УДК 512.25 + 519.3

А.Б.ГРИБОВ, И.В.РОМАНОВСКИЙ

ПРОГРАММИРОВАНИЕ СИМПЛЕКС-МЕТОДА И ЕГО ВАРИАНТОВ НА АЛГОЛе

В этой статье будет приведен ряд процедур симплекс-метода с различной организацией вычислений и с различными способами задания исходных данных. Предполагается, что читатель знаком с теоретическими основами симплекс-метода [I] . Основная наша цель-привести унифицированную запись алгорифмов на языке АЛГОЛ-60. Меньшее внимание уделяется сокращению текста программ. В конце статьи с общих позиций дается обзор других статей сборника, посвященных решению задач линейного программирования.

I. Если отвлечься от деталей и представить симплекс-метод в виде отдельных частей, то его можно записать так:

```
start; l1: new; place; work; go to l1; l2: fin.
```

Здесь процедура start преобразует необходимую информацию к удобному для расчетов виду и готовит начальное приближение. Процедура new вырабатывает новый способ (отыскивается вектор из матрицы ограничений, подлежащий вводу в базис). Если такого способа не находится и задача считается решенной, происходит переход к метке $l2$. Место, освобожденное в базисе (из базиса исключается занимавший это место вектор), определяется в процедуре place . Процедура work производит изменение базиса и преобразует информацию, связанную с базисом. (В прямом симплекс-методе - это таблица коэффициентов разложения, строка базисных цен и т.д. В модифицированном - это обратная матрица, двойственные переменные и т.д.). Наконец, процедура fin приводит окончательные результаты расчета к требуемому виду.

2^0. Рассмотрим для примера игру с матрицей a[I:m,1:n]. Как известно [2], можно считать матрицу положительной и решение игры свести к задаче: найти $X[j] \geq 0$, $j=1,2,\ldots,n$, удовлетворяющие условиям: $\sum_{j=1}^{n} a[i,j] \times X[j] \leq 1$, $i=1,2,\ldots,m$, и максимизирующие $\sum_{j=1}^{n} X[j]$.

Для этой задачи, при использовании прямого симплекс-метода, процедуру start можно записать следующим образом:

```
procedure start;
    begin integer i,j; ввод (a); it:=0;
        for i:=I step I until m do
            begin for j:=I step I until m do b[i,j]:=0;
                b[i,i]:=X[i]:=I;base[i]:=n+i;d[i]:=0
            end
    end start;
```

Здесь base[i] - номер i -го элемента базиса; X[i] - интенсивность способа с номером base[i]; d[i] - цена способа, имеющего номер base[i]; it - счетчик итераций. Описанная процедура вводит необходимую информацию (матрица a), выбирает начальный базис (в оазис включены векторы, соответствующие дополнительным переменным), строит связанные с ним план (векторы X и base) и вектор базисных цен (d) и формирует таблицу (матрица b) коэффициентов разложения дополнительных векторов.

В дальнейшем нам будет удобно использовать процедуру spos(a,j,p) которая по номеру j вырабатывает столбец коэффициентов разложения способа j таблицы a по базису и записывает его в p[I:m].

В рассматриваемом примере она будет выглядеть так:

```
procedure spos(a,j,p); value j;integer j;array a,p;
  begin integer i; for i:=I step 1 until m do p[i]:=a[i,j]end;
```

Процедура new может быть записана многими способами. Вот вариант, когда происходит просмотр всех основных переменных, начиная с первой, и выбирается переменная с максимальной оценкой. Если эта оценка равна нулю, то аналогично просматриваются дополнительные переменные. Соответственно процедуру new мы составим из двух: new I и new 2.

```
procedure new;
   begin integer i,j; real a2;
         procedure new 1;
            begin for j:=I step I until n do
                     begin spos(a,j,p); a2:=I;
                           for i:=1 step I until m do
                              a2:=a2-p[i]×d[i];
                           if a2>a1 then begin aI:=a2;jI:= j end
                     end;
                  if aI≠0 then begin spos(a,j,p); go to 1 end
            end new I;
         procedure new 2;
            begin for j:=I step 1 until m do
                     begin spos(b,j,p); a2:=0;
                           for i:=I step I until m do
                              if base[i]≤n then a2:=a2-p[i];
                              if a2>aI then
                                 begin a2:=a1; jI:=j end
                     end;
                  if aI=0 then go to 12;
                  spos(b,j1,q);jI:=jI+n
            end new 2;
      aI:=0; new 1; new 2; 1:
   end new;
```

В этой процедуре j I -целочисленная переменная,описываемая во внешнем блоке (глобальная по отношению к new). Значение её - номер подлежащего вводу в базис способа. q - массив,содержащий коэффициенты разложения способа j1 по базису (тоже локализуемый во внешнем блоке).

