ЛИТЕРАТУРА

1. *Л. Форд, Д. Фалкерсон.* Потоки в сетях. М., «Мир», 1966.
2. *С. М. Бородкин.* О решении минимаксной задачи о назначении.— «Автоматика и телемеханика», 1974, № 10.
3. *O. Gross.* The bottleneck assignment problem.— «RAND Corporation», Paper P-1630, 1959, March 6.
4. *Е. А. Диниц.* Алгоритм решения задачи о максимальном потоке в сети со степенной оценкой.— «Доклады АН СССР», 1970, т. 194, № 4.
5. *А. В. Карзанов.* Точная оценка алгоритма нахождения максимального потока, примененного к задаче «о представителях».— «Вопросы кибернетики». Труды семинара по комбинаторной математике. М., «Наука», 1973.
6. *R. S. Garfinkel.* An improved algorithm for the bottleneck assignment problem.— «Operations Research», 1971, v. 19, p. 1747—1751.

Е. А. Диниц

О РЕШЕНИИ ДВУХ ЗАДАЧ О НАЗНАЧЕНИИ

1. Введение. Классическая задача о назначении широко известна: если есть n работ и n исполнителей и заданы эффективности a_{ij} выполнения i-й работы j-м исполнителем, $i, j = 1, \ldots, n$, то требуется сделать назначение «на должности» π: $\pi(i) = \pi(j) \Leftrightarrow i = j$, такое, что суммарная эффективность $\sum_i a_{i\pi(i)}$ максимальна. Наиболее низкую оценку трудоемкости при решении этой задачи — $O(n^3)$ действий [1] — имеют алгоритмы, предложенные в работах [1], [3].

Мы рассмотрим естественное обобщение — несбалансированную задачу о назначении, характеризующуюся тем, что число исполнителей m больше числа работ n,

[1] Под действием понимается элементарная операция ЭВМ.

так что назначаются на работу не все, а только некоторые n из m исполнителей, заранее неизвестно какие. (Замечание: если число исполнителей меньше числа работ, то получается задача, эквивалентная рассматриваемой.)

Предлагаемый метод решения несбалансированной задачи о назначении позволяет решать ее с оценкой трудоемкости $O(n^3) + O(mn)$ действий или, если матрица эффективностей $\{a_{ij}\}$: а) неотрицательна; б) задана не только в адресном виде, но еще и списками ненулевых элементов в строках,— с оценкой $O(n^3) + O(p)$ действий, где p — общее количество ненулевых элементов a_{ij}.

Эта оценка, очевидно, минимально возможная при условии, что оценка для классической задачи о назначении — $O(n^3)$ действий. Действительно, первое слагаемое необходимо по условию, а второе соответствует просмотру всей информации: если в алгоритме игнорируется хотя бы один элемент a_{ij}, мы не гарантированы от того, что его значение столь велико, что $\pi(i) = j$ с необходимостью.

Предлагаемый метод имеет оценку $O(n^3)$ действий, если в исходной информации ненулевые элементы каждой строки заданы в порядке убывания (достаточно $n + 1$ максимальных элементов в каждой строке).

1. Опишем сначала алгоритм решения несбалансированной задачи о назначении, основанный на традиционных идеях. Ввиду его близости алгоритму из работы [1] и венгерскому алгоритму [2] обоснования не приводим.

Терминология далее будет чисто матричной. Поднабором в (n $\times$ m)-матрице $\{a_{ij}\}$ $(n \leqslant m)$ называем множество выделенных элементов: не более чем по одному в каждой строке и в каждом столбце. Набор — это поднабор из n элементов, т. е. такой, что в каждой строке выделен элемент. В задаче требуется найти набор с максимальной суммой элементов. (Замечание: задача на минимум аналогична задаче на максимум.)

В каждый момент алгоритма будет задан вектор потенциалов столбцов $\Delta = \{\Delta_i\}$, $1 \leqslant j \leqslant m$. Введем по-

нятие «Δ-разность»: $a_{ij_1} \overset{(\Delta)}{-} a_{ij_2} = (a_{ij_1} - \Delta_{j_1}) - (a_{ij_2} - \Delta_{j_2})$. Соответственно будем пользоваться понятиями «Δ-больше» и «Δ-максимальность».

Свободными будут называться столбец или строка, не содержащие выделенных элементов.

