строк;

7) r — число крайних направлений конуса C;

8) массив ind для хранения номеров ненулевых компонент крайних векторов конуса C;

9) массив x для хранения значений ненулевых компонент крайних векторов конуса C;

10) массив $c1$ для хранения целочисленных счётчиков, характеризующих счёт;

11) массив $c2$ для хранения рациональных счётчиков, характеризующих счёт.

8.7 ПРОЦЕДУРА НАХОЖДЕНИЯ ВСЕХ ВЕРШИН МНОГОГРАННИКА

Для заданного многогранника $P = \{x \mid Ax \rho b,\ x \geq 0\}$ процедура exray2 находит все его вершины и направления всех его крайних лучей. После нахождения всех вершин происходит их упорядочение по возрастанию значений заданной линейной функции cx. В формулировке задачи A — рациональная $m \times n$-матрица, ρ — вектор отношений $\leq, =, \geq$.

Параметрами процедуры exray 2 являются величины:

$m, n, n1, l, k, a, b, c, i0$ — имеют тот же смысл, что и в процедуре simp 6 (см. § 8.8);

$e, p1, p2, r, \mathrm{ind}, x, c1, c2$ — имеют то же значение, что и в процедуре exray 1 (см. § 8.6);

v — число вершин многогранника P;

cx — значение функции cx в вершине x многогранника P.

8.8 ПРОЦЕДУРА РЕШЕНИЯ ЛИНЕЙНОЙ ЗАДАЧИ ОПТИМИЗАЦИИ

Рассмотрим прямую и двойственную задачи

$$\begin{array}{llll} \min \sum\limits_{j=1}^{n} c_j x_j, & & & \max \sum\limits_{i=1}^{m} y_i b_i, \\ \sum\limits_{j=1}^{n} a_{i,j} x_j \leq b_i, & i \in I_1, & y_i \leq 0, & \\ \sum\limits_{j=1}^{n} a_{i,j} x_j = b_i, & i \in I_2, & y_i <=> 0 & . \\ \sum\limits_{j=1}^{n} a_{i,j} x_j \geq b_i, & i \in I_3, & y_i \geq 0, & \\ x_j \geq 0, & j \in J, & \sum\limits_{i=1}^{m} y_i a_{i,j} \leq c_j. & \end{array}$$

Множества индексов I_1, I_2, I_3, J относятся к прямой и двойственной задачам (некоторые из этих множеств могут быть пустыми). Обе задачи используют одни и те же исходные данные. Поэтому, говоря об исходных данных, будем иметь в виду только прямую задачу, т. е. задачу минимизации, для краткости называя ее задачей.

В формулировке задач используются обозначения:

m — число основных ограничений задачи;

n — число неотрицательных переменных задачи;

$a_{i,j}, b_i, c_j$ — заданные рациональные числа ($i = \overline{l,m},\ j = \overline{1,n}$);
$A = (a_{i,j})$ — $m \times n$-матрица ограничений;
a_i — i-я строка матрицы A, $i = \overline{1,m}$;
a_j — j-й столбец матрицы A, $j = \overline{1,n}$;
$I = \{1, 2, \ldots, m\}$, $I_1 = \{i \mid a_i x \le b_i,\ i \in I\}$, $I_2 = \{i \mid a_i x = b_i,\ i \in I\}$, $I_3 = \{i \mid a_i x \ge b_i,\ i \in I\}$;
$I_k \bigcap I_l = \varnothing$, $k \ne l$, $I_1 \bigcup I_2 \bigcup I_3 = I$;
$J = 1, 2, \ldots, n$;
x_j, $j = \overline{1,n}$, — переменные прямой задачи;
y_i, $i = \overline{1,m}$, — переменные двойственной задачи.

Процедура simp 6 предназначена для решения линейной задачи оптимизации. Она реализует модифицированный симплекс-метод. В результате её выполнения находятся оптимальные решения x^*, y^* прямой и двойственной задач, а также оптимальное значение $cx^* = y^*b$ целевой функции обеих задач. При помощи этой процедуры можно найти и допустимые решения рассматриваемых задач. Для нахождения допустимого решения прямой (двойственной) задачи достаточно положить $c_j = 0$, $j = \overline{1,n}$ ($b_i = 0,\ i = \overline{1,m}$).

