

УДК 512.25+519.1

А.А.АНИКЕИЧ, А.Б.ГРИБОВ

О РЕАЛИЗАЦИИ ЭФФЕКТИВНОГО АЛГОРИТМА РЕШЕНИЯ ТРАНСПОРТНОЙ  
ЗАДАЧИ ЛИНЕЙНОГО ПРОГРАММИРОВАНИЯ НА ЭВМ

В статье рассматривается способ реализации алгоритма решения транспортной задачи линейного программирования [1], предложенного А.Л.Брудно [2] (аналогичная организация вычислений в методе потенциалов предлагалась В.А.Булавским [3]). Программе на АЛГОЛ-60 предшествует описание алгоритма в терминах, удобных в дальнейшем изложении.

Задача состоит в следующем. Найти

$$x_{ij} \quad (i=1, \dots, m, j=1, \dots, n) ,$$

удовлетворяющие условиям

$$x_{ij} \geq 0 , \quad (1)$$

$$\sum_{i=1}^m x_{ij} = b_{m \cdot j} , \quad (2)$$

$$\sum_{j=1}^n x_{ij} = b_i \quad (3)$$

и доставляющие минимум функционалу

$$\sum_{i,j} c_{ij} x_{ij} .$$

При этом предполагается, что  $S = P$ , где

$$S = \sum_{j=1}^n b_{m \cdot j} , \quad P = \sum_{i=1}^m b_i . \quad (4)$$

Рассмотрим новую задачу, отличающуюся указанной тем ,

что вместо матрицы  $\{C_{ij}\}$  фигурирует матрица  $\{\bar{C}_{ij}\}$ , где

$$\bar{C}_{ij} = C_{ij} \quad \text{для} \quad i \neq s \quad \text{и} \quad C_{sj} = C_{sj} + u_s,$$

т.е. изменена одна строка добавлением некоторого числа ко всех элементам.

Известно, что обе задачи эквивалентны в том смысле, что оптимальный план одной будет оптимальным и во второй, а функционалы отличаются на слагаемое  $u_s \theta_s$ .

В рассматриваемом алгоритме это преобразование будет повторяться несколько раз.

В принятых нами обозначениях коэффициенты вектора  $u$  — ренты,  $q_k$  — координаты ( $i$  и  $j$ ) поставки  $x_{ij}$  в виде  $1000 \cdot i + j$  ( $j < 1000$ ),  $x_k$  — объем поставки ( $k = \overline{1, \dots, m \cdot n}$ ).

В процедуре предполагается, что  $\theta_k$  — целые. Сначала в каждом столбце  $j$  выбирается самая выгодная строка  $i(j)$ , т.е. отыскивается  $i$ , при котором достигается  $\min C_{ij}$ . Одновременно подсчитываются суммы условия (4). При невыполнении этого условия происходит выход к внешней метке.

По полученной системе координат происходит распределение поставок:

$$x_k = \min [a_{i_k}^{(\kappa)}, a_{j_k}^{(\kappa)}], \quad \text{где}$$

вектор  $a^{(\kappa)}$  представляет собой еще не реализованные мощности поставщиков и еще не удовлетворенные потребности потребителей перед реализацией  $K$ -ой поставки. Это происходит следующим образом:  $i_k$  и  $j_k$  вычисляются по  $q_k = 1000 \cdot i_k + j_k$ , а коэффициенты вектора  $a^{(\kappa+1)}$  пересчитываются по формулам:

$$a_{i_k}^{(\kappa+1)} = a_{i_k}^{(\kappa)} - x_k,$$

$$a_{j_k}^{(\kappa+1)} = a_{j_k}^{(\kappa)} - x_k,$$

$$a_s^{(\kappa+1)} = a_s^{(\kappa)} \quad (s \neq i_k, s \neq j_k).$$

При этом вычисления производятся от  $K = \overline{1, \dots, m \cdot n}$ , и для начального вектора  $a^{(1)}$  принимается  $a_s^{(1)} = \theta_s$ .

На каждой итерации граф поставок таков, что каждая его компонента связности есть дерево и распределение по этому дереву (определение  $q_k$  и  $x_k$ ) происходит так, чтобы получить максимальную суммарную поставку. То, что распределено, т.е.  $\sum_{k=1}^{m \cdot n} x_k$  — распределено оптимально. Если распределено

се ( $S=p$ ), то получен оптимальный план.

