

УДК 519. 95

В.М. ВОЛКОВ

**ПРОГРАММА ДЛЯ ПРИБЛИЖЕННОГО РЕШЕНИЯ ЧАСТНОЙ
ЗАДАЧИ ЛИНЕЙНОГО ПРОГРАММИРОВАНИЯ С БУЛЕВЫМИ
ПЕРЕМЕННЫМИ**

Особенность рассматриваемой здесь целочисленной линейной задачи состоит в том, что входящие в нее булевы переменные разбиты на непересекающиеся группы и для каждой группы требуется, чтобы сумма переменных этой группы была равна единице. Подобные модели достаточно универсальны, но наиболее естественным образом они возникают при исследовании отраслевого планирования размещения и специализации производства — одной из важных областей применения математико-экономических методов.

Каждому допустимому целочисленному решению рассматриваемой задачи можно поставить в соответствие некоторую вспомогательную, тоже целочисленную задачу, любое допустимое решение которой позволяет улучшить соответствующее решение основной задачи. Для нахождения допустимых решений вспомогательной задачи используется алгоритм, основанный на частичном переборе.

1. Постановка и анализ задачи.

Рассматриваемая задача может быть сформулирована следующим образом.

Задача 1:

$$\text{Минимизировать} \quad Z = \sum_{j=1}^n c_j x_j \quad (1)$$

при условиях:

$$\sum_{j=1}^n a_{ij} x_j \geq b_i, \quad i=1, \dots, m, \quad (2)$$

$$\sum_{j \in J_t} x_j = 1, \quad t=1, \dots, \ell, \quad (3)$$

$$x_j = 0 \quad \text{или} \quad 1, \quad j=1, \dots, n. \quad (4)$$

Множества J_t в условиях (3) определяются соотношениями:

$$J_t = \left\{ j: \sum_{k=0}^{t-1} n_k <_j \leq \sum_{k=0}^t n_k \right\}, \quad (5)$$

где n_0, n_1, \dots, n_ℓ — заданные числа, причем $n_0 = 0$. Числа c_j, a_{ij}, b_i в условиях (1) и (2) будем предполагать целыми.

Вектор $X = (x_1, \dots, x_n)$ размерности n называется допустимым вектором для задачи I, если его компоненты удовлетворяют условиям (2) — (4).

Пусть $X^0 = (x_1^0, \dots, x_n^0)$ — какое-либо допустимое решение задачи I. Предположим, что в выбранном допустимом решении в группе t переменная x_{j_t} равна 1, а остальные переменные этой группы равны 0. Для решения X^0 вычислим величины z^0 и $\Delta b_i^0, i=1, \dots, m$, по формулам:

$$z^0 = \sum_{t=1}^{\ell} c_{j_t} \quad (6)$$

$$\Delta b_i^0 = \sum_{t=1}^{\ell} a_{ij_t} - b_i, \quad i=1, \dots, m \quad (7)$$

Переменные $x_{j_t}, t=1, \dots, \ell$, вместе с переменными $\Delta b_i, i=1, \dots, m$, определяют базис в $(m + \ell)$ -мерном пространстве, соответствующий допустимому решению X^0 . Эти переменные мы будем называть базисными, а переменные $x_j, j \neq j_t, t=1, \dots, \ell$ — небазисными переменными.

Ограничения (3) задачи I позволяют легко найти представление функции z и базисных переменных как функций от небазисных переменных. Для этого достаточно ограничения (3) разрешить относительно базисных переменных x_{j_t} (в каждое такое ограничение входит только одна базисная переменная) и результат подставить в (1) и (2). Таким образом, решению X^0 будут соответствовать следующие разложения функции z и переменных Δb_i по небазисным переменным:

$$Z = Z^* + \sum_{t=1}^{\ell} \sum_{j \in J_t} (c_{jt} - c_j)(-x_j), \quad (8)$$

$$\Delta b_i = \Delta b_i^* + \sum_{t=1}^{\ell} \sum_{j \in J_t} (a_{ijt} - a_{ij})(-x_j), \quad i=1, \dots, m. \quad (9)$$

Для дальнейшего нам удобно предположить, что в формулы (8) и (9) входят также и базисные переменные — с нулевыми коэффициентами.