Параметрами процедуры simp 6 являются величины:

1) m — число строк матрицы A;

2) n — число столбцов матрицы A;

3) $n1$ — величина, вычисляемая по формуле $n1 = a' + n + n'$, где a' — число ненулевых элементов матрицы A (если их нет, то $a' = 0$), n — число столбцов матрицы A, n' — число нулевых столбцов матрицы A (если их нет, то $n' = 0; n' \le n$);

4) l — вектор отношений $\le, =, \ge$;

5) массив k для хранения строковых номеров элементов $a_{i,j}$ матрицы A;

6) массив a для хранения значений элементов $a_{i,j}$ матрицы A;

7) массив b для хранения свободных членов b_i столбца b;

8) массив c для хранения коэффициентов c_j минимизируемой функции cx;

9) $i0$ — признак, характеризующий существование оптимального решения или его отсутствие;

10) $a0$ — минимальное значение целевой функции cx задачи;

11) массив $c1$ для хранения целочисленных счётчиков, характеризующих счёт;

12) массив $c2$ для хранения рациональных счётчиков, характеризующих счёт.

В табл. 8.1 приведены результаты счёта (процедура simp 6, система АЛЬФА-6). В ней используются обозначения: $n0$ — номер задачи, m — число основных ограничений, n — число неотрицательных переменных, $n1$ — число ненулевых элементов $m \times n$-матрицы A основных ограничений, k' — число выполненных симплекс-итераций, t — время решения задачи в секундах.

Таблица 8.1

$n0$	m	n	$n1$	k'	t	$n0$	m	n	$n1$	k'	t
2401	32	88	408	121	9	2515	51	92	543	89	13
2402	50	225	1075	215	24	2516	67	187	543	345	62
2403	44	88	552	132	12	2701	119	81	543	170	69
2404	70	225	1475	417	63	2702	119	81	543	220	87
2405	87	88	436	285	60	2703	119	81	543	224	88
2406	47	45	280	66	8	2704	119	81	543	217	87
2407	51	46	310	77	10	2705	119	81	543	20	12
2411	7	12	46	9	1	2706	119	81	543	276	105
2412	13	26	176	12	2	2707	119	81	543	251	98
2413	33	51	356	160	11	2711	51	48	543	54	8
2414	28	57	263	35	4						

ПРИЛОЖЕНИЕ. ОСНОВНЫЕ ОБОЗНАЧЕНИЯ И СОКРАЩЕНИЯ

Числа

0 — число нуль;

$-\infty, +\infty$ — несобственный числа;

$\operatorname{sign}(a) = \begin{cases} -1, & a < 0, \\ 0, & a = 0, \\ 1, & a > 0; \end{cases}$ — знак действительного числа a;

$|a| = \begin{cases} -a, & a < 0, \\ a, & a \geq 0; \end{cases}$ — абсолютная величина действительного числа a;

$\lfloor a \rfloor$ — наибольшее целое число, не большее действительного числа a;

$\lceil a \rceil$ — наименьшее целое число, не меньшее действительного числа a;

$\lfloor \lfloor a \rfloor \rfloor = \lfloor a + 1/2 \rfloor$ — целое число, ближайшее к действительному числу a;

$a (\operatorname{mod} \delta)$ — неотрицательный остаток от деления целого числа a на положительное целое число δ;

$a \mid b$ — целое число a является делителем целого числа b;

$a \nmid b$ — целое число a не является делителем целого числа b;

$a/b, \frac{a}{b}$ — частное от деления действительного числа a на действительное число b;

$\gcd(a_1, a_2, \ldots, a_n)$ — наибольший общий делитель целых чисел $a_1, a_2, \ldots, a_n$ $(n \geq 1)$;

$\sum\limits_{i=1}^{n} a_i$ — сумма действительных чисел $a_1, a_2, \ldots, a_n$;

$\min\{a_1, a_2, \ldots, a_n\}$, $\min\limits_{i=1}^{n} a_i$ — число, наименьшее среди действительных чисел $a_1, a_2, \ldots, a_n$;

$\max\{a_1, a_2, \ldots, a_n\}$, $\max\limits_{i=1}^{n} a_i$ — число, наибольшее среди действительных чисел $a_1, a_2, \ldots, a_n$;

$a := b$ — a присвоить b, т. е. левой части соотношения присваивается значение правой части этого соотношения.

Векторы

0 — вектор, все компоненты которого равны нулю;

$a \equiv 0$ — все компоненты вектора $a = (a_1, a_2, \ldots, a_n)$ целочисленные;

$|a|, \|a\|$ — норма вектора a;

$ab, (a, b)$ — скалярное произведение векторов a и b;

a^T — вектор, транспонированный к вектору a;

$a \bmod (\delta) = (a_1 (\operatorname{mod} \delta_1), a_2 (\operatorname{mod} \delta_2), \ldots, a_n (\operatorname{mod} \delta_n))$ — вектор неотрицательных остатков для целочисленного вектора $a = (a_1, a_2, \ldots, a_n)$ м для целочисленного вектора $\delta = (\delta_1, \delta_2, \ldots, \delta_n)$;

$a + b$ — сумма векторов a и b.

