В. А. БУЛАВСКИЙ

ОБ ОДНОМ АЛГОРИФМЕ РЕШЕНИЯ ТРАНСПОРТНОЙ ЗАДАЧИ

Несмотря на то, что для решения транспортной задачи - наиболее простой из специальных задач линейного программирования - разработано сравнительно много различных алгорифмов (см., например, [I-4]), нам кажется, что в них не полностью учтены возможности использования специфики задачи для снижения трудоемкости проведения расчетов на ЭВМ. В настоящей статье делается попытка продвинуться в этом направлении дальше. При этом в предлагаемом алгорифме не налагается часто встречающегося требования разделения ингредиентов задачи на две группы (фигурирующего в классической постановке транспортной задачи), а решается задача на произвольной сети, состоящей из направленных звеньев. Поскольку ненаправленное звено представимо как два направленных, а избежать удвоения информации при реализации на ЭВМ можно программным путем, то последнее ограничение не является обременительным.

Часто рассматривается транспортная задача с дополнительными ограничениями сверху на интенсивности перевозок по ряду звеньев. Поскольку эти ограничения здесь могут быть учтены точно так же, как это делается в общей задаче линейного программирования, без увеличения реальной размерности задачи, то для большей ясности изложения будет рассмотрена чистая транспортная задача без дополнительных ограничений.

Математически транспортная задача на сети из r пунктов и N звеньев может быть сформулирована следующим образом.

Заданы числа $p_1, p_2, \ldots, p_r$ и N троек

$$(c^{(\nu)}, i^{(\nu)}, j^{(\nu)}), \quad \nu = 1, 2, \ldots, N, \tag{I}$$

где $c^{(\nu)}$ – вещественные числа, $i^{(\nu)}, j^{(\nu)} \in \{1, 2, \ldots, r\}$.

Требуется определить числа $x^{(\nu)}$, $\nu = 1, 2, \ldots, N$, из условий:

а) $x^{(\nu)} \geq 0, \quad \nu = 1, 2, \ldots, N;$

б) $\sum\limits_{j^{(\nu)}=\kappa} x^{(\nu)} - \sum\limits_{i^{(\nu)}=\kappa} x^{(\nu)} = p_\kappa, \quad \kappa = 1, 2, \ldots, r;$

в) достигает минимума величина

$$\sum_{\nu=1}^{N} c^{(\nu)} x^{(\nu)}. \tag{2}$$

При этом предполагается, что $\sum\limits_{\kappa=1}^{r} p_\kappa = 0$ и $i^{(\nu)} \neq j^{(\nu)}$ при всех ν.

Для удобства изложения в дальнейшем тройки символов (I) будут интерпретироваться как з в е н ь я сети, соединяющие н а ч а л ь н ы й п у н к т звена $i^{(\nu)}$ с его к о н е ч н ы м п у н к т о м $j^{(\nu)}$, а числа $x^{(\nu)}$ – как п о т о к и по соответствующим звеньям.

Будем называть ц е п о ч к о й, связывающей пункт κ_1 с другим пунктом κ_2, конечную последовательность звеньев (I), в которой первое и последнее звено (и только они) содержат пункты κ_1 и κ_2 соответственно, любые два соседних звена имеют общий пункт и никакие два несоседние звена общих пунктов не имеют. В цепочке, соединяющей пункт κ_1 с пунктом κ_2, каждое звено имеет определенную ориентацию: прямую, если его начальный пункт входит в предыдущее звено или совпадает с κ_1, и обратную в противном случае.

Б а з и с о м называется совокупность из $r-1$ звеньев, в которой любые два пункта связаны одной (и только одной) цепочкой. По любому базису можно однозначно определить потоки $x^{(\nu)}$ так, чтобы они отличались от нуля лишь на звеньях базиса и были выполнены условия "б". Если при этом окажутся выполненными и условия "а", то базис называется допустимым. В дальнейшем с каждым базисом будут связываться так определенные числа $x^{(\nu)}$.

В излагаемом алгорифме предполагается, что в решаемой задаче имеются допустимые базисы. При этом мы не останавливаемся на очевидных условиях существования такого базиса, равно как условиях существования решения (при наличии отрицательных $c^{(\nu)}$) и вопросах возможности зацикливания процесса в ситуациях вырождения.

Известно, что решение поставленной экстремальной задачи можно построить на звеньях некоторого базиса, положив остальные $x^{(\nu)}$ равными нулю. Такой базис называется оптимальным. В алгорифме оптимальный базис находится последовательным переходом от одного допустимого базиса к другому, как это делается в работах [2, 4] .