Алгоритм. Предварительный этап. Находим в каждой строке максимальный элемент. В столбцах, содержащих хотя бы один такой элемент, выделяем один из этих элементов. Получаем исходный поднабор. Полагаем потенциалы равными нулю.

Итерация. Исходная ситуация. Имеется поднабор из k, $1 \leqslant k \leqslant n$, элементов и вектор потенциалов, так что:

а) каждый выделенный элемент (т. е. элемент поднабора) Δ-максимален в своей строке.

б) $\Delta_j = \text{const}$ на множестве свободных столбцов.

Цель итерации: получить поднабор из $k + 1$ элемента и вектор потенциалов, удовлетворяющие условиям а) и б), с тем, чтобы перейти к следующей итерации, если $k + 1 < n$, или выдать полученный набор в качестве ответа, если $k + 1 = n$.

Алгоритм выполнения итерации. Пусть R обозначает множество свободных и (или) помеченных (см. ниже) столбцов; Q — множество помеченных строк; δ_i — разность между выделенным (Δ-максимальным) элементом i-й строки $a_{i\pi(i)}$ и элементом этой строки, Δ-максимальным среди элементов, принадлежащих столбцам из R; δ — минимум δ_i по непомеченным строкам. В начале итерации все столбцы и строки считаем непомеченными.

Начальный (нулевой) шаг. Вычислим величину δ и рассмотрим экстремальный элемент $a_{i_0j_0}$ (для которого разность, участвующая в определении величины δ_i, равна δ). Увеличим на δ потенциалы всех свободных столбцов (при этом элемент $a_{i_0j_0}$ станет Δ-максимальным в строке и каждый выделенный элемент останется Δ-максимальным в строке).

Если строка i_0 — свободная, то присоединяем элемент $a_{i_0 j_0}$ к поднабору и заканчиваем итерацию. Если она не свободна, то помечаем строку i_0 индексом j_0, помечаем столбец j^0, содержащий выделенный элемент $a_{i_0 j^0}$, индексом i_0, и переходим к 1-му шагу.

Внутренний (k-й) шаг $(1 \leqslant k \leqslant n-1)$. Вычислим величину δ и рассмотрим экстремальный элемент $a_{i_k j_k}$. Увеличим на δ потенциалы всех столбцов из R.

Если строка i_k не свободна, то помечаем строку i_k индексом j_k, помечаем столбец j^k, содержащий выделенный элемент $a_{i_k j^k}$, индексом i_k.

Если строка i_k — свободная, то изменяем поднабор следующим образом. Элемент $a_{i_k j_k}$ включаем в поднабор. Если столбец j_k — свободный, то заканчиваем итерацию. Если же он не свободный, то он помечен индексом i и содержит выделенный элемент $a_{i j_k}$. Удаляем этот элемент из поднабора. Строка i помечена некоторым индексом j. Включаем элемент a_{ij} в поднабор. Далее действуем по той же схеме до свободного столбца.

2. Следуя работам [1], [3], укажем экономный способ вычисления величины δ на каждом внутреннем шаге алгоритма итерации. Следует пользоваться справочной $q = \{q_i\}$. Перед каждым шагом q_i определено, если i-я строка не помечена; q_i состоит из:

1) величины δ_i;

2) номера j (i) такого, что элемент $a_{ij(i)}$ Δ-максимален среди элементов i-й строки, лежащих в столбцах из R. Ясно, что справочная q позволяет нам за $O(n)$ действий: а) находить начальный поднабор на предварительном этапе; б) находить величину δ и экстремальный элемент на любом шаге любой итерации.

Справочная q строится заново перед предварительным этапом и перед каждой итерацией. При переходе от одного шага итерации к следующему исправление справочной требует $O(n)$ действий: если j — столбец, помеченный на последнем шаге, то $q_i := \min\left(q_i,\ a_{i\pi(i)} \overset{(\Delta)}{-} a_{ij}\right)$.

Описанный алгоритм при использовании справочной имеет оценку трудоемкости $O(n^3)$ действий в классическом случае $n = m$ (см. [1], [3]).

3. Попробуем для несбалансированной задачи ($n < m$) так подправить алгоритм, чтобы сделать оценку «по возможности» не зависящей от m. Пусть справочная построена. Тогда трудоемкость предварительного этапа не зависит от m, а на итерациях от m зависит только трудоемкость изменения потенциалов свободных столбцов. С последним обстоятельством легко справиться: так как потенциалы всех свободных столбцов равны между собой, то их общее значение достаточно хранить только в одном определенном месте и изменять его, когда потребуется.