Матрицы

0 — матрица, все элементы которой равны нулю;

I_n — единичная матрица порядка n;

$\det A, |A|$ — определитель матрицы A;

$r(A)$ — ранг матрицы A;

A^T — матрица, транспонированная к матрице A;

A^{-1} — матрица, обратная к матрице A;

$A + B$ — сумма матриц A и B;

$AB, A \times B$ — произведение матриц A и B;

$A \otimes B$ — тензорное произведение матриц A и B.

Для векторов и матриц отношение равенства ($=$), неравенства ($\leq, \geq, <, >, \ll, \gg, <=>$), сравнения ($\equiv$), присваивания ($:=$) означают, что они выполняются покомпонентно. Числа, векторы, компоненты вектора, элементы матрицы обозначаются малыми буквами латинского и греческого алфавитов, а матрицы — большими буквами этих алфавитов. Для записи компонент вектора, обозначенного малой буквой, используется та же самая буква, снабжённая одним или несколькими индексами.

Символ p, например, в словах p-алгоритм, p-программа означает число параллельно работающих процессоров ($p \geq 2$).

Множества

$\varnothing$ — пустое множество, т. е. множество, не содержащее ни одного элемента;

$\{a_1.a_2, \dots, a_n\}$ — множество, состоящее из элементов $a_1, a_2, \dots, a_n$;

$\{a \mid \dots\}$ — множество элементов a, удовлетворяющих условию $\dots$;

B — множество, состоящее из чисел 0 и 1, $|B| = 2$;

B_n — множество, состоящее из всех n-векторов с бинарными компонентами, (значение бинарной компоненты равно 0 или 1), $|B| = 2^n (\gg 1)$;

R — множество всех действительных чисел;

R^+ — множество всех неотрицательных действительных чисел;

R_n — множество всех n-векторов с действительными компонентами;

R_n^+ — множество всех n-векторов с неотрицательными действительными компонентами;

Z — множество всех целых чисел;

Z^+ — множество всех неотрицательных целых чисел;

Z_n — множество всех n-векторов с целочисленными компонентами;

Z_n^+ — множество всех n-векторов с неотрицательными целочисленными компонентами;

$\mathrm{conv}A$ — выпуклая оболочка множества $A, A \subseteq R_n$;

$|A|$ — число элементов множества A;

$\inf X$ — наибольшая нижняя граница для числового множества X;

$\sup X$ — наименьшая верхняя граница для числового множества X;

$A \subset B$ — каждый элемент множества A является элементом множества B;

$A \bigcup B$ — объединение множеств A и B;

$A \bigcap B$ — пересечение множеств A и B;

$A \backslash B$ — разность двух множеств A и B;

$A \backslash a$ — множество, полученное из A исключением его элемента a;

$$i = \overline{k,l} - \{i_1, i_2, \dots\} = \begin{cases} k, k+1, \dots, l & \text{для } k < l, \\ k & \text{для } k = l, \\ k, k-1, \dots, l & \text{для } k > l, \end{cases} \quad \text{где } k \text{ и } l \text{ — целые числа.}$$

Математическая логика

$A \bigwedge B$ — конъюнкция двух высказываний A и B (A и B);

$A \bigvee B$ — дизъюнкция двух высказываний A и B (A или B);

$A \to B$ — импликация двух высказываний A и B (из A следует B);

$A \leftarrow\rightarrow B$ — эквивалентность двух высказываний A и B (из A следует B, из B следует A);

$\neg A$ — отрицание высказывания A (не A);

$\forall i$ — для всех i (квантор общности);

$\exists i$ — существует i (квантор существования);

Высказывание, выступающие в качестве условия и записываемое, как правило, после запятой или вертикальной линии, всегда является истинным.

Названия задач

Перед словом "задача" (соответственно в английском языке перед буквой P (Program)) может использоваться сокращения, состоящее либо из одной, либо из двух, либо из трёх букв, указанных ниже.

Б — бинарная, B — binary

Г — групповая, G — group

Д — дизъюнктивная, D — disjunctive

Е — дискретная, E — discrete

Л — линейная, L — linear

С — смешанная, M — mixed

Н — нелинейная, N — nonlinear

Ц — целочисленная, I — integer

При помощи таблицы перечислим часто встречающиеся задачи оптимизации. Полное название задачи оптимизации перед словом "задача" имеет два или три определения. Определение, стоящее непосредственно перед словом "задача" (З), характеризует вид ограничений (например, групповая, дизъюнктивная, линейная), а остальные определения — вид переменных (бинарная (Б), дискретная (Е), целочисленная (Ц), смешанная бинарная (СБ), смешанная дискретная (СЕ), смешанная целочисленная (СЦ)).