Следующая процедура place определяет значение целочисленной переменной iI - номер освобождаемого в базисе места.

```
procedure place;
   begin integer i,j; real a2,a3;j:=0;
         for i:=I step I until m do
            if q[i]>0 then
               begin a3:=X[i]/q[i];
                     if j=0 then begin a2:=a3;iI:=i;j:=I end
                            else if a3 < a2 then
                                    begin a2:=a3; iI:=i end
                end
   end place;
```

Процедура work преобразует информацию, связанную с очередным базисом, в связи с переходом к новому базису (новый базис получается из старого заменой способа, стоящего на месте iI , на способ jI). Сюда относится изменение векторов X , base и d и матриц a и b

```
procedure work;
   begin integer i,j;
         base[iI]:=jI; it:=it+I;
         d[iI]:=if jI≤ n then I else 0; X[iI]:=X[iI]/q[iI];
         for j:=I step I until n do a[iI,j]:=a[iI,j]/q[iI];
         for j:=I step I until m do b[iI,j]:=b[iI,j]/q[iI];
         for i:=I step I until m do if i≠iI then
             begin X[i]:=X[i]-X[iI]×q[i];
                   for j:=I step I until n do
                       a[i,j]:=a[i,j]-a[iI,j]×q[i];
                   for j:=I step I until m do
                       b[i,j]:=b[i,j]-b[iI,j]×q[i]
             end
   end work;
```

Остается описать процедуру fin.

```
procedure fin;
   begin integer i;real v;v:=0;
       for i:=I step I until m do
           if base[i]<n then v:=v+X[i];
       v:=I/v;
       for i:=I step I until m do X[i]:=X[i]×v;
       вывод (X,base,v,it); stop
   end fin;
```

Здесь печатается стратегия максимизирующего игрока, вычисляется и печатается значение игры v и выдается на печать количество проделанных итераций.

В целом программа имеет следующий вид:

```
   begin integer m,n,it,iI,jI; ввод (m,n);
         begin array a[I:m,I:n],b[I:m,I:m],X,p,q,d[I:m];
               real aI;
               integer array base[I:m];
         comment Вместо этого комментария следует поместить
описание приведенных выше процедур;
                     start;lI: new; place;work;go to lI;l2: fin
         end
   end
```

3°. Теперь для того же примера приведем текст программы модифицированного симплекс-метода.

Процедуру start можно оставить без изменения. Отметим, что матрица a , рассматривавшаяся ранее как таблица коэффициентов разложения способов по базису, здесь является таблицей способов, матрица b теперь имеет смысл матрицы, обратной к текущему базису, а вектор d - вектора двойственных переменных (с обратными знаками).

Также без изменений можно оставить и процедуру epos но теперь она генерирует сам способ j .

В процедуре new изменится только процедура new 2.

```
procedure new 2;
  begin for j:=I step I until m do
          if  -d[j]>aI then begin jI:=j+n;aI:=-d[j] end;
      if aI=0 then go to 12
  end new2;
```

В прямом симплекс-методе процедура new готовила не только способ, но и коэффициенты разложения его по базису.