В противном случае строится новый граф поставок и по нему новая система координат. Для этого все множество строк  $J$  разбивается на два множества  $J_1$  и  $J_2 = J \setminus J_1$ .  $J_1$  включает строку  $i$ , если увеличение  $b_i$  влечет увеличение максимальной суммарной поставки (это относится к имеющемуся графу). Строка  $i$  в этом случае называется недостаточным поставщиком, в противном случае - избыточным. Соответственно называются и поставки в этих строках.

Из множества  $J$  всех столбцов выделим множество  $J_1$  столбцов, в которых все поставки недостаточны.

Такое разбиение достигается следующим путем. Относим к  $J_1$  все столбцы  $j$ , для которых  $a_{m+j}^{m+n} > 0$ . По имеющейся системе координат просматриваем с конца все поставки и выполняем:

- 1) если  $j_k \in J_1$ , то  $i_k$  отнесем к  $J_1$ ;
- 2) если  $i_k \in J_1$  и  $x_k > 0$ , то  $j_k$  отнесем к  $J_1$ .

После этого новый граф получается из старого удалением всех поставок, для которых  $i \in J_1$ , а  $j \in J_2 = J \setminus J_1$ , и добавлением новой поставки, координаты которой есть координаты элемента матрицы  $C$ , при котором достигается

$$\Delta = \min_{J_1} (\min_{J_2} C_{ij} - \min_J C_{ij}). \quad (5)$$

Идея состоит в том, что у недостаточных поставщиков (у них не хватает продукта) увеличиваются цены, пока не появится новая возможность поставки.  $\Delta$  и есть новая рента, которая добавляется ко всем недостаточным строкам.

В формуле (5) имеются в виду  $C_{ij}$ , полученные на предыдущей итерации. В программе же считается, что ренты хранятся отдельно.

$u_i$  есть суммарная рента поставщика  $i$ , а  $\Delta$  и координаты новой поставки вычисляются по формуле:

$$\Delta = \min_{J_1} (\min_{J_2} (C_{ij} + u_i) - \min_J (C_{ij} + u_i)). \quad (5')$$

Можно показать, что  $J_1$  и  $J_2$  не пусты и что компоненты связности нового графа есть деревья, если деревом был предыдущий граф.

Система координат по данному графу получается выбором вершины с одним инцидентным с ней ребром (в дереве такие существуют) и присвоением ей номера  $\Gamma$  ( $\Gamma = m+n-p$ , где  $p$  -

число ребер в графе). Занумерованное ребро выбрасываем, а следующий номер присваиваем следующему найденному ребру в оставшемся графе (в ней все компоненты связности — опять деревья).

В программе одновременно с формированием графа считается число ребер, инцидентных каждой вершине.

В дальнейшем новая система координат получается из старой путем сдвига поставок (за счет выброшенных). Вновь введенной поставке присваивается минимальный возможный номер. Затем просматриваются оставшиеся поставки и очередной поставке присваивается минимально возможный номер, начиная с  $\Gamma$ , если хотя бы одна из вершин ребра этой поставки не имеет других инцидентных ребер. В противном случае этой поставке присваивается наибольший возможный номер, начиная с  $m + n$ .

После этого по полученной системе координат вновь производится распределение поставок и т.д., пока все не распределится.

Процесс не закичивается, поскольку суммарная поставка от итерации к итерации не убывает. Если же она остается одной и той же, то это влечет за собой увеличение числа недостаточных поставщиков.

Приведем теперь запись алгоритма на языке АЛГОЛ-60 [4].

```
procedure transport (b,c,x,q,u,m,n,s,p,z);
  value m,n; real z; integer m,n,s,p;
  array c,u; integer array b,x,q;
  comment c - матрица расстояний, b - вектор, первые
  m компонент которого показывают мощность поставщиков, а последние n компонент - потребности потребителей, x - вектор поставок, q - вектор координат i и j этих поставок, u - вектор рент, z - значение функционала, s - суммарный объем потребности потребителей, p - суммарная мощность производителей;
```

```
  begin integer array a[I:m+n];
    integer i ,i1,i2,j,j1,k,r; real m1,m2,m3,e;
    comment Здесь e - большое относительно элементов матрицы c число. Обычно вполне достаточно взять e равным  $10^6$ ;
```

```
    e:= $10^{12}$ ;
    r:=m+1; s:=p:=0;
    for i:=1 step 1 until m do
      begin p:=p+b[i]; u[i]:=q[i]:=0 end;
    for j:=1 step 1 until n do
      begin s:=s+b[m+j]; m1:=e;
```

```

    for i:=I step I until m do if mI>c[i,j] then
        begin q[j+m]:=I000*i+j; mI:=c[i,j] end
    end;
    if s≠p then go to fin 1;