Символ $\varnothing$ означает, что перед словом "задача" отсутствует соответствующее прилагательное. Перечёркнутые клетки в приведённой таблице указывают на отсутствие соответствующих задач.

Таблица названий задач оптимизации

Ограничения \ Переменные	$\varnothing$	Б	Е	Ц	СБ	СЕ	СЦ
$\varnothing$							
Блочная							
Дизъюнктивная							
Групповая			×			×	
Вогнутая							
Выпуклая							
Квадратичная							
Линейная							З
Нелинейная							
Полиномиальная							
Сепарабельная							

Клетке З, например, соответствует смешанная целочисленная линейная задача.

Приведем еще несколько сокращений: П-задача о покрытии; Р-задача о разбиении; У-задача об упаковке.

Формулировки основных задач оптимизации

Бинарная линейная задача и её линейная релаксация

$$\min\{cx \mid Ax \geq b,\ x_j = 0,1,\ j = \overline{1,r}\},$$

$$\min\{cx \mid Ax \geq b,\ 0 \leq x_j \leq 1,\ j = \overline{1,r}\}.$$

Дискретная линейная задача и её линейная релаксация

$$\min\{cx \mid Ax \geq b,\ x_j \in D_j,\ j = \overline{1,r}\},$$

$$\min\{cx \mid Ax \geq b\}.$$

Целочисленная линейная задача и её линейная релаксация

$$\min\{cx \mid Ax \geq b,\ x \geq 0,\ x_j \equiv 0,\ j = \overline{1,r}\},$$

$$\min\{cx \mid Ax \geq b,\ x \geq 0\}.$$

Смешанная бинарная линейная задача и её линейная релаксация

$$\min\{cx + dy \mid Ax + By \geq b,\ x_j = 0,1,\ j = \overline{1,r},\ y \geq 0\},$$

$$\min\{cx + dy \mid Ax + By \geq b,\ 0 \leq x_j \leq 1,\ j = \overline{1,r},\ y \geq 0\}.$$

Смешанная дискретная линейная задача и её линейная релаксация

$$\min\{cx + dy \mid Ax + By \geq b,\ x \in D_j,\ j = \overline{1,r},\ y \geq 0\},$$

$$\min\{cx + dy \mid Ax + By \geq b,\ y \geq 0\}.$$

Смешанная целочисленная линейная задача и её линейная релаксация

$$\min\{cx + dy \mid Ax + By \geq b,\ x \geq 0, x_j \equiv 0,\ j = \overline{1,r},\ y \geq 0\},$$

$$\min\{cx + dy \mid Ax + By \geq b,\ x \geq 0,\ y \geq 0\}.$$

Смешанная целочисленная линейная задача и её лагранжева релаксация

$$\min\{cx \mid A^1x \geq b^1,\ A^2x \geq b^2,\ x \geq 0, x_j \equiv 0,\ j \in N_1 \subseteq N\},$$

$$l(u) = \min\{(c - uA^2)x + ub^2 \mid A^1x \geq b^1,\ x \geq 0,\ x_j \equiv 0,\ j \in N_1 \subseteq N\}.$$

Целочисленная линейная задача и её коническая релаксация

$$\max\{c_Bx_B + c_Nx_N \mid Bx_B + Nx_N = b,\ x_B \geq 0,\ x_B \equiv 0,\ x_N \geq 0,\ x_N \equiv 0\},$$

$$\max\{c_Bx_B + c_Nx_N \mid x_B \equiv 0,\ x_N \geq 0,\ x_N \equiv 0\}.$$

Целочисленная групповая задача

$$\min\Big\{\sum_{j=1}^{r} c'_jx_j \mid \sum_{j=1}^{r} h_jx_j = h_0,\ x_j \in Z^+,\ j = \overline{1,r}\Big\},$$

групповая релаксация целочисленной линейной задачи

$$\min\{cx \mid Ax = b,\ x \geq 0,\ x_j \equiv 0\}.$$

Смешанная целочисленная групповая задача

$$\min\Big\{\sum_{j=1}^{r} c'_jx_j + \sum_{j=1}^{s} d'_jy_j \mid \sum_{j=1}^{r} \varphi(a_j)x_j + \varphi\Big(\sum_{j=1}^{s} b_jy_j\Big) = \varphi(b),\ x_j \in Z^+,\ j = \overline{1,r},\ y_j \geq 0,\ j = \overline{1,s}\Big\},$$