Здесь же коэффициенты разложения вычисляются в процедуре place.

```
procedure place;
   begin integer i,j; real a2,a3;iI:=0;
       for i:=I step 1 until m do
           begin if jI>n then p[i]:=b[i,jI-n]
                         else
                    begin p[i]:=0;
                          for j:=I step I until m do
                              p[i]:=p[i]+b[i,j]×q[j]
                    end;
                 if p[i]>0 then
                    begin a3:=X[i]/p[i];
                          if iI=0∨a3<a2 then
                             begin a2:=a3;iI:=i  end
                    end
           end
   end plase;
```

Перерабатываемая в следующей процедуре информация включает вектор X, base, d и матрицу b .

```
procedure work;
   begin integer i,j;
         base[iI]:=jI; it:=it+I; X[iI]:=X[iI]/p[iI];
         for j:=1 step 1 until m do
             begin b[iI,j]:=b[iI,j]/b[iI];
                   d[j]:=d[j]+b[iI,j]×aI
             end;
         for i:=I step I until m do
             if i≠iI then begin X[i]:=X[i]-X[iI]× p[i];
                                forj:=1step I until m do
                                    b[i,j]:=b[i,j]-b[iI,j]× p[i]
                          end
   end work;
```

В процедуре fin дополнительно к тому, что делалось ранее, сделаем выдачу (d) стратегии второго (минимизирующего)игрока.

```
procedure fin;
   begin integer i;real v; v:=0;
         for i:=I step I until m do
             if base[i]≤n then v:=v+X[i];v:=I/v;
         for i:=I step I until m do
             begin X[i]:=X[i]×v; d[i]:=d[i]×v end;
         вывод (X,base,d,v,it);stop
   end fin;
```

Сама программа сохраняет прежний вид.

4°. В этом параграфе мы запишем полностью программу модифицированного симплекс-метода для общей задачи линейного программирования в каноническом представлении.

Найти $X \geq \Phi$, удовлетворяющий $AX=B$ $(B \geq \Phi)$ и минимизирующий CX

```
begin integer m,n,it,iI,jI;real e,g,z; ввод (m,n,e,gz);
      begin real a0,aI;
            array p,q[0:m],d,x[I:m],a[0:m,I:n],b[I:m,I:n];
            integer array base[I:m];
```

comment m - количество ограничений, n -количество переменных, e - малое число, gz - большое число, элементы строки C (целевой формы) отнесены к нулевой строке матрицы A, a0 - барьер*);

*) Барьером служит число, превышение которого оценкой способа рассматривается как остановка поиска переменной, достаточно хорошей для включения в базис (ср. [3]).

```
procedure start;
   begin integer i,j;it:=0;ВВОД (a,X);a0:=gz;
         for i:=I step I until m do
             begin for j:=I step I until m do b[i,j]:=0;
                   b[i,j]:=I;base[i]:=0;d[i]:=gz
             end
   end start;
procedure spos(j,p);value j;integer j;array p;
   begin integer i;
         for i:=0 step I until m do p[i]:=a[i,j]
   end spos;
procedure new;
   begin integer i,j;real a2:aI:=0;
         for j:=I step I until n do
             begin spos(j,p);a2:=-p[0];
                   for i:=I step I until m do
                                a2:=a2+p[i]×d[i];
                   if a2>aI then
                      begin aI:=a2;jI:=j;
                            if aI>a0 then go to 1
                      end
             end;
         if aI<e then go to 12;a0:=aI/2;1: spos(jI,q);
        it:=it+I
   end new;

procedure place;
   begin integer i,j;real a2,a3;iI:=0;
         for i:=1 step I until m do
             begin p[i]:=0;
                   for j:=I step I until m do
                       p[i]:=p[i]+b[i,j]×q[j];
                   if p[i]>0 then
                      begin a3:=X[i]/p[i];
                            if iI=0∨a3<a2 then
                               begin a2:=a3;iI:=i
                               end
                      end
             end;
         if iI=0 then begin  ВЫВОД (aI,jI,p,q);go to 12 end
```

```
    end place;
procedure work;
    begin integer i,j;base[iI]:=jI;X[iI]:=X[iI]/p[iI];
          for j:=I step 1 until m do
              begin b[iI,j]:=b[iI,j]/p[iI];
                    a[j]:=a[j]-b[iI,j]×aI
              end;
          for i:=I step 1 until iI-1,iI+1 step I until m do
              begin X[i]:=X[i]-X[iI]×p[i];
                    for j:=I step 1 until m do
                        b[i,j]:=b[i,j]-b[iI,j]×p[i];
              end
    end work;
procedure fin;
    begin integer i,j; ВЫВОД(it,base,X,d);
          for i:=0 step I until m do p[i]:=0;
          for j:=I step I until m do
              if base[j]>0 then
                   begin spos(base[j],q);
                         for i:=0 step I until m do
                             p[i]:=p[i]+q[i]×X[j]
                   end;
        ВЫВОД  (p);stop
    end fin;
start; 1I:new;place;work;go to 1I;12: fin
end end
```

