УДК 519.8

ТРИ ПРОЦЕДУРЫ СИМПЛЕКС-МЕТОДА

В.И.Болдырев, В.И.Хохлюк

Известно, что для решения задачи линейного программирования (LP) часто используется симплекс-метод. Уже существует ряд программ, реализующих этот метод с учетом специфики машины и специальной структуры задачи.

В данной работе для решения общей задачи LP предлагаются три процедуры, которые реализуют модифицированный симплекс-метод и отличаются друг от друга способом задания и обработки числового материала. Основной целью этой работы является не максимальное повышение размерности решаемых задач (для этого существуют другие средства), а стремление дать максимально удобные при многократном использовании процедуры, экономно и надежно работающие как части других программ.

В §1 дано краткое математическое описание алгоритма, который реализован в виде трех процедур. В §2 приводится процедура SIMP1, в которой исходные данные и текущая обратная матрица записаны в полной форме. В §3 предлагается процедура SIMP2, в которой исходные данные записаны в краткой форме, а текущая обратная матрица - в полной форме. В §4 дана процедура SIMP 3 , в которой исходные данные и текущая обратная матрица записаны в краткой форме.

Отметим несколько особенностей предлагаемых процедур.

1. Процедуры записаны на алгоритмическом языке АЛГОЛ-60.

2. Исходные данные и текущая обратная матрица могут быть записаны как в полной, так и в краткой форме (без нулей).

3. Анализируются параметры задачи m, n, где m - число ограничений (равенств, неравенств, исключая неравенства вида $x_j \geq 0$, $j=1,\ldots,n$), n - число неотрицательных переменных. В результате анализа параметров m и n хранится и обрабатывается квадратная матрица порядка min(m,n).

4. Экономичный учет вырожденности полностью исключает возможность зацикливания вычислительного процесса.

5. Решение задачи в два этапа исключает трудно определяемый параметр, называемый "достаточно большим числом".

Отладка предложенных АЛГОЛ - программ и решение тестовых задач были проведены на ЭВМ БЭСМ-6 с использованием системы автоматизации программирования Алгибр и БЭСМ - АЛГОЛ [2,3].

§I. Описание алгоритма

I. Постановка задачи и общий ход её решения. Общая задача ЛП может быть сформулирована в следующем виде.

Найти минимум линейной функции $\sum_{j=1}^{n} c_j x_j$ при условиях:

$$\begin{aligned}
&\sum_{j=1}^{n} a_{ij} x_j \le b_i \qquad (i \in I_1),\\
&\sum_{j=1}^{n} a_{ij} x_j = b_i \qquad (i \in I_2),\\
&\sum_{j=1}^{n} a_{ij} x_j \ge b_i \qquad (i \in I_3),\\
&x_j \ge 0 \qquad (j=1,\dots,n).
\end{aligned} \tag{I}$$

Здесь использованы обозначения: m - число ограничений (исключая неравенства вида $x_j \ge 0$); n - число неотрицательных переменных; a_{ij}, b_i, c_j $(i=1,\dots,m;\ j=1,\dots,n)$ - заданные вещественные числа; I_1, I_2, I_3 - непересекающиеся подмножества множества $I=\{1,\dots,m\}$, для которых $I_1 \cup I_2 \cup I_3 = I$. Матрицу (a_{ij}) порядка $m \times n$ в дальнейшем будем называть матрицей ограничений задачи (I).

Заменяя систему уравнений

$$\sum_{j=1}^{n} a_{ij} x_j = b_i \qquad (i \in I_2)$$

эквивалентной ей системой неравенств

$$\sum_{j=1}^{n} a_{ij} x_j \ge b_i \qquad (i \in I_2),$$

$$\sum_{i \in I_2} \left(\sum_{j=1}^{n} a_{ij} x_j - b_i \right) \le 0$$

и умножая каждое неравенство вида $\le$ в задаче (I) на -1, получаем $m+1$ неравенств вида $\ge$ (исключая неравенства $x_j \ge 0$).

Используя матричные обозначения, сформулируем следующую пару двойственных задач.

Прямая задача (I): $\min\{ cx \mid A' x \ge b', x \ge 0 \}$.

Двойственная задача (I*): $\max\{ yb' \mid yA' \le c,\ y \ge 0 \}$.

Здесь c - n-строка, x - n-столбец, b' - $(m+1)$-столбец, y - $(m+1)$-строка, A' - матрица порядка $(m+1) \times n$, полученная из матрицы коэффициентов задачи (I) указанным выше способом.

Если $n < m+1$, то решение прямой задачи заменяется решением двойственной задачи. В этом случае задача I* формулируется как задача (I), т.е. задача

$$\max\{ yb' \mid yA' \le c,\ y \ge 0 \}$$

эквивалентна задаче

$$-\min\{ -b'^{T} y^{T} \mid -A'^{T} y^{T} \ge -c^{T},\ y^{T} \ge 0 \}.$$

Здесь ^{T} - знак транспонирования.

Следовательно, в дальнейшем достаточно рассматривать только прямую задачу вида

$$\min\{ cx \mid Ax \ge b,\ x \ge 0 \} \quad \text{при} \quad m \le n. \tag{2}$$

Здесь и дальше одни и те же символы A, b, c, x, m, n, а также A', b' используются для обозначения различных величин, смысл которых определяется контекстом.

Для удобства описания алгоритма запишем задачу (2) в форме равенств с неотрицательными свободными членами. Вводя неотрицательные слабые переменные $s_1, \dots, s_m$, умножая на -1 (если это нужно для получения неотрицательного свободного члена), сформулируем основную задачу

$$\min\{ cx \mid A'x + I's = b',\ x \ge 0,\ s \ge 0 \},$$

где $b' \ge 0$. Здесь I' - диагональная матрица, диагональными элементами которой могут быть только числа -1 и 1.