групповая релаксация смешанной целочисленной линейно задачи

$$\min\{cx + dy \mid Ax + By = b,\ x \geq 0,\ x_j \equiv 0,\ y \geq 0\}.$$

Дизъюнктивная задача

$$\min\Big\{cx \mid \bigvee_{h \in H} (A^hx \geq b^h),\ x \geq 0\Big\}.$$

Смешанная целочисленная нелинейная задача

$$\min\{f(x,y) \mid g_i(x,y) \geq 0,\ i = \overline{1,m},\ x \geq 0,\ x \equiv 0,\ y \geq 0\}.$$

Задача о ранце

$$\max\{cx \mid ax \leq b,\ x \in Z_n^+\}.$$

Бинарная задача о ранце

$$\max\{cx \mid ax \leq b,\ x \in B_n\}.$$

Многомерная задача о ранце

$$\max\{cx \mid Ax \leq b,\ 0 \leq x \leq d,\ x \equiv 0\}.$$

Задача о покрытии множества

$$\max\{cx \mid Ax \geq e,\ x \in B_n\}.$$

Задача о разбиении множества

$$\max\{cx \mid Ax = e,\ x \in B_n\}.$$

Задача об упаковке множества

$$\max\{cx \mid Ax \le e,\ x \in B_n\}.$$

Задача о β-покрытии множества

$$\max\{cx \mid Ax \ge \beta,\ x \in B_n\}.$$

Задача о β-разбиении множества

$$\max\{cx \mid Ax = \beta,\ x \in B_n\}.$$

Задача о β-упаковке множества

$$\max\{cx \mid Ax \le \beta,\ x \in B_n\}.$$

Задача о потоке минимальной стоимости. Минимизировать

$$\sum_{i \in V} \sum_{j \in V(i)} c_{i,j} x_{i,j}$$

при условиях

$$\sum_{j|inV(i)} x_{i,j} - \sum_{j \in V'(i)} x_{i,j} \begin{cases} \le a_i, & i \in V_1, \\ = 0, & i \in V_2, \\ \le -b_i, & i \in V_2, \end{cases}$$

$$0 \le x_{i,j} \le d_{i,j},\ (i,j) \in E.$$

Задача о назначениях. Максимизировать

$$\sum_{i=1}^{m} \sum_{j=1}^{m} c_{i,j} x_{i,j}$$

при условиях

$$\sum_{j=1}^{m} x_{i,j} = 1,\ i = \overline{1,m},\ \sum_{i=1}^{m} x_{i,j} = 1,\ j = \overline{1,m},\ x_{i,j} = 0,1,\ i = \overline{1,m},\ j = \overline{1,m}.$$

Целочисленная распределительная задача. Максимизировать

$$\sum_{i=1}^{m} \sum_{j=1}^{n} c_{i,j} x_{i,j}$$

при условиях

$$\sum_{i=1}^{m} x_{i,j} = 1,\ j = \overline{1,n}, \quad \sum_{j=1}^{n} a_{i,j} x_{i,j} \ge a_i,$$

$$i = \overline{1,m},\ x_{i,j} \ge 0,\ i = \overline{1,m},\ j = \overline{1,n},\ x_{i,j} \equiv 0,\ i = \overline{1,m},\ j = \overline{1,n}.$$

Задача о выборе оптимального варианта. Минимизировать

$$\sum_{i=1}^{m} \sum_{r=1}^{n_i} c_{i,r} x_{i,r}$$

при условиях

$$\sum_{i=1}^{m} \sum_{r=1}^{n_i} a_{i,r,k} x_{i,r} \ge b_k,\ k = \overline{1,p}, \quad \sum_{r=1}^{n_i} x_{i,r} = 1,\ i = \overline{1,m},\ x_{i,r} = 0,1,\ i = \overline{1,m},\ r = \overline{1,n_i}.$$

Задача о размещении. Минимизировать

$$\sum_{i=1}^{m}\Big(\sum_{j=1}^{n} c_{i,j}x_{i,j} + d_i y_i\Big)$$

при условиях

$$\sum_{i=1}^{m} x_{i,j} = b_j,\ j = \overline{1,n},\ \sum j = 1^n x_{i,j} \le a_i y_i,$$

$$i = \overline{1,m},\ x_{i,j} \ge 0,\ i = \overline{1,m},\ j = \overline{1,n},\ y_i = 0,1,\ i = \overline{1,m}.$$

Литература

[1] Адельсон-Вольский Г.М., Диниц Е.А., Карзанов А.В. Потоковые алгоритмы,—М.: Наука, 1975.—120с.