Формулы (8) и (9) представляют собой сокращенную симплексную таблицу, соответствующую допустимому решению X^* . Эта таблица не содержит разложений базисных переменных x_{j_t} , $t=1, \dots, \ell$.

Обозначим коэффициенты симплексной таблицы, соответствующей некоторому решению X , через s_{ij} , $i=0, 1, \dots, m$, $j=0, 1, \dots, n$, а саму таблицу — через S .

Допустимому решению X^* будут соответствовать обозначения s_{ij}^* и S^* . Все s_{ij} , очевидно, целые.

Сформулируем теперь вспомогательную задачу, соответствующую решению X^* задачи 1.

Задача 2. Найти неотрицательный вектор $Y = (y_1, \dots, y_n)$ из условий:

$$\sum_{j=1}^n s_{0j}^* y_j \geq 1; \quad (10)$$

$$\sum_{j=1}^n s_{ij}^* y_j \leq s_{i0}^*, \quad i=1, \dots, m; \quad (11)$$

$$\sum_{j \in J_t} y_j \leq 1, \quad t=1, \dots, \ell; \quad (12)$$

$$y_j = 0 \quad \text{или} \quad 1. \quad (13)$$

Легко проверить справедливость следующего утверждения (см [1]): для того, чтобы некоторое допустимое решение задачи 1 было оптимальным, необходимо и достаточно, чтобы соответствующая этому решению вспомогательная задача 2 не имела допустимых решений.

Каждое допустимое решение вспомогательной задачи 2 позволяет улучшить соответствующее (допустимое решение задачи 1.

Любое решение задачи 2 представляют собой неотрицательную линейную комбинацию столбцов таблицы S^* (кроме нулевого столбца). Коэффициенты этой линейной комбинации могут при-

иметь лишь два значения 0 и 1, и, кроме того, $\sum_{j \in J_1} y_j \leq 1$ (условие (I2)).

Рассмотрим простейший случай, когда решением задачи 2 является какой-либо один столбец таблицы S^* . Тогда вектор Y имеет одну отличную от нуля компоненту y_{j_0} : $y_{j_0} = 1$, $j_0 \in J_{1*}$. Для того, чтобы получить новое решение $X' = (x'_1, \dots, x'_n)$, помножим

$$\begin{aligned} x'_{j_0} &= x_{j_0} = 1, \\ x'_j &= x_j, \quad \text{если } j \notin J_{1*}. \end{aligned} \quad (I4)$$

Таблица S' , соответствующая решению X' , получается из таблицы S^* по формулам:

$$\begin{aligned} s'_{ij} &= s^*_{ij} - s^*_{ij_0}, \quad j = 0, \quad j_0 \in J_{1*}, \quad i = 0, 1, \dots, m; \\ s'_{ij} &= s^*_{ij}, \quad j \neq 0, \quad j \notin J_{1*}, \quad i = 0, 1, \dots, m. \end{aligned} \quad (I5)$$

Новое решение X' является допустимым решением — это следует из соотношений (II). Значение функции Z для X' меньше соответствующего значения для решения X^* , так как

$$Z' = s'_{10} = s^*_{10} - s^*_{1j_0} = Z^* - s^*_{1j_0} < Z^*.$$

В случае, когда решение задачи 2 содержит более чем одну ненулевую компоненту, процесс получения нового решения и соответствующей ему таблицы аналогичен описанному выше: он состоит в последовательном выполнении преобразований (I4) и (I5) для каждой ненулевой компоненты вектора Y .

2. Алгоритм. Для начала работы алгоритма необходимо иметь некоторое допустимое решение X^* . В процессе работы алгоритма выявляется возможность улучшения исходного решения. Если решение X^* можно улучшить, то вычисляется новое решение X' . В противном случае алгоритм работу заканчивает. Решение X' получается аналогичным образом.