Решение основной задачи эквивалентно решению следующей задачи

$$\min\{ cx + Met \mid A'x + I's + It = b',\ x \ge 0,\ s \ge 0,\ t \ge 0 \},$$

где M - достаточно большое положительное число; $e = (1,\dots,1)$, $t = (t_1,\dots,t_m)^T$ - столбец искусственных переменных, I - единичная матрица порядка $m \times m$.

Из этой задачи выделим вспомогательную задачу

$$\min\{ et \mid A'x + I's + It = b',\ x \ge 0,\ s \ge 0,\ t \ge 0\}.$$

При решении задачи LP сначала решается вспомогательная задача, а затем - основная задача. Первый этап решения состоит в определении допустимого базисного решения основной задачи путем нахождения оптимального решения вспомогательной задачи. Начальный допустимый базис вспомогательной задачи состоит из единичных векторов матрицы I ($b' \ge 0$), а существование её оптимального решения гарантируется видом самой задачи. Связь между вспомогательной и основной задачами более подробно рассмотрена ниже.

Как вспомогательная, так и основная задача решаются при помощи симплекс-метода. Известно, что симплекс-метод состоит из последовательности одинаковых по виду итераций. Описание произвольной итерации симплекс-метода дано в п.2.

2. Произвольная итерация симплекс-метода.

Введем обозначения:

$N = \{1,\dots,n\}$;

$A^j = (a_{1j},\dots,a_{mj})^T \quad (j \in N)$;

$b = (b_1,\dots,b_m)^T$; $y = (y_1,\dots,y_m)$;

$yA^j = \sum_{i=1}^{m} y_i a_{ij}$ - скалярное произведение.

Рассмотрим следующую пару двойственных задач.

Прямая задача:

$$\min\{ \sum_{j \in N} c_j x_j \mid \sum_{j \in N} x_j A^j = b,\ x_j \ge 0 \quad (j \in N)\}.$$

Двойственная задача:

$$\max\{ \sum_{i=1}^{m} y_i b_i \mid yA^j \le c_j \quad (j \in N)\}.$$

Предположим, что заданы допустимое базисное множество номеров $J = \{j_1,\dots,j_m\}$ ($J \subset N$) и обратная матрица B^{-1}, где $B = (A^{j_1},\dots,A^{j_m})$.

Произвольную итерацию алгоритма осуществляем в следующие четыре этапа.

I. Находим единственное решение y системы линейных уравнений $yA^j = c_j$ ($j \in J$), т.е. системы $yB = c(J)$.

2. Если для найденного вектора y выполняются неравенства $yA^j \le c_j$ ($j \in N$), то векторы ($x(j) = B^{-1} b$, $x_j = 0$, $j \notin J$) и $y = c(j)B^{-1}$ являются оптимальными в прямой и двойственной задачах (процесс заканчивается). В противном случае находим некоторый номер j_0, для которого $yA^{j_0} > c_{j_0}$ и величина $yA^{j_0} - c_{j_0}$ максимальна (первый такой номер $j_0 \in N$). Переходим к следующему этапу.

3. Находим коэффициенты разложения вектора A^{j_0} по базису B, т.е. решаем систему $Bz = A^{j_0}$, $z = B^{-1} A^{j_0}$. Запишем тождество

$$\varepsilon A^{j_0} + \sum_{j \in J} (x_j - \varepsilon z_j) A^j = b;$$

при

$$\varepsilon \ge 0 \quad \text{и} \quad x_j - \varepsilon z_j \ge 0 \quad (j \in J)$$

получаем некоторый допустимый вектор прямой задачи, который обозначаем через x^ε. Имеем

$$cx^\varepsilon = c(j)x(j) + \varepsilon c_{j_0} - \varepsilon c(j)z = c(J)x(J) + \varepsilon (c_{j_0} - yA^{j_0}) \le c(J)x(J).$$

Вектор x^ε может дать меньшее значение целевой функции.

4. Если $z_j \le 0$ для всех $j \in J$, то целевая функция не ограничена снизу, и процесс закончен.

Находим $\varepsilon_0 = \min_{j \in J,\, z_j > 0} (x_j / z_j)$. Определим множество $J_0 = \{ j : j \in J,\ x_j / z_j = \varepsilon_0 \}$. Если вырожденность учитывать не надо, то выбираем первый номер j^* в множестве J_0 для его исключения из этого множества (случай вырожденности рассмотрим в замечании).

Строим новое допустимое базисное множество $J' = (J \setminus j^*) \cup j_0$. Новому допустимому базисному множеству J' отвечает новый базис B', вектор $x(J')$ и вектор $y(J')$.

Принимая построенное допустимое базисное множество J' за исходное, повторяем этапы I-4. Прежде чем это сделать, необходимо произвести следующие преобразования.

Пусть неособенные матрицы $P = (p_{ik})$ и $P' = (p'_{ik})$ отличаются всего лишь одним столбцом с номером ν: $p_{ik} = p'_{ik}$ ($i \in M = \{1,\dots,m\}$; $k \in M \setminus \nu$).

Тогда элементы обратной матрицы $P'^{-1}=(q'_{ik})$ выражаются через элементы обратной матрицы $P^{-1}=(q_{ik})$ следующим образом:

$$q'_{\nu k}=q_{\nu k}/z_\nu\,,\quad q'_{ik}=q_{ik}-(q_{\nu k}/z_\nu)\,z_i$$

$$(\,i\in M\setminus\nu\,;\ k\in M\,),$$

где $(\,z_1,\dots,z_m\,)^T=P^{-1}(\,p'_{1\nu},\dots,p'_{m\nu}\,)^T$, $z_\nu\neq 0$.

Вычислим векторы $x(J')$ и $y(J')$:

$$x'_j=x_j-\varepsilon_0 z_j\qquad(j\in J'\setminus j_0),$$

$$x'_{j_0}=\varepsilon_0;\ y(J')=y_j-(\,y(J)\,A^{j_0}-c_{j_0})Q'_\nu$$

(Q'_ν – ν –я строка обратной матрицы B'^{-1}).