[2] Ахо А., Хопкрофт Дж., Ульман Дж. Построение и анализ вычислительных алгоритмов: Пер. с англ.—М.: Мир, 1979.—536с.

[3] Базара М., Шетти К. Нелинейное программирование: теория и алгоритмы: Пер. с англ.—М..: Мир, 1982.—584 с.

[4] Барский А. Б. Планирование параллельных вычислительных процессов.—М.: Машиностроение, 1980.—192 с.

[5] Барсук В. А., Губин Н. М. Математические методы планирования и управления в хозяйстве связи.—М.: Связь, 1974.—336с.

[6] Бахтин А. Е. Методы решения задач территориально-производственного планирования.—Новосибирск: Наука, 1976.—235 с.

[7] Болдырев В. И., Хохлюк В. И. Процедура один симплекс-метода для решения общей задачи линейного программирования// Процедуры системы АЛЬФА-6. Информ. оператив. материал.—Новосибирск, 1978—С. 119—124.

[8] Болдырев В. И., Хохлюк В. И. Процедура два симплекс-метода для решения общей задачи линейного программирования // Процедуры системы АЛЬФА-6: Информ. оператив. материал.—Новосибирск, 1978.—С. 125—130.

[9] Булавский В. А., Звягина Р. А., Яковлева М. А. Численные методы линейного программирования (специальные задачи).—М.: Наука, 1977—368 с.

[10] Вальковский В. А. Лекции по параллельному программированию (Параллелизм в ЭВМ и языках программирования): Учеб. пособие.—Новосибирск: НГУ, 1982.—88 с.

[11] Виноградов И. М. Основы теории чисел.—М.: Наука, 1981—176 с.

[12] Виноградова Т. Д. Целочисленная распределительная задача // Изв. АН СССР. Техн. кибернетика.—1969.—№ 4.—С. 47—53.

[13] Вычислительные системы/ Под ред. Э. В. Евреиова.—М.: Финансы и статистика.—Вып 2, 1981, 168 с. Вып 3, 1982, 144 с. Вып. 4, 1983, 143 с.

[14] Головкин Б. А. Параллельные вычислительные системы.—М.: Наука, 1980.—520 с.

[15] Гольштейн Е. Г., Юдин Д. Б. Задачи линейного программирования транспортного типа.—М.: Наука, 1969.—384 с.

[16] Гэри М., Джонсон Д. Вычислительные машины и труднорешаемые задачи: Пер. с англ.—М.: Мир, 1982.—416 с.

[17] Димитриев Ю. К., Хорошевский В. Г. Вычислительные системы из мини-ЭВМ.–М.: Радио и связь, 1982.—304 с.

[18] Евреинов Э. В. Однородные вычислительные системы, структуры и среды.—М.: Радио и связь, 1981.—208 с.

[19] Евреинов Э. В., Хорошевский В. Г. Однородные вычислительные системы.—Новосибирск: Наука, 1978.—316с.

[20] Емеличев В. А., Ковалев М. М.. Кравцов М. К. Многогранники, графы, оптимизация.—М.: Наука, 1981.—342 с.

[21] Емеличев В. А., Комлик В. И. Метод последовательности планов для решения задач дискретной оптимизации.—М.: Наука 1981.—208 с.

[22] Исследование операций: Пер. с англ. В 2-х т./Под ред. Дж- Моудера, С. Элмаграби,—М.: Мир, 1981.—Т. 1.—712 с.; Т. 2.—679 с.

[23] Каргаполов М. И., Мерзляков Ю. И. Основы теория групп.—М. Наука, 1982.—288 с.

[24] Кнут Д. Искусство программирования для ЭВМ: Пер. с англ. В 3-х т.—М.: Мир, 1976—1978.—Т. 2. Получисленные алгоритмы, 1977.—724 с,

[25] Кофман А., Дебазей Г. Сетевые методы планирования: Пер. с франц.—М.: Прогресс, 1968.—182 с.

[26] Кристофидес Н. Теория графов. Алгоритмический подход; Пер. с англ.—М.: Мир, 1978.—432 с.

[27] Лебедева Т. Т., Михалевич В, С., Рощин В. А. и др. Структура, состав и назначение пакета программ ДИСПРО для решения задач дискретной оптимизации // Изв. АН СССР, Техническая кибернетика,— 1983.—№ 1. С. 193—196.

[28] Липаев В. В. Распределение ресурсов в вычислительных системах.—М.; Статистика, 1979.—247 с.

[29] Лотов А. В. Введение в экономико-математическое моделирование.—М.: Наука, 1984.—392 с.