Теперь от m зависит только трудоемкость построения справочных. Предположим, что для каждой строки матрицы $\{a_{ij}\}$ перед решением задачи имеется список $n + 1$ максимальных элементов этой строки, расположенных в порядке убывания (с указанием для каждого из них номера его столбца). Тогда на предварительном этапе построение справочной (нахождение максимальных элементов в строках) тривиально.

Построение справочной перед каждой итерацией при наличии списков максимальных элементов в строках сводится к следующему. Для построения q_i берем очередной элемент i-го списка (на первой итерации — второй) и проверяем, лежит ли он в свободном столбце. Если да, то по нему определяем q_i. Если же он соответствует несвободному столбцу, то переходим к следующему элементу списка, и т. д. Заканчивая работу со списком, запоминаем место взятого элемента списка как очередного элемента для следующей итерации.

В обоснование такого способа построения справочных следует отметить следующее. Во-первых, элементы списка, расположенные перед очередным, всегда лежат в несвободных столбцах. Поэтому δ_i здесь определяется элементом i-й строки, максимальным из элементов, лежащих

в свободных столбцах. Этот элемент также и Δ-максимален среди них, так как для свободных столбцов $\Delta_j = \text{const}$. Во-вторых, так как несвободных столбцов не может быть больше n, то очередной элемент в списке не может иметь номер, больший $n + 1$. Так что длины $n + 1$ для каждого списка хватает.

Оценим трудоемкость построения справочной с использованием списков максимальных элементов в строках. Если обращение к очередному элементу списка оканчивается неудачей, т. е. он принадлежит несвободному столбцу, то мы «потеряли» на этом $O(1)$ действий. Таких потерь всего не более n^2 (общее количество элементов в несвободных столбцах), т. е. всего теряется $O(n^2)$ действий. Трудоемкость каждого построения справочной, не считая этих потерь, очевидно, $O(n)$ действий. Так как итераций не более n, то всего построение справочных требует $O(n^2)$ действий.

Таким образом, при наличии списков $n + 1$ максимальных элементов в строках трудоемкость решения несбалансированной задачи о назначении предложенным алгоритмом составляет $O(n^3)$ действий.

4. Для нахождения $n + 1$ максимальных элементов из m достаточно $O(m + n \log_2 m)$ действий (алгоритм с такой оценкой см. ниже или в [4]). Тем самым общая трудоемкость решения задачи о назначении оценивается как $O(n^3 + mn + n^2 \log_2 m)$ действий.

Для получения оценки $O(n^3 + mn)$, декларированной в начале статьи, нужно показать, что $n \log_2 m = O(n^2 + m)$. Действительно, $n \log_2 m = o(n\sqrt{m})$, а $n\sqrt{m} = \sqrt{n^2 m} \leqslant (n^2 + m)/2$.

В случае, когда матрица a_{ij} неотрицательна и для каждой ее строки задан список ее ненулевых элементов в количестве m_i штук, $p = \sum_i m_i$, была декларирована оценка $O(n^3 + p)$. Она получается аналогично предыдущей из того, что $\sum_i n \log_2 m_i = o\left(\sum_i (n^2 + m_i)\right) = o(n^3 + p)$.

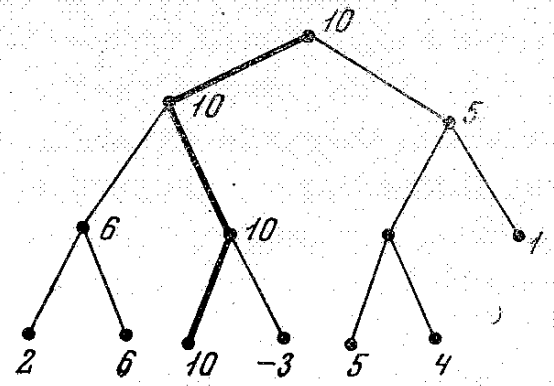

Рис. 1

5. Алгоритм нахождения максимальных элементов в массиве. Один максимальный элемент из m находится за $m - 1$ сравнений (т. е. $O(m)$ действий) по принципу «проигравший выбывает». Если проводить сравнения по олимпийской системе (..., 1/4 финала, полуфинал, финал), то результаты сравнений можно изобразить двоичным деревом, состоящим из $]\log_2 m[$[2] уровней (пример показан на рис. 1).