ЗАМЕЧАНИЕ. Рассмотрим случай вырожденности. Параметр r, характеризующий степень вырожденности, равен числу нулевых базисных компонент вектора $x(J)$ $(r=0,1,\dots,m)$.

При $r=0,1$ вырожденность не учитывается. При $r\geq 2$ определение номера J^* для его исключения производится по следующим правилам.

Пусть $\bar J$– допустимое базисное множество с вырожденностью $r\geq 2$. Разобьем множество $\bar J$ на два подмножества $\bar J=R_1\cup R_2$, где $R_2=\{\,j:\ j\in\bar J,\ x_j=0\}=\{\,\bar j_1,\dots,\ \bar j_r\},\ R_1=\bar J\setminus R_2$.

Начиная с множества $\bar J$ и до тех пор, пока не изменится значение целевой функции (значение изменится, если $\varepsilon_0\neq 0$), рассматриваем векторы b, $A^{\bar j_1},\dots,A^{\bar j_r}$ и соответственно их разложения по базису:

$$x(\bar J),\ z^{\bar j_1},\dots,z^{\bar j_r}.$$

Определяем последовательные множества :

$$\varepsilon_0=\min_{j\in\bar J,\,z_j>0}\left(\frac{x_j}{z_j}\right),\quad \bar J_0=\left\{j;\ j\in\bar J,z_j>0\,,\left(\frac{x_j}{z_j}\right)=\varepsilon_0\right\},$$

$$\varepsilon_1=\min_{j\in\bar J_0}\left(\frac{z^{\bar j_1}_j}{z_j}\right),\qquad \bar J_1=\left\{j:\ j\in\bar J_0\,,\left(\frac{z^{\bar j_1}_j}{z_j}\right)=\varepsilon_1\right\}.$$

На некотором s–м шаге множество $\bar J_s$ $(1\leq s\leq r)$ будет состоять только из одного элемента j^*. Вектор с номером j^* подлежит исключению из базиса.

3. Связь между вспомогательной и основной задачами.

При переходе от вспомогательной задачи к основной возможны следующие три случая.

I. Если оптимальное решение вспомогательной задачи во множестве J не содержит искусственных переменных t_i, переходим к решению основной задачи.

2. Если оптимальное решение вспомогательной задачи во множестве J содержит только нулевые искусственные переменные t_i (при этом значение целевой функции равно нулю), то все эти нулевые искусственные переменные t_i "просто исключаются" (см. замечание I) и переходим к основной задаче.

3. Если оптимальное решение вспомогательной задачи во множестве J содержит хотя бы одну положительную искусственную переменную и решается прямая задача, то исходная задача несовместна. Если же решается двойственная задача, то столбец свободных членов полагается равным нулю, все искусственные переменные t_i "просто исключаются", а затем решается полученная новая вспомогательная задача. Если минимум этой задачи конечен, то исходная задача совместна, а значения её целевой функции не ограничены снизу; если же минимум этой задачи бесконечен, то исходная задача несовместна.

ЗАМЕЧАНИЕ I. Переменная j' "просто исключается" – означает, что вектор базиса с номером $j'\in J$, соответствующий этой переменной, заменяется на любой другой вектор, соответствующий неискусственной переменной, т.е. следует найти такой вектор из матрицы (A',I'), который имеет ненулевую компоненту $z_{j'}$ в разложении по данному базису.

ЗАМЕЧАНИЕ 2. Определив допустимое базисное множество j основной задачи, вычисляем $y(j)$ из соотношения:

$$y(j)=c(j)\,B^{-1}$$

В следующих параграфах приводятся АЛГОЛ-программы, реализующие описанную здесь вычислительную схему.

При составлении АЛГОЛ-программ была использована унифицированная запись из [I].

§2. Первая процедура симплекс-метода

Приведем процедуру SIMP1 симплекс-метода задачи (I), в которой в полной форме записываются матрица ограничений по строкам, столбец свободных членов, строка коэффициентов целевой функции и текущая обратная матрица.

Обращение к процедуре имеет вид:

SIMP 1 (m,n,l,a,b,c,i0,a0,x).

Здесь:

m – параметр типа целый – число ограничений, за исключением ограничения вида $x_j \geq 0$ (j=1,...,n);

n – параметр типа целый – число неотрицательных переменных;

l – параметр, имеющий в качестве описания целый массив l[1:m+1] и состоящий из чисел –I, 0, I в зависимости от вида ограничений задачи (I), а именно: l[i] = –1, если i-е ограничение задачи (I) вида ≤ ; l[i] = 0 , если i-е ограничение в задаче (I) вида = ; l[i] = 1, если i-е ограничение в задаче (I) вида ≥ ;

a – параметр, имеющий в качестве описания вещественный массив a[1:m+1, 1:n] и m первых строк которого являются m строками ограничений задачи (I);

b – параметр, имеющий в качестве описания вещественный массив b[1:m+1] и m первых компонент которого являются m компонентами столбца свободных членов в задаче (I);

c – параметр, имеющий в качестве описания вещественный массив c[1:n] и состоящий из коэффициентов целевой функции;

i0 – параметр типа целый. Если по окончании процедуры i0 = –1, то решения задачи (I) не существует (ограничения несовместны); если i0 = 0, решения задачи (I) не существует (значения целевой функции не ограничены снизу); если i0 > 0 , решение задачи существует;

a0 – параметр типа вещественный. Если по окончании процедуры i0 > 0 , то он равен оптимальному значению целевой функции, если i0 ≤ 0 , этот параметр не несет никакой информации;

x – параметр, имеющий в качестве описания вещественный массив x[1:n] . Если по окончании процедуры i0 > 0 , то в

массиве x находятся компоненты оптимального решения задачи (I); если i0=0 ,то-компоненты допустимого решения; если i0 = –I, то массив x не несет никакой информации о задаче.