[30] Лэсдон Л. С. Оптимизация больших систем: Пер. с англ.—М.: Наука, I975.—432 с.

[31] Михалевич В. С., Кукса А. И. Методы последовательной оптимизации в дискретных сетевых задачах оптимального распределения ресурсов.—М.: Наука, 1983.—208 с.

[32] Михалевич В. С., Сергиенко И. В., Лебедева Т. Т. и др. Пакет прикладных программ ДИСПРО, предназначенный для решения задач дискретного программирования //Кибернетика.—1981.—№ З.—С. 117—137.

[33] Михалевич В. С., Сергиенко И. В., Трубин В. А. и др. Пакет прикладных программ для решения задач производственно-транспортного планирования большой размерности (ПЛАНЕР) // Кибернетика.— 1983.—№ 3.—С. 57—71.

[34] Муртаф Б. Современное линейное программирование. Теория и практика: Пер. с англ.—М.: Мир, 1984.—222 с.

[35] Пакеты прикладных программ: Методы оптимизации/ Под ред. А. А. Самарского.–М.: Наука, 1984.-161 с.—Алгоритмы к алгоритмические языки).

[36] Пападимитриу Х., Стайглиц К. Комбинаторная оптимизация. Алгоритмы и сложность: Пер. с англ.—М.: Мир, 1985.—512 с.

[37] Прангишвили И. В., Виленкин С. Я., Медведев И. Л, Параллельные вычислительные системы с общим управлением.—М.: Энергоатомиздат, 1983.—312с.

[38] Применение пакетов прикладных программ по экономико-математическим методам в АСУ/ Б. Я. Курицкий, Г. П. Алексеенко, Ю. В. Виткин и др.—М.; Статистика, 1980.—200 с.

[39] Пярнпуу А. А., Хохлюк В. И. Параллельные вычисления в прикладных задачах//Сообщение по прикладной математике/Вычислительный центр АН СССР—М., 1985.—64 с.

[40] Самофалов К. Г., Луцкий Г. М. Основы построения конвейерных ЭВМ. —Киев: Вища школа, 1981.—224 с.

[41] Сергеев С. И. Алгоритмы решения задачи коммивояжёра (обзор)// Моделирование технико-экономических процессов, 1981.—с. 3-37

[42] Сергиенко И. В., Лебедева Т. Т., Рощин В. А., Приближённые методы решения дискретных задач оптимизации.—Киев: Наукова думка, 1980.—273 с.

[43] Системы параллельной обработки: Пер. с англ./Под ред. Д. Ивенса.—М.: Мир, 1985.—416 с.

[44] Слисенко А. О. Сложностные задачи теории вычислений // Успехи мат. наук.—1981.—Т. 36.—Вып. 6(222).—С. 21—103.

[45] Справочник по оптимизационным задачам в АСУ/В. А. Бункин, Д. Колев, Б. Я. Курицкий и др.—Л.: Машиностроение, 1984.—212 с

[46] Танаев В. С., Гордон В. С., Шафранский Я. М. Теория расписаний. Одностадийные системы.—М.: Наука, 1984.—384 с.

[47] Теория расписаний и вычислительные машины: Пер. с англ./ Под ред. Э. Г. Коффмана.—М.: Наука, 1984.—335 с.

[48] Фаддеева В. Н., Фаддеев Д. К. Параллельные вычисления в линейной алгебре//Кибернетика.—1977.—№ 6.—С. 28—40.

[49] Фаддеева В. Н., Фаддеев Д. К. Параллельные вычисления в линейной алгебре. II//Кибернетика.–1982.—№ З.—С. 18—31.

[50] Хохлюк В. И. Процедура для нахождения наибольшего общего делителя и коэффициентов разложения // Алгоритмы и программы: Информ. бюлл.—1979.—№ 1.—С. 20.

[51] Хохлюк В. И., Ясенев М, В. Процедура для нахождения всех крайних лучей конуса // Алгоритмы и программы: Информ. бюлл.—1979.—№2.—С. 21—22.

[52] Хохлюк В. И., Ясенев М. В. Процедура для нахождения всех вершин и неограниченных направлений многогранника//Алгоритмы и программы: Информ. бюлл.—1979.—№ 2.—С. 21.

[53] Хохлюк В. И. Задачи целочисленной оптимизации (преобразования): Учеб. пособие.—Новосибирск: НГУ, 1979.—92с.

[54] Хохлюк В. И. Задачи целочисленной оптимизации (алгоритмы): Учеб. пособие. — Новосибирск: НГУ, 1980.—92 с.