Пусть мы заменили значение максимального элемента на $-\infty$. Тогда второй по величине элемент станет максимальным. Для его нахождения достаточно получить

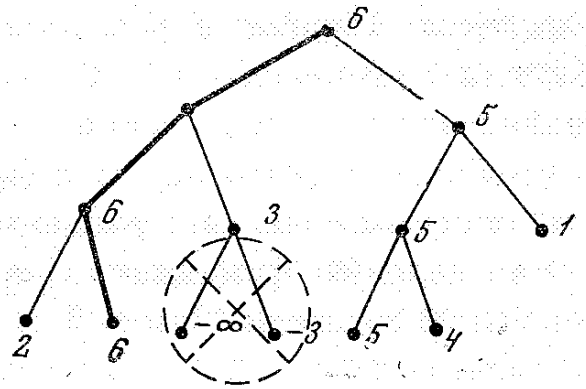

Рис. 2

аналогичное дерево сравнений для преобразованного массива (рис. 2). Такое дерево легко строится исправлением имеющегося. Следует лишь заново провести сравнения, в которых участвовал максимальный элемент. (Этим сравнениям соответствуют вершины дерева, лежащие на пути

[2] $]x[$ есть наименьшее целое, не меньшее x.

из его корня в концевую вершину, соответствующую максимальному элементу). Всего требуется не более $\lceil \log_2 m \rceil$ сравнений, т. е. $O(\log_2 m)$ действий.

Следующие по величине элементы в массиве следует находить последовательно, аналогично нахождению второго по величине. Оценка трудоемкости — $O(m + k \cdot \log_2 m)$ действий для нахождения k максимальных элементов.

6. Ввиду того что приведенный в п. 5 алгоритм находит максимальные элементы последовательно в порядке убывания, можно несколько скорректировать схему решения несбалансированной задачи о назначении: можно не строить заранее списки, а получать очередные максимальные элементы по мере необходимости.

II. Кроме задач о назначении на максимум (минимум) суммы, одной из которых посвящена часть I, широко известна минимаксная задача о назначении, иначе задача о назначении на узкие места (см., например, статью А. Я. Гордона в настоящем сборнике). Рассмотрим класс Σ-минимаксных задач, обобщающий оба традиционных класса задач о назначении.

Пусть имеются n работ и n исполнителей и задана (n × n)-матрица эффективностей $A = \{a_{ij}\}$ выполнения работ исполнителями. Назначение «на должности» — это соответствие $i \to \pi(i)$: $\pi(i_1) = \pi(i_2) \Leftrightarrow i_1 = i_2$. Назначение π определяет набор элементов $N = N(\pi) = \{a_{i\pi(i)}\}$. Как целевую функцию $c = c(N)$, соответствующую назначению π, введем сумму k $(1 \leqslant k \leqslant n)$ максимальных элементов набора $N(\pi)$. Требование Σ-минимаксной задачи о назначении состоит в нахождении назначения, минимизирующего значение функции c. При $k = n$ Σ-минимаксная задача превращается в задачу о назначении на минимум суммы, а при $k = 1$ — в минимаксную.

Σ-минимаксные задачи произошли из потребностей оптимального целераспределения (как отчасти и минимаксные задачи).