Процедуру SIMP 1 целесообразно использовать, например, в случае, когда нет информации о заполняемости нулями матрицы ограничений задачи (I), либо когда её заполняемость ненулевыми элементами превышает 50%.

```
procedure  SIMP1 (m,n,l,a,b,c,i0,a0,x); value m,n;  integer m,
    n,i0; real a0; integer array l; real array a,b,c,x; begin
    integer i,j,p,q;
procedure simp (m,n,b,c); begin integer j1,h0,r,s,u,w;real a1,
    a2,e; integer array bt[1:m];real array p,y,z[1:m],bm[1:m,
    1:m]; boolean b1;
procedure start;begin e:= 10-8;for i:=1 step 1 until m do begin
    if b[i]<-e then begin l[i]:=1;  for j:=1  step 1 until n
    do  if q=1  then a[i,j]:= -a[i,j]  else a[j,i]:= -a[j,i];
    b[i]:=-b[i] end else l[i]:=-1; for j:=1 step 1 until m do
    bm[i,j]:=0; bm[i,i]:=y[i]:=1; bt[i]:=n+m+i end; w:=0;b1:=
    false
end start;
procedure spos(j); value j; integer j;begin if j≤n then begin
    for i:=1 step 1 until m do p[i]:=if q=1 then a[i,j]  else
    a[j,i] end else begin for i:=1 step 1 until m do p[i]:=0;
    if j≤n+m then p[j-n]:=sign(l[j-n])  else p[j-n-m]:=1
end end spos;
procedure mult (u,b,a); begin a:=0; for u:=1 step 1 until m do
    a:=a+b x p[u]
end mult;
procedure new; begin j:=1; a1:=0;
s1: if j≤r then begin spos(j);  a2:= if j≤n  then (if r=n+m
    then c[j] else 0) else(if j≤n+m then 0 else 1);  mult(u,
    y[u],a0); a0:=a0-a2; if a0>a1+e then begin a1:=a0; j1:=j
    end; j:=j+1;goto s1 end else begin if a1<e then begin if
    r=n+m then begin if q=2∧b1 then i0:=0; goto l6 end; goto
    l2
end end end new;
```

```
procedure place; begin integer h,s,t; integer array gm[1:m];
     real array x[1:m]; spos(j1); i0:=h:=s:=0; for i:=1 step
     1 until m do begin if b[i]< e then h:=h+1; mult(u, bm[i,
     u], z[i]); if z[i] >e then begin a2:=b[i]/z[i]; if a2< e
     then begin s:=s+1;gm[s]:=i end;if i0=0 v a2< a0 then begin
     a0:=a2; i0:=i end end end; if h >1 ∧ a0< e then begin w:=
     w+1; if w=1 then begin h0:=h; h:=0; for i:=1 step 1 until
     m do if b[i]< e then begin h:=h+1; l[h]:= sign(l[h]) x
     bt[i] end end; for h:=1 step 1 until h0 do begin if s=1
     then goto s2; spos(abs(l[h])); t:=s; i0:=s:=0; for j:=1
     step 1 until t do begin i:=gm[j]; mult(u, bm[i,u], x[i]);
     a2:=x[i]:=x[i]/z[i]; if i0=0 ∨ a2< a0 then begin a0:=a2;
     i0:=i end end; for j:=1 step 1 until t do begin i:=gm[j];
     if abs(x[i]-a0)< e then begin s:=s+1;gm[s]:=i end end end;
s2:  a0:=0 end else w:=0; if i0=0 then begin if q=2 then
     begin i0:=-1; goto lk end; goto 16
end end place;

procedure work; begin bt[i0]:=j1; b[i0]:=a0; for j:=1 step 1
     until m do begin bm[i0,j]:=bm[i0,j]/z[i0]; y[j]:=y[j]-a1
     x bm[i0,j] end; for i:=1 step 1 until i0-1, i0+1 step 1
     until m do begin b[i]:=b[i]-a0 x z[i]; for j:=1 step 1
     until m do bm[i,j]:=bm[i,j]-bm[i0,j] x z[i]
end end work;

procedure simple; begin a0:=0;
s3:  if s≤ m then begin if bt[s]> r then begin i0:=s; j:=1;
s4:  spos(j); mult (u,bm[s,u],z[s]);if abs(z[s])> e then begin
     j1:=j;mult(u,y[u],a1);for i:=1 step 1 until s-1, s+1 step
     1 until m do mult(u,bm[i,u],z[i]); work; s:=s+1; goto s3
     end; j:=j+1; goto s4 end; s:=s+1; goto s3
end end simple;

procedure fin; begin a0:=0;for i:=1 step 1 until m do if bt[i]
     ≤ n then a0:=a0+c[bt[i]] x b[i];if q=1 then begin for j:=
     1 step 1 until n do x[j]:=0; for i:=1 step 1 until m do
     if bt[i]≤ n then x[bt[i]]:=b[i] end else begin for i:=1
     step 1 until m do x[i]:=abs(y[i]); a0:=-a0
end end fin;
```



```
start;  r:=n+2 x m;
  l1:   new; place; work; goto l1;
  l2:   r:=n+m; for i:=1 step 1 until m do if bt[i]> r then
        begin s:=i; goto l3 end; goto l5;
  l3:   for i:=s step 1 until m do if bt[i]> r∧ b[i]> e then
        goto l4; simple; goto l5;
  l4:   b1:=true; if q=1 then begin i0:=-1; goto lk end; for
        i:=1 step 1 until m do b[i]:=0; simple;
  l5:   for j:=1 step 1 until m do p[j]:= if bt[j]≤ n then
        c[bt[j]] else 0; for j:=1 step 1 until m do begin
        y[j]:=0; for i:=1 step 1 until m do y[j]:=y[j]+p[i]x
        bm[i,j] end; goto l1;
  l6:   fin;
  lk:   end simp;

        p:=m;b[m+1]:=0;for j:=1 step 1 until n do a[m+1,j]:=0;
        for i:=1 step 1 until m do begin if l[i]=-1 then begin
        for j:=1 step 1 until n do a[i,j]:=-a[i,j];b[i]:=-b[i]
        end; if l[i]=0 then begin p:=m+1;for j:=1 step 1 until
        n do a[p,j]:=a[p,j]-a[i,j]; b[p]:=b[p]-b[i] end end;
        q:=1; if n< p then begin q:=2; for i:=1 step 1 until p
        do begin for j:=1 step 1 until n do a[i,j]:=-a[i,j];
        b[i]:=-b[i] end; for j:=1 step 1 until n do c[j]:=
        -c[j]; simp(n,p,c,b) end else simp(p,n,b,c)
end simp1;
```