Для каждого базиса могут быть определены (с точностью до постоянного слагаемого однозначно) числа y_k, $k=1,2,\ldots,r$, называемые п о т е н ц и а л а м и п у н к т о в, такие, что

$$y_{j(\nu)} - y_{i(\nu)} = c^{(\nu)},$$

если звено с номером ν входит в базис. При этом выполнение условий

$$y_{j(\nu)} - y_{i(\nu)} \leqslant c^{(\nu)}, \quad \nu = 1,2,\ldots,N , \qquad (3)$$

свидетельствует об оптимальности рассматриваемого базиса [2]. Если же одно из неравенств (3) нарушено, то для перехода к новому допустимому базису нужно включить в базис соответствующее звено и одновременно удалить то звено цепочки, связывающей начальный пункт вводимого звена с его конечным пунктом, которое имеет в ней прямую ориентацию и минимальный поток.

Излагаемый ниже алгорифм характеризуется тем, что очередной базис хранится выписанным в таком порядке, что не нарушена очередность звеньев ни в одной из цепочек, связывающих пункт 1 с другими пунктами. В остальном порядок звеньев может быть произвольным.

Указанный порядок следования звеньев базиса достигается тогда и только тогда, когда один из пунктов каждого звена либо имеет номер 1, либо входит в какое-нибудь звено, предшествующее данному. Этот пункт звена будем называть в е р х н и м, а второй н и ж н и м. При этом каждому звену базиса соотнесем характеристику π, равную единице, если верхним является началь-

ный пункт звена, и нуль в противном случае. Пополненные этой характеристикой упорядоченные звенья базиса составляют с п и с о к A. Таким образом, s-ая строка списка A, которую будем обозначать α_s, имеет вид

$$(\pi_s, c_s, i_s, j_s), \quad s=1,2,\ldots,r-1.$$

Помимо списка A, алгорифм использует еще рабочий с п и с о к B, который состоит из r строк и используется на разных этапах алгорифма. Строку с номером k этого списка будем обозначать через β_k.

Принятый порядок записи звеньев базиса поддерживается при переходе к новому базису, для чего в алгорифме на каждом шаге происходит некоторая перестановка строк списка A. Это позволяет, во-первых, вычислять потенциалы y_k для очередного базиса за один просмотр списка A, во-вторых, не хранить величины $x^{(\nu)}$, а вычислять их по мере надобности, в-третьих, сравнительно просто отыскивать нужные при переходе к новым базисам цепочки.

При описании алгорифма нам удобно будет пользоваться следующей символикой. Запись

$$R \Rightarrow Z \tag{4}$$

следует понимать так: в дальнейшем под символом Z понимать то, что стоит слева от стрелки. В качестве R может стоять либо некоторый символ, либо число, либо формула, по которой вычисляется значение Z. Несколько записей вида (4), разделенные запятыми и взятые в квадратные скобки, означают, что присвоение значений правым частям производится одновременно. Запись вида

$$\text{если } [\ldots], \text{ то } [\ldots]$$

означает, что действия, перечисленные внутри второй пары квадратных скобок, выполняются лишь при удовлетворении всех условий, перечисленных в первой паре скобок. Действия, указания о которых записаны столбиком, производятся последовательно, в порядке их записи. Наконец, фигурная скобка справа с перечислением значений некоторого параметра означает, что данный кусок алгорифма надлежит выполнять последовательно при этих значениях параметра.

I. Вычисление потенциалов

В этой части алгорифма список B используется под запоминание потенциалов пунктов: в строке k списка помещается потенциал пункта k. Поскольку в определении потенциалов имеется некоторый произвол, то можно потенциал первого пункта положить равным нулю. Затем поочередно на всех звеньях базиса по известному потенциалу верхнего пункта на основании формул

$$y_{j_s} - y_{i_s} = c_s, \quad s = 1, 2, \ldots, r-1,$$

определяется потенциал нижнего пункта. Алгорифм вычисления потенциалов может быть кратко записан так:

$$0 \Rightarrow b_1,$$

$$\left.\begin{array}{l} \text{если } [\pi_s = 0], \text{ то } [b_{j_s} - c_s \Rightarrow b_{i_s}] \\ \text{если } [\pi_s = 1], \text{ то } [b_{i_s} + c_s \Rightarrow b_{j_s}] \end{array}\right\} \; s = 1, 2, \ldots, r-1.$$

2. Проверка условий оптимальности

Эта часть алгорифма заключается в проверке условий (3). Если они выполнены, то вычисления заканчиваются и по имеющемуся оптимальному базису надлежит лишь вычислить соответствующие ему потоки. В противном случае будет обнаружено звено

$$(\pi_0, c_0, i_0, j_0),$$

которое следует ввести в базис. При этом полагаем $\pi_0 = 0$.

3. Изменение базиса

Изменение базиса осуществляется за два (вообще говоря, неполных) просмотра списка A в обратном порядке.