Пусть X^z — допустимое решение задачи I, полученное на шаге z , а S^z — соответствующая ему симплексная таблица. Обозначим через D^z множество допустимых решений задачи 2, соответствующей X^z , а через D^z_g — множество целочисленных линейных комбинаций g столбцов таблицы S^z , $g \leq \ell$, с коэффициентами, удовлетворяющими условию (I2). Если для решения X^z

$$\left(\bigcup_{g=1}^{\ell} D^z_g \right) \cap D^z = \emptyset, \quad \lambda \leq \ell,$$

то решение X^z может быть улучшено, т.е. можно получить решение X^{z+1} с меньшим, чем у X^z , значением функции z . Переход от X^z к X^{z+1} и от S^z к S^{z+1} осуществляется с использованием формул (14) и (15) так, как было описано в предыдущем пункте. Вычисления продолжают до тех пор, пока для некоторого решения X^z не окажется

$$\left(\bigcup_{j=1}^{\ell} D_j^z \right) \cap D^z = \Lambda.$$

Решение X^z принимается за окончательное приближенное решение исходной задачи.

Для реализации на ЭВМ изложенного алгоритма была составлена α -схема, которая приводится ниже. Программа составлена таким образом, что λ - максимальное число ненулевых компонент в проверяемых решениях задачи 2 - является одним из параметров программы, который может изменяться в пределах от 1 до ℓ . При $\lambda = \ell$ программа превращается в программу полного перебора, а при $\lambda = 1$ улучшение исходного решения осуществляется только лишь за счет наилучшего выбора, проводимого независимо в каждой группе переменных. На практике значение λ определяется в зависимости от размера задачи и числа групп переменных таким образом, чтобы объем частичного перебора был бы не слишком велик, т.е. чтобы его можно было осуществить в приемлемое время. Время счета при фиксированном λ пропорционально величине $C_n^{\lambda} \bar{n}^{\lambda}$, где \bar{n} - среднее значение n_t , $t=1, \dots, \ell$. Как показали эксперименты, во многих случаях для получения хорошего приближения к оптимуму достаточно было взять $\lambda = 3$. Для этого случая составлена отдельная программа.

3. Программы. Программа для случая $\lambda = 3$.

begin integer m,n,l,i,j,k,r,g,q,t;

ввод (m,n,l);

begin integer array A[I:m-1,1:n], o[1:m-1], X[1:n],

P,Q[0:1]; integer a1,a2,a3,a4,a8,a9,a10;

comment В программе приняты несколько иные обозначения по сравнению с теми, которые использовались при описании алгоритма. Полная размерность матрицы задачи I равна $m \times n$, где m - число всех ограничений, считая и функцию (I), n - число переменных задачи. Число непересекающихся групп переменных равно ℓ . В машину должны быть введены массивы: A - матрица коэффициентов ограничений (2), причем её первая строка

составлена из коэффициентов функции (1), вектор \bar{b} , у которого $\bar{b}[1] = 0$, а остальные компоненты равны правым частям ограничений (2), имеющееся допустимое решение X и вектор P , для которого $P[0] = m - 1 - \ell =$ числу ограничений (2) и $P[i] = n_i, i = 1, \dots, \ell$;

procedure simplex (a2,a3);

begin integer iI,jI,kI; for iI:=1 step 1 until m-1

do b[iI]:=b[iI]-A[iI,a3]; for kI:=1 step 1 until m-1

do {P[iI]:=A[iI,a3]; for jI:=Q[a2]+1 step 1 until Q[a2+1]

do A[kI,jI]:=A[kI,jI]-P[iI]};

for jI:=Q[a2]+1 step 1 until Q[a2+1] do X[jI]:=0;

X[a3]:=1;

end;

comment Процедура simplex (a2,a3) осуществляет преобразование симплексной таблицы и решения X , соответствующее одной положительной компоненте найденного допустимого решения вспомогательной задачи. Параметры α_2 и α_3 равны соответственно номеру этой компоненты и номеру группы (без единицы), в которую она входит;

procedure plan;

begin

aI:=0; for j:=1 step 1 until n do if X[j] > 0 then

{P[aI+1]:=j; aI:=aI+1};