§3. Вторая процедура симплекс-метода

Приведем процедуру **SIMP2** симплекс-метода для задачи (I), матрица ограничений которой записывается в краткой форме (без нулей), а столбец свободных членов, строка коэффициентов целевой функции и текущая обратная матрица – в полной форме.

Обращение к процедуре имеет вид

SIMP2(m,n,n1,l,k,a,b,c,i0,a0,x).

Здесь параметры m,n,l,b,c,i0,a0,x означают то же, что и соответствующие параметры SIMP1 из §2. Новые параметры имеют следующий смысл:

n1 – параметр типа целый – равен сумме числа ненулевых элементов матрицы ограничений задачи (I) плюс n плюс число ну-

левых столбцов (столбцов, полностью состоящих из нулей), если они существуют;

k - параметр, имеющий в качестве описания целый массив k[1:n1] . В массив k заносятся номера строк ненулевых элементов, если столбец матрицы ограничений задачи (I) не является полностью нулевым, и одно из чисел 1,...,m ,если он нулевой. При этом столбцы матрицы ограничений перебираются в порядке возрастания их номеров, и номера строк при выбранном столбце заносятся в порядке их возрастания. После каждого столбца (будь то нулевой, либо ненулевой) ставится цифра 0, которая является признаком окончания столбца матрицы ограничений;

a - параметр, имеющий в качестве описания вещественный массив a [1:n1] . В массив a заносятся ненулевые элементы столбца матрицы ограничений задачи (I), если он не является полностью нулевым, 0 - в противном случае.Столбцы записываются в порядке возрастания их номеров, а ненулевые элементы столб - ца - в порядке возрастания нумерации строк матрицы ограничений. Признаком окончания столбца является постановка 0 (точно так же, как и при описании массива k).