При первом просмотре для каждого звена вычисляется поток x. При этом прослеживаются цепочки, соединяющие пункты i_0 и j_0 с пунктом 1, до их слияния (тем самым замыкается цепочка между пунктами i_0 и j_0). На этих цепочках отыскивается удаляемое из базиса звено, т. е. звено, ориентированное от пункта i_c

к пункту j_0, на котором поток минимален. Попутно запоминается, которая из цепочек оказалась порванной ($\varphi = 0$, если цепочка от i_0 порвана, а от j_0 целая; $\varphi = 1$, если наоборот) и количество λ звеньев порванной цепочки, расположенных в списке A после удаляемого звена. Эта часть алгорифма в принятой нами символике выглядит так:

$$\left.\rho_K \Rightarrow \beta_K\right\} \quad K = 1, 2, \ldots, r,$$

$$\left[0 \Rightarrow l_1,\ 0 \Rightarrow l_2,\ +\infty \Rightarrow \varepsilon,\ i_0 \Rightarrow \alpha,\ j_0 \Rightarrow \beta\right],$$

$$\left.\begin{array}{ll}
\text{если } [\pi_s = 0] & \text{то } [-\beta_{i_s} \Rightarrow x,\ \beta_{i_s} + \beta_{j_s} \Rightarrow \beta_{j_s}] \\
\text{если } [\pi_s = 1] & \text{то } [\beta_{j_s} \Rightarrow x,\ \beta_{i_s} + \beta_{j_s} \Rightarrow \beta_{i_s}] \\
\text{если } [\pi_s = 0, i_s = \alpha, x < \varepsilon] & \text{то } [x \Rightarrow \varepsilon,\ s \Rightarrow \gamma,\ 0 \Rightarrow \varphi,\ l_1 \Rightarrow \lambda] \\
\text{если } [\pi_s = 1, j_s = \beta, x < \varepsilon] & \text{то } [x \Rightarrow \varepsilon,\ s \Rightarrow \gamma,\ 1 \Rightarrow \varphi,\ l_2 \Rightarrow \lambda] \\
\text{если } [\pi_s = 0, i_s = \alpha] & \text{то } [l_1 + 1 \Rightarrow l_1,\ j_s \Rightarrow \alpha] \\
\text{если } [\pi_s = 0, i_s = \beta] & \text{то } [l_2 + 1 \Rightarrow l_2,\ j_s \Rightarrow \beta] \\
\text{если } [\pi_s = 1, j_s = \alpha] & \text{то } [l_1 + 1 \Rightarrow l_1,\ i_s \Rightarrow \alpha] \\
\text{если } [\pi_s = 1, j_s = \beta] & \text{то } [l_2 + 1 \Rightarrow l_2,\ i_s \Rightarrow \beta]
\end{array}\right\} \quad s = r-1, r-2, \ldots, \bar{s}$$

Цикл этой части алгорифма заканчивается на том значении параметра $\bar{s}$, после которого окажется $\alpha = \beta$. После этого интересующие нас в дальнейшем символы имеют следующий смысл:

γ – номер строки списка A, в которой находится удаляемое звено;

λ – количество звеньев порванной цепочки, расположенных в списке A после строки γ;

$\varphi = 0$, если порванная цепочка начиналась от i_0;

$\varphi = 1$, если порванная цепочка начиналась от j_0.

Чтобы сформировать новый базис, осталось заменить звено в строке с номером γ на новое и произвести необходимую перестановку списка A. Для этого помещаем вводимое звено, приписав ему признак π, равный φ, в строку с номером $r - \lambda$ списка B и снова просматриваем список A в обратном порядке. При этом, как и раньше, прослеживаем цепочки и выписываем звенья целой цепочки в список B выше вводимого звена, а звенья пор-

ванной цепочки – ниже его, меняя у последних признак π на противоположный. Встречающиеся по пути не принадлежащие цепочкам звенья уплотняем в конец списка A. Такой просмотр списка A продолжается до строки $\gamma+1$, после чего образовавшийся в нем кусок пустых строк заполняется переупорядоченными звеньями цепочек из списка B.

В приводимой ниже записи этой части алгорифма стрелка

$$\stackrel{\pi}{\Longrightarrow}$$

означает пересылку звена с изменением признака π на противоположный.