```

comment Здесь *fin 1* и ниже *fin 2* - нелокализованные в процедуре метки. Выход *fin 1* означает, что не соблюдено условие (4), а *fin 2* - получен оптимальный план;

```

1: for k:=I step I until m+n do a[k]:=b[k]; s:=z:=0;
    for k:=r step I until m+n do
        begin i:=q[k]÷I000; j:=q[k]-I000*i+m;
            x[k]:=if a[i]<a[j] then a[i] else a[j];
            a[i]:=a[i]-x[k]; a[j]:=a[j]-x[k];
            z:=z+x[k]*c[i,j-m]; s:=s+x[k]
        end;

```

```

    if s=p then go to fin 2;

```

comment Получен план, состоящий из  $m+n-\Gamma+1$  компоненты. Число  $\Gamma$  может меняться при переходе от одной итерации к другой. Если при этом выполняется условие  $s=p$ , то имеющийся план оптимален;

```

    for i:=I step I until m do a[i]:=0;
    for k:=m+n step -I until r do
        begin i:=q[k]÷I000; j:=q[k]-I000*i+m;
            if a[j]>0 then a[i]:=I;
            if a[i]=I ∧ x[k]>0 then a[j]:=I
        end;
    mI:=e;
    for j:=I step I until n do if a[j+m]>0 then
        begin m2:=m3:=e;
            for i:=I step I until m do
                begin if m3>c[i,j]+u[i] then m3:=c[i,j]+u[i];
                    if a[i]=0 then
                        begin if m2>c[i,j]+u[i] then
                            begin m2:=c[i,j]+u[i]; i2:=i end
                        end
                    end;
                end;
            z:=m2-m3;
            if mI>z then begin mI:=z; jI:=j; iI:=i2 end

```

```

    end;
    for i:=I step I until m do
        if a[i]>0 then u[i]:=u[i]+mI;
    for k:=I step I until m+n do
        begin x[k]:=q[k]; q[k]:=0 end;
    s:=m+n;
    for k:=s step -I until r do
        begin i:=x[k]÷I000;
            j:=x[k]-I000×i+m;
            if ¬(a[i]>0 ∧ a[j]=0) then
                begin x[s]:=x[k]; s:=s-I;
                    q[i]:=q[i]+I; q[j]:=q[j]+I
                end
            end;
        end;
    x[s]:=I000×iI+jI;
    q[iI]:=q[iI]+I;
    q[jI+m]:=q[jI+m]+I; r:=s; iI:=m+L;
    for k:=I step I until iI do a[k]:=q[k];
    for k:=r step I until m+n do
        begin i:=x[k]÷I000;
            j:=x[k]-I000×i+m;
            if a[i]=I ∨ a[j]=I
                then begin a[i]:=a[i]-I;
                    a[j]:=a[j]-I;
                    q[s]:=x[k];
                    s:=s+I
                end
            end
            else begin q[iI]:=x[k]; iI:=iI-I and
        end;
    end;
    go to 1
end

```

Вышеприведенная процедура отлажена на М-20 с помощью транслятора ТА-1. Решение задач ведется по рабочей программе, полученной на АЛФА-трансляторе.

Решение задачи размера 17 x 34 заняло 25 сек. машинного времени, задачи размера 10 x 90 - 50 сек.

В заключение авторы выражают благодарность редакции сборника "Алгоритмы и алгоритмические языки" за ряд замечаний, способствовавших улучшению данной работы.

### Л и т е р а т у р а

1. Л.В. Канторович и М.К. Гавурин. Применение математических методов в вопросах анализа грузопотоков. "Проблемы повышения эффективности работы транспорта", М., Изд. АН СССР, 1949, 110-138.
2. А.Л. Брудно. Решение транспортной задачи методом вычеркивающей нумерации. "Применение цифровых вычислительных машин в экономике. Транспортная задача линейного программирования". М., Изд. АН СССР, 1962, 17-38.
3. В.А. Булавский. Об одном алгоритме решения транспортной задачи - "Оптимальное планирование", вып. 2, Новосибирск, 1964, 41-49.
4. Алгоритмический язык АЛГОЛ-60. Пересмотренное сообщение. Перевод с английского под ред. А.П. Ершова, С.С. Лаврова и М.Р. Шура-Бура, М., 1965.

Поступила в редакцию  
15.XI.1967 г.