Процедуру SIMP2 целесообразно использовать,например, в случае, когда заполняемость ненулевыми элементами матрицы ог - раничений задачи (I) меньше 50%.

```
procedure SIMP2(m,n,n1,l,k,a,b,c,i0,a0,x);value m,n,n1;integer
    m,n,n1,i0; real a0; integer array l,k; real array a,b,c ,
    x; begin integer i,j,p,q,m1; real e,sm; integer array
    kp[1:m+2], mu[1:n+1];

procedure simp (m,n,mu,b,c); begin integer j1,h0,r,s,u,v,w;
    real a1,a2;integer array d,bt[1:m];real array p,y,z[1:m],
    bm[1:m,1:m]; boolean b1;

procedure start;begin for j:=1 step 1 until n1 do if b[k[j]]<
    -e then a[j]:=-a[j]; for i:=1 step 1 until m do begin if
    b[i]< -e then begin l[i]:=1;b[i]:=-b[i]end else l[i]:=-1;
    for j:=1 step 1 until m do bm[i,j]:=0; bm[i,i]:=y[i]:=1;
    bt[i]:=n+m+i end; w:=0; b1:=false
end start;

procedure spos(j); value j; integer j; begin for i:=1 step 1
    until m do p[i]:=d[i]:=0; if j≤ n then begin for i:=mu[j]
```



```
    step 1 until mu[j+1]-1 do begin v:=1+i-mu[j]; p[v]:=a[i];
    d[v]:=k[i] end; if d[1]=0 then begin v:=1;d[1]:=1 end end
    else begin v:=1;if j≤ n+m then begin p[1]:=sign (l[j-n]);
    d[1]:=j-n end else begin p[1]:=1; d[1]:=j-n-m
end end end spos;

procedure mult (u, b, a); begin a:=0; for u:=1 step 1 until v
    do a:=a+b x p[u]
end mult;

procedure new; begin j:=1; a1:=0;
s1: if j≤ r then begin spos(j); a2:=if j≤ n then (if r=n+m
    then c[j] else 0) else (if j≤ n+m then 0 else 1); mult
    (u,y[d[u]],a0); a0:=a0-a2; if a0> a1+e then begin a1:=a0;
    j1:=j end; j:=j+1; goto s1 end else begin if a1< e then
    begin if r=n+m then begin if q=2∧b1 then i0:=0; goto l6
    end; goto l2
end end end new;

procedure place; begin integer h,s,t; integer array gm[1:m];
    real array x[1:m]; spos (j1); i0:=h:=s:=0; for i:=1 step
    1 until m do begin if b[i]< e then h:=h+1; mult (u,bm [i,
    d[u]], z[i]); if z[i]> e then begin a2:=b[i]/z[i];if a2<e
    then begin s:=s+1;gm[s]:=i end;if i0=0∨a2< a0 then begin
    a0:=a2; i0:=i end end end; if h> 1∧a0< e then begin w:=
    w+1; if w=1 then begin h0:=h; h:=0; for i:=1 step 1 until
    m do if b[i]< e then begin h:=h+1; l[h]:=sign (l[h]) x
    bt[i] end end; for h:=1 step 1 until h0 do begin if s=1
    then goto s2; spos (abs(l[h])); t:=s; i0:=s:=0; for j:=1
    step 1 until t do begin i:=gm[j]; mult (u, bm[i,d[u]] ,
    x[i]); a2:=x[i]:=x[i]/z[i]; if i0=0∨a2< a0 then begin
    a0:=a2; i0:=i end end; for j:=1 step 1 until t do begin i
    :=gm[j]; if abs(x[i]-a0)< e then begin s:=s+1; gm[s]:=i
    end end end;
s2: a0:=0 end else w:=0; if i0=0 then begin if q=2 then begin
    i0:=-1; goto lk end; goto l6
end end place;
```

```
procedure work; begin bt[i0]:=j1; b[i0]:=a0; for j:=1 step 1
    until m do begin bm[i0,j]:= bm[i0,j]/z[i0]; y[j]:=y[j]-a1
    x bm[i0,j] end; for i:=1 step 1 until i0-1, i0+1 step 1
    until m do begin b[i]:=b[i]-a0 x z[i]; for j:=1 step 1
    until m do bm[i,j]:= bm[i,j]-bm[i0,j] x z[i]
end end work;

procedure simple; begin a0:=0;
s3: if s≤m then begin if bt[s]>r then begin i0:=s; j:=1;
s4: spos (j); mult(u, bm[s,d[u]],z[s]); if abs (z[s])> e then
    begin j1:=j; mult (u,y[d[u]], a1); for i:=1 step 1 until
    s-1,s+1 step 1 until m do mult (u,bm[i,d[u]],z[i]); work;
    s:=s+1; goto s3 end; j:=j+1; goto s4 end; s:=s+1; goto s3
end end simple;

procedure fin; begin a0:=0; for i:=1 step 1 until m do if
    bt[i]≤ n then a0:=a0+c[bt[i]] x b[i]; if q=1 then begin
    for j:=1 step 1 until n do x[j]:=0; for i:=1 step 1 until
    m do if bt[i]≤ n then x[bt[i]]:=b[i] end else begin for
    i:=1 step 1 until m do x[i]:= abs(y[i]); a0:= -a0
end end fin;

start; r:=n+2 x m;
  l1: new; place; work; goto l1;
  l2: r:=n+m; for i:=1 step 1 until m do if bt[i]> r then
      begin s:=i; goto l3 end; goto l5;
  l3: for i:=s step 1 until m do if bt[i]> r∧b[i]> e then
      goto l4; simple; goto l5;
  l4: bl:=true; if q=1 then begin i0:=-1; goto lk end; for
      i:=1 step 1 until m do b[i]:=0; simple;
  l5: for j:=1 step 1 until m do p[j]:=if bt[j]≤ n then c[bt
      [j]] else 0; for j:=1 step 1 until m do begin y[j]:=0;
      for i:=1 step 1 until m do y[j]:=y[j]+p[i] x bm[i,j]
      end; goto l1;
  l6: fin;
  lk: end simp;

procedure trans; begin integer h,l,r,s,t,u,v; integer array
    lm[1:m1]; real array bm[1:m1]; kp[1]:=1; s:=t:=u:=0;for
    h:=1 step 1 until p do begin
s1: for j:=1 step 1 until n do for i:=mu[j] step 1 until mu
    [j+1]-1 do if k[i]=h then begin t:=t+1; u:=u+1; lm[u]:=j;
    k[i]:=0; bm[u]:=a[i]; if t=n1 then goto s3; if u=m1 then
    goto s2 end; goto s4;
s2: v:=0; r:=n1+1; l:=n; for i:=n1 step -1 until s+1 do begin
    if k[i]≠0 then begin r:=r-1; k[r]:=k[i]; a[r]:=a[i] end
    else v:=v+1; if i=mu[l] then begin mu[l]:=mu[l]+v; l:=l-1
    end end;
s3: for j:=1 step 1 until u do begin k[s+j]:=lm[j]; a[s+j]:=
    bm[j] end; s:=s+u; u:=0; if t<n1 then goto s1;
s4: kp[h+1]:=t+1
end end trans;
    e:= 10-8; sm:=b[m+1]:=0; p:=m; q:=0; k[1]:=-k[1]; for j:=
    1 step 1 until n1 do begin i:=abs(k[j]);if i≠0 then begin
    if l[i]=-1 then a[j]:=-a[j]; if l[i]=0 then sm:=sm-a[j ]
    end else begin if j≠n1 then k[j+1]:=-k[j+1];if abs(sm)> e
    then begin a[j]:=sm; k[j]:=m+1;sm:=0; q:=q+1 end end end;
    for i:=1 step 1 until m do begin if l[i]=-1 then b[i]:=-b
    [i]; if l[i]=0 then begin p:=m+1; b[p]:=b[p]-b[i]end end;
    if p=m+1 then begin if q=0 then begin if b[p]> e then
    begin i0:=-1; goto lk end; p:=m end end; q:=0; for j:=1
    step 1 until n1 do if k[j]≠0 then begin q:=q+1; k[q]:=
    k[j]; a[q]:=a[j] end; n1:=q; q:=1; j:=0; for i:=1 step 1
    until n1 do if k[i]< 0 then begin j:=j+1; mu[j]:=i;k[i]:=
    -k[i] end; mu[j+1]:=n1+1; if n< p then begin q:=2; for
    i:=1 step 1 until p do b[i]:=-b[i]; for j:=1 step 1 until
    n1 do a[j]:=-a[j];for j:=1 step 1 until n do c[j]:=-c[j];
    m1:=(p↑2+p)/2; trans; simp (n,p,kp,c,b) end else simp(p,
    n,mu,b,c);
lk: end simp2;
```

§ 4. Третья процедура симплекс-метода

Приведем процедуру SIMP3 симплекс-метода для задачи (I), в которой матрица ограничений и текущая обратная матрица записываются в краткой форме (без нулей), а столбец свободных членов и строка коэффициентов целевой функции – в полной форме.

Обращение к процедуре имеет вид:

SIMP3(m,n,n1,m1,l,k,a,b,c,i0,a0,x).