Если $[\varphi=0]$, то $[(\pi_0,c_0,i_0,j_0)\Rightarrow b_{r-\lambda},\ i_0\Rightarrow\alpha,\ j_0\Rightarrow\beta]$,

если $[\varphi=1]$, то $[(\pi_0,c_0,i_0,j_0)\stackrel{\pi}{\Rightarrow} b_{r-\lambda},\ i_0\Rightarrow\beta,\ j_0\Rightarrow\alpha]$,

$$[r-\lambda\Rightarrow\mu,\ r-\lambda\Rightarrow\nu,\ r\Rightarrow\lambda],$$

если $[\pi_s=0,\ i_s=\alpha]$, то $[j_s\Rightarrow\alpha,\ \mu+1\Rightarrow\mu,\ a_s\stackrel{\pi}{\Rightarrow}b_{\mu+1}]$

если $[\pi_s=0,\ i_s=\beta]$, то $[j_s\Rightarrow\beta,\ \nu-1\Rightarrow\nu,\ a_s\Rightarrow b_{\nu-1}]$

если $[\pi_s=0,\ i_s\neq\alpha,\ i_s\neq\beta]$, то $[\lambda-1\Rightarrow\lambda, a_s\Rightarrow a_{\lambda-1}]$ } $s=r-1, r-2,\ldots,\gamma-1,$

если $[\pi_s=1,\ j_s=\alpha]$, то $[i_s\Rightarrow\alpha,\ \mu+1\Rightarrow\mu,\ a_s\stackrel{\pi}{\Rightarrow}b_{\mu+1}]$

если $[\pi_s=1,\ j_s=\beta]$, то $[i_s\Rightarrow\beta,\ \nu-1\Rightarrow\nu,\ a_s\Rightarrow b_{\nu-1}]$

если $[\pi_s=1,\ j_s\neq\alpha, j_s\neq\beta]$, то $[\lambda-1\Rightarrow\lambda,\ a_s\Rightarrow a_{\lambda-1}]$

$$b_{\nu+\tau}\Rightarrow a_{\gamma+\tau}\Big\}\ \tau=0,1,\ldots,r-\nu.$$

После окончания работы этой части алгорифма список A оказывается заполненным новым базисом с нужным упорядочением звеньев. Теперь надлежит вычислить новые потенциалы и т. д., до получения оптимального базиса.

4. Вычисление потоков

После получения оптимального базиса возникает необходимость вычислить потоки на этом базисе. Приводимый ниже алгорифм осуществляет это, занимая под величины потоков соответствующие звеньям базиса строки списка A.

$$p_\kappa\Rightarrow b_\kappa\Big\}\ \kappa=1,2,\ldots,r,$$

если $[\pi_s=0]$, то $[-b_{i_s}\Rightarrow a_s,\ b_{i_s}+b_{j_s}\Rightarrow b_{j_s}]$

если $[\pi_s=1]$, то $[b_{j_s}\Rightarrow a_s,\ b_{i_s}+b_{j_s}\Rightarrow b_{i_s}]$ } $s=r-1,r-2,\ldots,1.$

В заключение сделаем некоторые замечания, касающиеся реализации описанного алгорифма на ЭВМ.

З а м е ч а н и е I. При реализации на ЭВМ проверку условий оптимальности целесообразно делать независимым циклом, т. е. после очередного изменения базиса и вычисления новых потенциалов начинать просмотр полного списка звеньев (I) не с начала, а с того места, где на прошлом шаге было обнаружено нарушение условий (3). При этом вводить в базис следует лишь звено, на котором невязка в условиях (3) превышает некоторое число (так называемый "барьер"), которое периодически дробится в процессе вычислений. Впрочем, это замечание, по-видимому, относится ко многим алгорифмам линейного программирования.

З а м е ч а н и е 2. Может показаться, что в алгорифме совершается ненужная работа по вычислению потоков на каждом шаге, которой можно было бы избежать, сохраняя эти потоки. Однако при хранении потоков, помимо занимаемого ими места, мы прогадывали бы еще на их исправлении и перестановке при изменении базиса.

З а м е ч а н и е 3. При наличии свободного места в памяти машины можно было бы потенциалы хранить на отдельном поле. При этом перевычисление их на каждом шаге следовало бы начинать со строки списка A с номером γ, т. е. перевычислять потенциалы лишь по той части базиса, которой коснулась перестройка. Поскольку, однако, потенциалы строятся за один просмотр, выигрыш от этого был бы не слишком большим.

З а м е ч а н и е 4. Почти все приемы построения начального плана транспортной задачи легко приспосабливаются для получения базиса с нужным здесь упорядочением. Поэтому этот вопрос при описании алгорифма не рассматривался.

Л И Т Е Р А Т У Р А

I. Б р у д н о А. Л. Решение транспортной задачи методом вычеркивающей нумерации. В кн. "Применение цифровых вычислительных машин в экономике". М., 1962, стр. 17-38.

2. К а н т о р о в и ч Л. В. и Г а в у р и н М. К. Применение математических методов в вопросах анализа грузопотоков. В кн.

"Проблемы повышения эффективности работы транспорта", М., 1949, стр. 110-138.

3. Л у р ь е А. Л. Алгоритм решения транспортной задачи путем приближения условно-оптимальными планами. - Вычисл. математика, 1961, № 7, стр. 151-160.

4. Я к о в л е в а М. А. Задача о минимуме транспортных затрат. В кн. "Применение математики в экономических исследованиях", т. I. М., 1959, стр. 390-399.