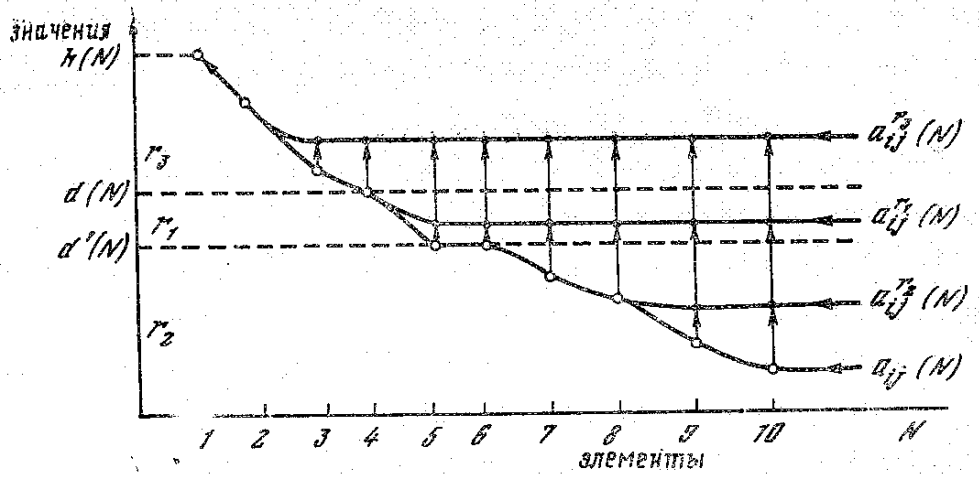

Рис. 3

1. Схема решения задачи. Терминология далее будет чисто матричной. Конечной целью решения задачи будем считать нахождение набора $\overline{N}$, минимизирующего значение функции c. Значение максимального элемента набора N будем обозначать $h(N)$, k-го по величине — $d(N)$, $k+1$-го — $d'(N)$. Определения поднабора и набора см. в части I. Далее r — вещественное число.

(Вспомогательной) r-задачей назовем задачу о назначении на минимум суммы, матрицей для которой служит $A^r = \{a^r_{ij} = \max(r, a_{ij})\}$. Целевую функцию r-задачи обозначим c^r_Σ. Рассмотрим произвольный набор N и выясним, как изменяется значение целевой функции при переходе от исходной задачи к вспомогательной r-задаче.

Рассмотрим три случая (рис. 3).

1) $d'(N) \leqslant r \leqslant d(N)$. Имеем $c^r_\Sigma(N) - c(N) = (n - k)\, r$.

2) $r < d'(N)$. При этом $c^r_\Sigma(N)$ отличается от $c(N)$ на сумму $n - k$ элементов, каждый из которых не меньше r и хотя бы один ($k+1$-й по величине) строго больше r. Так что $c^r_\Sigma(N) - c(N) > (n - k)\, r$.

3) $r > d(N)$. При этом $c^r_\Sigma(N)$ отличается от $c(N)$, во-первых, на сумму $n - k$ элементов, каждый из которых равен r, и, во-вторых, тем, что каждое из остальных k слагаемых $c^r_\Sigma(N)$ не меньше соответствующего слагаемого

$c(N)$ и хотя бы одно из них (k-е по величине) строго больше. Так что и здесь $c^r_\Sigma(N) - c(N) > (n-k)\,r$.

Из проведенных рассмотрений вытекает справедливость следующих утверждений.

Теорема 1. Пусть $\bar{N}$ — оптимальный набор для Σ-минимаксной задачи.

A. $\bar{N}$ — оптимальный набор для вспомогательной $d(\bar{N})$-задачи.

B. Любой оптимальный набор N вспомогательной $d(\bar{N})$-задачи оптимален для исходной задачи.

C. Для любого такого набора N $d'(\bar{N}) \leqslant d(N) \leqslant d(\bar{N})$, т. е., в частности, не менее чем $n-k+1$ элемент набора N не превосходит $d(\bar{N})$.

Утверждение **B** теоремы 1 позволяет принять следующую схему решения Σ-минимаксной задачи о назначении. Пусть $r_1, r_2, \ldots, r_m$ — упорядоченная по возрастанию последовательность различных значений элементов матрицы A. Решив вспомогательные r_l-задачи, $l = 1, 2, \ldots, m$, вычислив для оптимального набора каждой из них значение функции c и выбрав из них набор с минимальным ее значением, получим ответ исходной задачи.

Приведенную схему (далее — r-схему) можно реализовать алгоритмом с оценкой трудоемкости $O(kmn^2)$ (т. е., вообще говоря, $O(kn^4)$ машинных действий) [3]. Для случая $k = 2$ модифицированный алгоритм имеет оценку $O(n^3)$ действий.

2. Реализация r-схемы. Каждую из r_l-задач можно решать, например, с помощью алгоритма [1]. Этот алгоритм затрачивает по $O(n^2)$ действий на итерацию, а количество итераций равно n минус количество элементов начального поднабора. В качестве начального можно взять любой поднабор, каждый элемент которого минимален в своей строке. Покажем, как построить алгоритм, чтобы

---

[3] Как указывает С. М. Бородкин в [5], класс задач, для которых m невелико, практически важен.

начальный поднабор всегда содержал $n - k + 1$ элемент (это обеспечит декларированные выше оценки трудоемкости).