вывод (a8,a9,a10,b,P);

end

comment Процедура plan вычисляет и печатает номера ненулевых компонент решения X , полученного после просмотра каждой из областей D_1, D_2, D_3 . Число допустимых решений вспомогательной задачи, найденное в каждой из этих областей равно соответственно $\alpha_8, \alpha_9, \alpha_{10}$. Печатается также вектор \bar{b} , для которого $\bar{b}[1] = 2, \bar{b}[i+1] = -4 \bar{b}_i$;

ввод (A,b,X,P);

$V_I : Q[0] := 0; \text{ for } t := 1 \text{ step } 1 \text{ until } I \text{ do}$

$Q[t] := Q[t-1] + P[t];$

```

for i:=1 step 1 until m-1 do
{a3:=a3+1; for j:=1 step 1 until n do if X[j]=1 then
a3:=a3+A[i,j]; b[i]:=a3-b[i]};
for t:=1 step 1 until m-1 do
for i:=0 step 1 until 1-I do
for j:=Q[i]+1 step 1 until Q[i+1] do if X[j]=1 then
{a1:=A[t,j]; for r:=Q[i]+1 step 1 until Q[i+1] do
A[t,r]:=a1-A[t,r]}; a8:=a9:=a10:=0;

```

comment Оператор V_1 преобразует матрицу ограничений (2) исходной задачи в сокращенную симплексную таблицу, соответствующую допустимому решению X ;

```

V2: for k:=0 step 1 until 1-1 do
for j:=Q[k]+1 step 1 until Q[k+1] do if A[I,j] ≥ I
then {r:=0; for i:=2 step 1 until n-1 do
if A[i,j] ≤ b[i] then r:=r-I;
if r=P[0] then {a8:=a8+I; simplex(k,j)}};
plan;

```

comment Оператор V_2 просматривает область D_1 . Следующие операторы V_3 и V_4 просматривают области D_2 и D_3 соответственно;

```

V3: for k:=0 step 1 until 1-2 do
for j:=Q[k]+1 step 1 until Q[k+1] do
for t:=k+1 step 1 until 1-I do
for r:=Q[t]+1 step 1 until Q[t+1] do
if (X[r]≠1) ^ (A[I,j]+A[I,r] ≥ I) then
{a4:=0; for i:=2 step 1 until m-1 do
if b[i]-A[i,j]-A[i,r] ≥ 0 then a4:=a4+I;
if a4=P[0] then
{a9:=a9+I; simplex(k,j); simplex(t,r)}};
plan;

```

```

V4: for k:=0 step 1 until 1-3 do
for j:=Q[k]+1 step 1 until
for t:=k+1 step 1 until 1-2 do

```

```

for r:=Q[t]+1 step 1 until Q[t+1] do
for i:=t+1 step 1 until 1-I do
for q:=Q[i]+1 step 1 until Q[i+1] do
if A[I,j]+A[I,r]+A[I,q] ≥ I then
  {a2:=0; for g:=2 step 1 until m-1 do
if b[g]-A[g,j]-A[g,r]-A[g,q] ≥ 0 then a2:=a2+I;
if a2:=P[0] then
  {aIO:=aIO+I; simplex(k,j); simplex(t,r); simplex(i,q);
  plan}} ;

```

V₅: вывод (a8,a9,aIO,b,X,A);

end end *

Программа для случая произвольного λ .

```

begin integer m,n,l, $\lambda$ ,i,j,k,t;
ввод (m,n,l, $\lambda$ );
begin integer array A[I:m-1,I:n],b[I:m-1],X[I:n],
P,Q[0:1],R[I:1]; integer aI,a2,a3,a4,a5,a6;
procedure simplex(a3,a4);

```

comment эта процедура совпадает с точностью до обозначений её аргументов с procedure simplex(a2,a3), описанной в предыдущей программе;