Здесь параметры m,n,n1,l,k,a,b,c,a0,x имеют тот же смысл,что и соответствующие параметры процедуры SIMP2 из §3. Новые параметры имеют следующий смысл:

m1 - параметр типа целый означает, что под текущую обратную матрицу отводится не более чем 2 x m1 ячеек ОЗУ.

i0 - параметр типа целый. Если по окончании процедуры i0> 0 , то решение задачи (I) существует, если i0 = 0 , решения задачи (I) не существует (значения целевой функции не ограничены снизу); если i0 =-1, решения задачи (I) не существует (ограничения несовместны); если i0 =-2 , не хватает отведенного под текущую обратную матрицу места (текущая обратная матрица занимает больше 2 x m1 ячеек ОЗУ).

Процедуру SIMP3 целесообразно использовать, например,в случае, когда заполняемость ненулевыми элементами матрицы ограничений задачи (I) на много меньше 50% и когда текущая обратная матрица,записываемая в полной форме, не помещается в оперативную память машины.

```
procedure SIMP3(m,n,n1,m1,l,k,a,b,c,i0,a0,x); value m,n,n1,m1;
    integer m,n,n1,m1,i0; real a0; integer array l,k; real
    array a,b,c,x; begin integer i,j,p,q; real e, sm; integer
    array kp[1:m+2],mu[1:n+1];

procedure simp(m,n,mu,b,c); begin integer j1,h,h0,r,s,u,v,w;
    real a1,a2; integer array d,bt[1:m],lb[1:m+1],lm[1:m1 ];
    real array p,y,z[1:m], bm[1:m1]; boolean b1;

procedure start; begin h:=1; for j:=1 step 1 until n1 do if b
    [k[j]]< -e then a[j]:=-a[j]; for i:=1 step 1 until m do
     begin if b[i]< -e then begin l[i]:=1; b[i]:=-b[i] end
     else l[i]:=-1; bm[i]:=y[i]:=1;lm[i]:=lb[i]:=i;bt[i]:=n+
     m+i end; lb[m+1]:=m+1; w:=0; b1:=false
end start;

procedure spos(j); value j; integer j; begin for i:=1 step 1
    until m do p[i]:=d[i]:=0; if j≤ n then begin for i:=mu
    [j] step 1 until mu[j+1]-1 do begin v:=1+i-mu[j]; p[v]:=
    a[i]; d[v]:=k[i] end; if d[1]=0 then begin v:=1;d[1]:=1
    end end else begin v:=1; if j≤ n+m then begin p[1]:=sign
    (l[j-n]); d[1]:=j-n end else begin p[1]:=1;d[1]:=j-n-m
end end end spos;
```



```
procedure mult(a); begin a:=0; for u:=1 step 1 until v do a:=
    a+y[d[u]] x p[u]
end mult;

procedure prod(i,z); begin integer j,u; z:=0; u:=1; for j:=lb
    [i] step 1 until lb[i+1]-1 do begin
s0: if lm[j]≥ d[u]then begin if lm[j]=d[u] then z:=z+bm[j] x
    p[u];u:=u+1;if u≤ v then goto s0 else goto sk end end;
sk: end prod;

procedure new; begin j:=1; a1:=0;
s1: if j≤ r then begin spos(j); a2:=if j≤ n then (if r=n+m
    then c[j] else 0) else (if j≤ n+m then 0 else 1); mult
    (a0); a0:=a0-a2; if a0> a1+e then begin a1:=a0;j1:=j end;
    j:=j+1;goto s1 end else begin if a1< e then begin if r=n+
    m then begin if q=2∧b1 then i0:=0; goto l6 end;goto l2
end end end new;

procedure place; begin integer h,s,t; integer array gm[1:m];
    real array x[1:m]; spos(j1); i0:=h:=s:=0; for i:=1 step
    1 until m do begin if b[i]< e then h:=h+1; prod(i,z[i]);
    if z[i]> e then begin a2:=b[i]/z[i]; if a2< e then begin
    s:=s+1; gm[s]:=i end; if i0=0∨a2< a0 then begin a0:=a2;
    i0:=i end end end;if h> 1∧a0 <e then begin w:=w+1;if w=1
    then begin h0:=h; h:=0;for i:=1 step 1 until m do if b[i]
    < e then begin h:=h+1; l[h]:=sign(l[h]) x bt[i] end end;
    for h:=1 step 1 until h0 do begin if s=1 then goto s2;
    spos(abs(l[h])); t:=s; i0:=s:=0; for j:=1 step 1 until t
    do begin i:=gm[j]; prod(i,x[i]); a2:= x[i]:= x[i]/z[i];
    if i0=0∨a2 <a0 then begin a0:=a2; i0:=i end end;for j:=1
    step 1 until t do begin i:=gm[j]; if abs(x[i]-a0)< e then
    begin s:=s+1;gm[s]:=i end end end;
s2: a0:=0 end else w:=0; if i0=0 then begin if q=2 then begin
    i0:=-1;goto lk end; goto l6
end end place;

procedure work;begin integer k,l,p,q,r,s,t,u,v,w; real sm,tau;

procedure f(lf,a,b); begin lf:=if h=1 then a else b
end f;
```

```
procedure g(u,sm); value u,sm; integer u; real sm; begin if
     abs(sm)> e then begin lm[w]:=u; bm[w]:=sm; w:=w+h
end end g;

     bt[i0]:=j1; b[i0]:=a0; for j:=lb[i0] step 1 until lb[i0
     +1]-1 do begin bm[j]:=bm[j]/z[i0];y[lm[j]]:=y[lm[j]] - a1
     x bm[j] end;for i:=1 step 1 until i0-1, i0+1 step 1 until
     m do b[i]:=b[i]-a0 x z[i]; h:=-h; f(p,1,m); f(q,m,1) ;