Пусть $X^r = \{a_{ij} \mid a_{ij} \leqslant r \Leftrightarrow a_{ij}^r = r\}$. В силу утверждения **С** теоремы 1 достаточно решить только те r_l-задачи, для которых множество X^{r_l} содержит поднабор P из $n - k + 1$ элемента. Иначе говоря, если $\tilde{l}$ — минимальный номер такой задачи, то нужно решать r_l-задачи при $l \geqslant \tilde{l}$. Найти $\tilde{l}$ и поднабор P из $n - k + 1$ элемента, принадлежащий $X^{r_{\tilde{l}}}$, можно методом, предложенным А. Я. Гордоном для решения минимаксной задачи о назначении (см. статью в настоящем сборнике). Для наших потребностей следует в алгоритме Гордона остановиться в тот момент, когда будет получен поднабор P из $n - k + 1$ элемента. Трудоемкость такого предварительного этапа составит всего $O(n^3)$ действий (оценка А. Я. Гордона).

Поднабор P принадлежит любому X^{r_l}, $l \geqslant \tilde{l}$, так как $X^{r_l} \supseteq X^{r_{\tilde{l}}} \supseteq P$. Поэтому каждая из матриц A^r допускает P в качестве начального поднабора, что и требовалось.

В заключение отметим следующее. Вспомогательные r_l-задачи следует решать в порядке возрастания l: $l = \tilde{l}, \tilde{l} + 1, \ldots$. Тогда если оптимальный набор $\hat{N}$ некоторой $r_{\hat{l}}$-задачи будет целиком содержаться в множестве $X^{r_{\hat{l}}}$, то следующие r_l-задачи: $l = \hat{l} + 1, \ldots, m$, можно не решать. Действительно, рассмотрим оптимальный набор $\overline{N}$ исходной задачи. Если бы выполнялось $d(\overline{N}) > r_{\hat{l}}$, то набор $\overline{N}$ был бы хуже набора $\hat{N}$ в смысле функции c, чего не может быть. С другой стороны, неравенство $d(N) \leqslant r_{\hat{l}}$ означает, что $d(\overline{N})$-задача уже решалась и оптимальный для исходной задачи набор уже найден, что и требовалось.

3. Схема решения при $k = 2$. Начало решения, как и в общем алгоритме,— предварительный этап, состоящий в нахождении индекса $\tilde{l}$ и построении поднабора P из

$n - k + 1 = n - 1$ элемента, лежащего в множестве $X^{r\tilde{l}}$. Ниже предлагается r'-схема и дается ее реализация, требующая всего $O(n^2)$ действий (без учета трудоемкости предварительного этапа).

Назовем набор N r-правильным, если все его элементы, кроме, быть может, максимального, принадлежат множеству X^r. Положим функцию $c' = c'(N)$ равной значению $h(N)$ максимального элемента набора N. Определим (вспомогательную) r'-задачу как задачу нахождения r-правильного набора, минимизирующего функцию c'.

Теорема 2. Пусть $\overline{N}$ — оптимальный набор для Σ-минимаксной задачи при $k = 2$.

А. Любой оптимальный набор N' вспомогательной $(d(\overline{N}))'$-задачи оптимален для исходной задачи.

В. Для такого набора $h(\overline{N}') = h(\overline{N})$, $d(\overline{N}') = d(\overline{N})$.

Доказательство. Набор $\overline{N}$ $(d(\overline{N})$-правилен. Поэтому $h(N')$, как минимум функции c' на $d(\overline{N})$-правильных наборах, не превосходит $h(\overline{N})$. Так как набор N' $d(\overline{N})$-правилен, $d(N') \leqslant d(\overline{N})$. Тем самым $c(N') = h(N') + d(N') \leqslant h(\overline{N}) + d(\overline{N}) = c(\overline{N})$. Так как $c(\overline{N})$ минимально возможна, неравенства превращаются в равенства, что и доказывает оба утверждения теоремы.

Утверждение **А** теоремы 2 позволяет для Σ-минимаксной задачи при $k = 2$ заменить в r-схеме r_l-задачи на $(r_l)'$-задачи.

Утверждение **В** позволяет сделать больше: вместо решения $(r_l)'$-задачи находить только максимальный элемент $a_{ij}(l)$ оптимального для $(r_l)'$-задачи набора N_l'. Действительно, при этом найти $(r_l)'$-задачу, доставляющую оптимальный набор для исходной задачи, можно, найдя минимум по l сумм $a_{ij}(l) + r_l$ (следствие из утверждения **В**).