ввод (A,b,X,P);

comment параметры исходных данных m,n,l и вводимые в память машины массивы A,b,X и P те же, что и в предыдущей программе. Дополнительно вводится параметр λ , характеризующий размер частичного перебора, осуществляемого для нахождения допустимых решений вспомогательной задачи;

```

V1: Q[0]:=0; for k:=1 step 1 until 1 do Q[k]:=Q[k-1]+P[k];
for i:=1 step 1 until m-1 do
  {a3:=0; for j:=1 step 1 until n do
if X[j]=1 then a3:=a3+A[i,j]; b[i]:=a3-b[i]} ;
for t:=1 step 1 until m-1 do
  {for i:=0 step 1 until 1-I do
    {for j:=Q[i]+1 step 1 until Q[i+1] do {if X[j]=1 then

```



```
{ a6:=A[t,j]; for k:=Q[i]+1 step I until Q[i+1] do A[t,k]:=
  = a6-A[t,k] ) } ;
```

```
aI:=a2:=a5:=0; for k:=I step I until l do
```

```
R[k]:=if k ≤ λ then I else 0;
```

```
go to V4;
```

comment оператор V_1 , как и в предыдущей программе, преобразует матрицу ограничений (2) исходной задачи в сокращенную симплексную таблицу, соответствующую допустимому решению X . Далее рассматриваются возможные комбинации λ столбцов полученной симплексной таблицы. Последовательность просматриваемых комбинаций определяется последовательностью векторов R . Первым вектором в этой последовательности является вектор $R = (\underbrace{1, 1, \dots, 1}_\lambda, 0, 0, \dots, 0)$, а последним $R = (0, \dots, 0, \underbrace{P[l-\lambda+1], \dots, P[l]}_{l-\lambda})$. Правила построения этой последовательности (операторы V_2 и V_3) обеспечивают просмотр всех возможных комбинаций λ столбцов симплексной таблицы. А так как в симплексную таблицу включены столбцы, соответствующие базисным переменным (эти столбцы состоят из нулей), то будут просмотрены и все комбинации с числом столбцов, меньшим чем λ .

```
V2: if aI < l - λ then { aI:=aI+I; for i:=I step I until l
```

```
do R[i]:=if (i > aI)^(i ≤ aI+λ) then I else 0;
```

```
a2:=0; go to V4 } ; go to V5;
```

```
V3: for i:=aI+I step I until l do if (R[i] > 0)^(R[i] < P[i])
then
```

```
{ R[i]:=R[i]+I; for j:=aI+I step I until i-I do
```

```
if R[j] > 0 then R[j]:=I; go to V4 } ;
```

```
if a2 < l - λ - aI then
```

```
{ a2:=a2+I; for i:=aI+I step I until aI+λ-I do
```

```
R[i]:=I; R[aI+λ-I+a2]:=0; R[aI+λ+a2]:=I; go to V4
```

```
else go to V2;
```

```
V4: a3:=0; for j:=aI+I step I until l do
```

```
if R[j] > 0 then a3:=a3+A[I, Q[j-I]+R[j]];
```

```
if a3 < I then go to V3;
```

```
for i:=2 step I until m-1 do
```

```
{ a4:=0; for j:=aI+I step I until l do
```

```
if R[j] > 0 then a4:=a4+A[1, Q[j-I]+R[j]];
```

```

if  $a_4 > b[1]$  then go to  $v_3$  ;
 $a_5 := a_5 + 1$ ; for  $k := a_1 + 1$  step 1 until 1 do
if  $R[k] > 0$  then simplex( $k-1, Q[k-1] + R[k]$ );
go to  $v_3$ ;

```

comment оператор v_4 исследует допустимость рассматриваемой комбинации;

```

 $v_5$ :  $a_6 := 0$ ; for  $j := 1$  step 1 until  $n$  do if  $x[j] = 1$  then
{  $P[a_6 + 1] := j$ ;  $a_6 := a_6 + 1$  } ;
вывод ( $a_1, a_2, a_3, b, P, R$ )
end end *

```

Л и т е р а т у р а

1. A. Ben - Israel, A. Charnes. On some problems of diophantine programming. Cahiers du centre d'etudes de recherche operationnelle. Bruxells, 1962, 4, N 4.
2. Ю.И. Волков. О некоторых упрощенных приемах анализа и решения целочисленных задач линейного программирования. Доклады на Всесоюзной конференции по отраслевому планированию. Новосибирск, 1966.

Поступила в редакцию
5 апреля 1969 г.