[55] Хохлюк В. И. О параллелизме в целочисленной оптимизации// Вычислительные системы.—1981. Вып. 89,—С. 3—22.

[56] Хохлюк В. М. Алгоритмы целочисленной оптимизаций (Вспомогательные АЛГОЛ-процедуры): Учеб. пособие.—Новосибирск: НГУ, 1982.—92 с.

[57] Хохлюк В. И. Алгоритмы секущих плоскостей: Учеб, пособие.—Новосибирск: НГУ, 1983.—92 с.

[58] Ху Т. С. Целочисленное программирование и потоки в сетях: Пер. с англ.—М.: Мир, 1974.—519 с.

[59] Черников С. Н. Линейные неравенства.—М.: Наука, 1968.—488 с.

[60] Шихаев К. Н. Разностные алгоритмы параллельных вычислительных процессов.—М.: Радио и связь, 1982.—136 с.

[61] Шор Н. З. Методы минимизации недифференцируемых функций и их приложения.—Киев: Наукова думка, 1979.—200 с.

[62] Элементы параллельного программирования / В. А. Вальковский, В. Е. Котов, А. Г. Марчук, Н. Н. Миренков,—М.; Радио и связь, 1983.—240 с.

[63] Эльстер К. Х., Рейнгардт Р., Шойбле М., Донат Г. Введение в нелинейное программирование: Пер. с нем.—М.: Наука 1985.—264 с.

[64] Один Д. Б., Горяшко А. П., Немировский А. С. Математические методы оптимизации устройств и алгоритмов АСУ.—М.:Радио и связь, 1982.—288 с.

[65] Balas E. Integer Programming and Convex Analysis Intersection Cuts from Outer Pollars//Math. Programming.—1972.—Vol. 2, N 3.—P. 330—382.

[66] Balas E., Padberg M. W. Set Partitioning-A Survey//Combinatorial Optimization.—Chichester, 1979.—P. 151—210.

[67] Beale E. M. L, Branch and Bound Methods for Mathematical Programming Systems//Discrete optimization. II.—Amsterdam,. 1979.—P. 201—219.

[68] Foulds L. R. Combinatorial Optimization.—Berlin—Heidelberg—New York—Tokyo: Springer-Verlag, 1984.—280 p.

[69] Garfinkel R. S., Nemhauser G. L. Integer Programming.—New York—London—Sydney—Toronto: J. Wiley Sons, 1972.—428 p.

[70] Integer Programming and Related Areas: A Classified Bibliography/Ed. C. Kosting.—Berlin—Heidelberg—New York—Springer-Verlag, 1976.—495 p.

[71] Integer Programming and Related Areas; A Classified Bibliography. 1976—1978/Ed.Hausmann.—Berlin—Heidelberg—New York; Springer-Verlag, 1978.—314 p.

[72] Integer Programming and Related Areas; A Classified Bibliography. 1978—1981/Ed. R. von Randow.—Berlin—Heidelberg—New York: Springer-Verlag, 1982.—338 p.

[73] Jeroslow R. Cutting-Plane Theory: Algebraic Methods // Discrete-Mathematics.—1978.—Vol. 23, N 2. P. 121—150.

[74] Kuck D. J. Parallel Processing of Ordinary Program//Adv. Comput.—1976.—N 15.—P. 119—179.

[75] Lambiotte J. J., Volgt R. G. The Solution of Tridiagonal Linear Systems on the CDC STAR-100 Computer//ACM Trans Math Software.—1975.—Vol. 1, N 4.—P.308—329.

[76] Land A., Powell S. Computer Codes for Problem of Integer Programming//Discrete Optimization. II.—Amsterdam 1979.—P. 221—269.

[77] Martello S., Toth P. The 0-1 Knapsack Problem//Comblnatoriali Optimization.—Chichester, 1979.—P. 237—279.

[78] Non-linear Parametric Optimization/Bank B., Guddat J., Klatte D., Kummer B., Tammer K.—Berlin: Akademle-Verlag 1982.—228 p.

[79] Parallel Computations/Ed. by G. Rodrigue.—New York and Academic Press, 1982.—404 p.

[80] Pierce J. F., Crowston W. B. Tree-search Algorithms for Quadratic-Assignment Problems//Nav. Res. Log. Quart.—1971.—Vol. 18.—N 1.—P. 1—36

[81] Stelnberg D. 1. The Fixed Charge Problem//Nav. Res. Log-Quart.—1970.—Vol. 17, N 2.—P. 217—235.

[82] Zakharov V. Parallelism and Array Processing//IEEE Trans.—1984. —Vol.C-33, N 1.—P. 45—78.

Оглавление