В полученной r'-схеме решения Σ-минимаксной задачи при $k = 2$ появляется еще завершающий этап: когда найдено $\overline{l}$, минимизирующее $a_{ij}(l) + r_l$, нужно «дорешать»

$(r_{\bar{l}})'$-задачу. А именно следует построить набор, содержащийся в $X^{r_{\bar{l}}} \cup a_{ij}(l)$. Для этого достаточно огрубить значения элементов матрицы A:

$$A' = \{a'_{ij}\}, \qquad a'_{ij} = \begin{cases} 0, & \text{при} \quad a_{ij} \in X^{r_{\bar{l}}} \cup a_{ij}(l); \\ 1, & \text{при} \quad a_{ij} \notin X^{r_{\bar{l}}} \cup a_{ij}(l) \end{cases}$$

и решить для матрицы A' либо задачу о назначении на минимум суммы, либо минимаксную задачу о назначении, либо задачу о представителях подмножеств. Так как в качестве начального в любом варианте можно взять поднабор P из $n - 1$ элемента, то достаточно одной итерации с трудоемкостью $O(n^2)$ действий.

4. Исследование r'-задачи. Пусть i_0 обозначает номер строки, а j_0 — номер столбца, не содержащих элементов из P (такие строка и столбец единственны). Из общей теории задач о назначении выводится, что оптимальный набор для $(r_l)'$-задачи, $l \geqslant \tilde{l}$, можно получить из поднабора P с помощью некоторой чередующейся цепочки переназначений: включить в поднабор элемент $a_{i_0 j_1} \notin P$, исключить из него элемент $a_{i_1 j_1} \in P$, включить $a_{i_1 j_2} \notin P$, . . ., исключить $a_{i_q j_q} \in P$, включить $a_{i_q j_0} \notin P$. Так как набор должен быть r_l-правильным, то не более чем один (далее — «особый») элемент такой цепочки может не принадлежать X^{r_l}. Чередующиеся цепочки, обладающие этим свойством, будем называть r_l-правильными.

Ясно, что если набор N получен из поднабора P с помощью r_l-правильной цепочки с особым элементом a_{ij}, то a_{ij} — максимальный элемент в N. Если же в ходе последовательного рассмотрения ($l = \tilde{l}, \tilde{l} + 1, \ldots$) набор N впервые получен с помощью r_l-правильной цепочки без особого элемента, то $h(N) = r_l$. Таким образом, для осуществления r'-схемы достаточно для $l = \tilde{l}, \tilde{l} + 1, \ldots$ находить минимальные элементы $a_{ij}(l)$ r_l-правильных цепочек, пока не получится $a_{ij}(l) = r_l$.

Пусть $S = S(r)$ — множество «i_0-достижимых» строк: таких строк i, что существует чередующаяся цепочка, лежащая в X^r, начинающаяся с элемента строки i_0 и кончающаяся элементом строки i. Аналогично определим множество i_0-достижимых столбцов $S' = S'(r)$. Используя цепочки, заканчивающиеся элементом столбца j_0, определим аналогично множества j_0-достижимых столбцов — $T = T(r)$ и j_0-достижимых строк — $T' = T'(r)$.

Ясно, что особый элемент r-правильной цепочки принадлежит подматрице $S(r) \times T(r)$, и обратно; для каждого элемента этой подматрицы существует содержащая его r-правильная цепочка. Поэтому r'-схему можно реализовать как последовательное нахождение в подматрицах $S(r_l) \times T(r_l)$, $l = \tilde{l}, \tilde{l}+1, \ldots$, минимальных элементов.

5. Алгоритм последовательного нахождения минимальных элементов в подматрицах $S(r_l) \times T(r_l)$. Предположим сначала, что имеется алгоритм, строящий множества $S(r_l)$ и $T(r_l)$ для последовательных значений $l = \tilde{l}, \tilde{l}+1, \ldots$, начиная с $S = T = \phi$. Так как множества S и T только растут, то метод вычисления текущего минимума ясен: когда к S (к T) добавляется строка (столбец), то следует просмотреть все ее (его) элементы, сравнивая их со значением текущего минимума и исправляя его, если требуется.