     f(s,lb[i0],lb[i0+1]-1); f(t,lb[i0+1]-1,lb[i0]);f(k,lb[1],
     lb[m+1]-1); f(w,1,m1); r:=s-t; if h=1 then lb[1]:=1 else
     lb[m+1]:=m1+1; for i:=p step h until q do begin if i=i0
     then begin l:=t; s:=w; t:=w-r;goto w3 end; f(l,lb[i+1]-1,
     lb[i]);if abs(z[i])< e then goto w3;for j:=s step h until
     t do begin u:=lm[j]; sm:=-bm[j] x z[i];
w1:  if k x h> l x h then goto w2; v:=lm[k]; tau:=bm[k]; if
     u x h≥ v x h then begin k:=k+h; if u=v then g(v,sm+tau)
     else begin g(v,tau); goto w1 end end else begin
w2:  g(u,sm) end end; if k x h< w x h then begin i0:=-2;goto
     lk end;
w3:  for j:=k step h until l do g(lm[j], bm[j]); k:=l+h;if h=1
     then lb[i+1]:=w else lb[i]:=w+1
end end work;

procedure simple; begin a0:=0;
s3:  if s≤ m then begin if bt[s]>r then begin i0:=s; j:=1;
s4:  spos(j);prod(s,z[s]); if abs(z[s])> e then begin j1:=j;
     mult(a1); for i:=1 step 1 until s-1,s+1 step 1 until m do
     prod(i,z[i]); work; s:=s+1; goto s3 end; j:=j+1; goto s4
     end; s:=s+1; goto s3
end end simple;

procedure fin; begin a0:=0; for i:=1 step 1 until m do if bt
     [i]≤ n then a0:=a0+c[bt[i]] x b[i]; if q=1 then begin for
     j:=1 step 1 until n do x[j]:=0;for i:=1 step 1 until m do
     if bt[i]≤n then x[bt[i]]:=b[i] end else begin for i:=1
     step 1 until m do x[i]:=abs(y[i]); a0:=-a0
end end fin;

start; r:=n+2 x m;
  l1:  new; place; work; goto l1;
```



```
  l2:  r:=n+m; for i:=1 step 1 until m do if bt[i]> r then
       begin s:=i; goto l3 end; goto l5;
  l3:  for i:=s step 1 until m do if bt[i]>r∧b[i]>e then
       goto l4; simple; goto l5;
  l4:  bl:=true; if q=1 then begin i0:=-1; goto lk end; for
       i:=1 step 1 until m do b[i]:=0; simple;
  l5:  for j:=1 step 1 until m do begin p[j]:= if bt[j]≤ n
       then c[bt[j]] else 0;y[j]:=0 end;for i:=1 step 1 until
       m do for j:=lb[i] step 1 until lb[i+1]-1 do y[lm[j]]:=
       y[lm[j]]+p[i] x bm[j]; goto l1;
  l6:  fin;
  lk:  end simp;

procedure trans; begin integer h,l,r,s,t,u,v; integer array lm
     [1:m1]; real array bm[1:m1]; kp[1]:=1; s:=t:=u:=0; for
     h:=1 step 1 until p do begin
s1:  for j:=1 step 1 until n do for i:=mu[j] step 1 until mu
     [j+1]-1 do if k[i]=h then begin t:=t+1; u:=u+1; lm[u]:=j;
     k[i]:=0; bm[u]:=a[i]; if t=n1 then goto s3; if u=m1 then
     goto s2 end; goto s4;
s2:  v:=0; r:=n1+1; l:=n; for i:=n1 step -1 until s+1 do begin
     if k[i]≠0 then begin r:=r-1; k[r]:=k[i]; a[r]:=a[i] end
     else v:=v+1; if i=mu[l] then begin mu[l]:=mu[l]+v; l:=l-1
     end end;
s3:  for j:=1 step 1 until u do begin k[s+j]:=lm[j]; a[s+j]:=
     bm[j] end; s:=s+u; u:=0; if t< n1 then goto s1;
s4:  kp[h+1]:=t+1
end end trans;

     e:= 10-8; sm:=b[m+1]:=0; p:=m; q:=0; k[1]:=-k[1]; for j:=1
     step 1 until n1 do begin i:=abs(k[j]); if i≠0 then begin
     if l[i]=-1 then a[j]:=-a[j]; if l[i]=0 then sm:=sm-a[j]
     end else begin if j≠n1 then k[j+1]:=-k[j+1]; if abs(sm)>
     e then begin a[j]:=sm; k[j]:=m+1; sm:=0; q:=q+1 end end
     end;for i:=1 step 1 until m do begin if l[i]=-1 then b[i]
     :=-b[i]; if l[i]=0 then begin p:=m+1; b[p]:=b[p]-b[i] end
     end;if p=m+1 then begin if q=0 then begin if b[p]> e then
     begin i0:=-1; goto lk end; p:=m end end; q:=0; for j:=1
     step 1 until n1 do if k[j]≠0 then begin q:=q+1; k[q]:= k
```

```
    [j]; a[q]:=a[j] end; n1:=q;  q:=1;  j:=0; for i:=1 step 1
    until n1 do if k[i]<0 then begin j:=j+1;mu[j]:=i; k[i]:=
    -k[i] end; mu[j+1]:=n1+1; if n< p  then  begin q:=2;  for
    i:=1 step 1 until p do b[i]:=-b[i]; for j:=1 step 1 until
    n1 do a[j]:=-a[j];for j:=1 step 1 until n do c[j]:=-c[j];
    trans; simp(n,p,kp,c,b) end else simp(p,n,mu,b,c);
lk: end simp3;
```

Литература

1. ГРИБОВ А.Б., РОМАНОВСКИЙ И.В. Программирование симплекс-метода и его вариантов на АЛГОЛе.-В кн.: Оптимальное планирование. Вып. 12. Новосибирск, 1969, с. 5-27.

2. ЕРШОВ А.П., КОЖУХИН Г.И., ПОТТОСИН И.В. Руководство к пользованию системой АЛЬФА. ВЦ СО АН СССР. Новосибирск, 1968.

3. КУРОЧКИН В.М., ПОШИВАЛОВ Д.Б. и др. Система БЭСМ-АЛГОЛ (методическое руководство по программированию).ВЦ МГУ,М.,1969.

Поступила в ред.-изд.отд.
29 октября 1973 года