Покажем теперь, как наращивать множества S и T соответственно наращиванию множества $X = X^r$ при переходе к последовательным значениям $r = r_{\tilde{l}}, r_{\tilde{l}+1}, \ldots$. Пусть при $r = r_l$ в множестве X появился новый элемент a_{ij}. Если i-я строка не принадлежит S (кратко, $i \notin S$) и $j \notin T$, то не делаем ничего.

Пусть $i \in S$ (случай $j \in T$ аналогичен). Если $j \in T$, то имеем $a_{ij} \leqslant r$ & $a_{ij} \in S(r_l) \times T(r_l)$, т. е. следует перейти к завершающему этапу. Если $j \notin T$ и $j \in S'$, то не делаем ничего. Если же $j \notin T$ и $j \notin S'$, то включаем j-й столбец в S', находим в нем элемент $a_{i_1 j} \in P$ ($j \neq j_0$,

так как $j_0 \in T$) и включаем i^1-ю строку в S. Далее просматриваем i^1-ю строку и для каждого ее элемента, принадлежащего X, делаем то же, что и для элемента a_{ij}. Заканчиваем эту деятельность, когда все новые строки в S просмотрены.

Обоснование алгоритма не приводим.

Оценка трудоемкости алгоритма — $O(n^2)$ действий. Ее справедливость основывается на том факте, что алгоритм состоит из просмотров строк и столбцов матрицы (каждого — конечное число раз) и отдельных элементов матрицы (каждого — конечное число раз), причем одноразовая деятельность с элементом матрицы требует конечного числа действий.

6. Оценка трудоемкости для $k = 2$. Эта оценка складывается из оценки $O(n^3)$ для предварительного этапа, $O(n^2)$ — для основной части алгоритма (п. 5), $O(m)$, или $O(n^2)$, на вычисление минимума сумм $a_{ij}(l) + r_l$ и $O(n^2)$ для завершающего этапа (п. 4.). Следовательно, общая оценка решения Σ-минимаксной задачи о назначении при $k = 2$ — $O(n^3)$ действий. Тем самым с точностью до величин меньшего порядка трудоемкость при $k = 2$ совпадает с трудоемкостью при $k = 1$ (при решении минимаксной задачи о назначении).

Замечание. Часто при постановке задачи в исходных матрицах имеются недопустимые элементы (со значениями $+\infty$). Пусть число допустимых элементов исходной матрицы — $p \leqslant n^2$ и для каждой строки и каждого столбца имеется список допустимых элементов в них. Как указывает А. Я. Гордон, оценка его алгоритма при этом имеет вид $O(np)$ действий. Можно проверить, что остальные части алгоритма решения Σ-минимаксной задачи при $k = 2$ оцениваются в $O(p)$ действий. Таким образом, общей оценкой становится $O(np)$ действий.

ЛИТЕРАТУРА

1. *Е. А. Диниц, М. А. Кронрод.* Один алгоритм решения задачи о назначении.— «Доклады АН СССР», 1969, т. 189, № 1.
2. *Г. Кун.* Венгерский метод решения задачи о назначении.— «Методы и алгоритмы решения транспортной задачи». Госстатиздат, 1963.
3. *I. Edmonds, R. M. Karp.* Theoretical improvements in algorithmic efficiency for network flow problems.— «J. of ACM», 1972, v. 19, N 2.
4. *В. П. Гришухин.* Об одном классе алгоритмов сравнения.— «Исследования по дискретной математике». М., «Наука», 1973.
5. *С. М. Бородкин.* О решении минимаксной задачи о назначении.— «Автоматика и телемеханика», 1974, № 10.

А. В. Карзанов

СПРАВОЧНАЯ ДЛЯ ВЫБОРКИ МАКСИМАЛЬНОГО ЭЛЕМЕНТА И ЕЕ ПРИЛОЖЕНИЯ

1. При алгоритмическом решении ряда задач дискретной математики возникает потребность в решении следующей задачи А.

Имеется совокупность объектов, каждому из которых отнесено число. Производится многошаговый процесс. На каждом шаге требуется выбрать объект с максимальным (минимальным) числом. На шаге из совокупности удаляются некоторые объекты и добавляются новые (не бывшие ранее).

Пусть N — число объектов, побывавших в совокупности за рассматриваемое число шагов. В [1] описан способ организации информации, применяя который можно решить задачу А с оценкой числа действий $O(N \cdot \log_2 N)$.

Предлагается другой алгоритм решения задачи А с той же оценкой — метод неоднородных справочных. Спра-