5⁰ Для решения задач со смешанными ограничениями:

$A_1X \leqslant B_1$, $A_2X \geqslant B_2$, $A_3X = B_3$ ($B_1, B_2, B_3 \geqslant 0$)

достаточно изменить процедуру new и первую строку программы.

```
procedure new;
    begin integer i,j;real a2;aI:=0;
          for j:=I step I until m do
              begin spos(j,p);a2:=-p[0];
                    for i:=I step I until m do
                        a2:=a2+p[i]×d[i];
                    if a2>aI then
                       begin aI:=a2;jI:=i;
                             if aI>a0 then go to 1
                       end
              end;
```

```
l: if aI < e
     then
          begin for i:=I step I until m do q[i]:=0;
                for j:=I step I until mI do
                    if d[j]>aI then
                        begin aI:=d[j];jI:=j;end ;
                for j:=I step I until m2 do
                    if -d[j+mI]>aI then
                        begin aI:=-d[j+mI];jI:=j+mI end;
                if aI < e then go to l2;
                q[jI]:=if  jI<= mI then I else -I;
                jI:=n-jI-n
          end
     else spos(jI,i);
   it:=it+I;if aI< a0 then a0:=aI/2
end new;
```

Первая строка программы заменяется двумя:

```
begin integer m,mI,m2,n,it,iI,jI;real e,gz;
     ввод  (m,mI,m2,n,e,gz);
```

Здесь mI - количество ограничений типа $\leq$,
а m2 - количество ограничений типа $\geq$.

6^0. Рассмотрим случай, когда запоминаются только ненулевые элементы матрицы ограничений и функционала.
В этом случае достаточно описать и ввести

t - число ненулевых элементов ;

a[I:t]-вектор,хранящий ненулевые элементы расширенной матрицы A (нулевая строка в расширенной матрице есть строка C), выписанные по столбцам ;

str[I:t] - вектор, j -тый элемент которого указывает номер строки, в которой стоит элемент a[j] ;

nam[I:n+I]- вектор,элемент которого nam[j]указывает номер та в векторе a , на котором стоит первый,отличный от нуля элемент столбца j ; nam[n+I]=t+I;

и изменить процедуру spos .

```
procedure spos(j,p); value j;integer j;array p;
    begin integer i,k;k:=0;
          for i:=0 step I until m do
              if i=str[nam[j]+k]
                 then begin p[i]:=a[nam[j]+k];k:=k+I;
                            if nam[j]+k=nam[j+I] then k:=k-I;
                      end
                 else p[i]:=0
    end  spos;
```

Описание программы можно сделать следующее:

```
    begin integer t,m,n,it,iI,jI;real e,gz;
        ввод   (t,m,n,e,gz);
         begin real a0,aI;
               array p,q[0:m],d,X[I:m],a[I:t],b[I:m,I:m];
               integer array base[I:m],nam[I:n],str[I:t];
               procedure start;
                   begin integer j,i;it:=0;
                       ввод   (a,str,nam,X);a0:=gz;
```

и далее,как в основном тексте.

7⁰. В этом параграфе приводится программа модифицированного симплекс-метода для задач линейного раскроя.
Постановка задачи такова:
Прутья различных видов (вид прута определяется его длиной), поступающие непрерывно в данном отношении,требуется нарезать на детали так, чтобы детали с указанными длинами создавали комплекты с определенным отношением. Задача состоит в нахождении системы таких технологических раскроев, которая дает минимальные отходы. Технологичным мы называем раскрой, при котором получается не более,чем задано (tI), разных деталей и не более,чем задано (t),всего деталей.

Итак,рассматриваются: m1 видов деталей и m2 видов прутьев. Всего m=m1+m2 элементов.
Заданы длины элементов (c[i] - длина детали i , если i≤ mI , и длина прута с номером i-mI в противном случае) и комплектность элемен.ов (т.е. задано отношение k[I]:k[2]:...:k[mI] выпуска деталей и отношение k[m1+1]:...:k[m] поступления прутьев).

Раскрой $H=(h_0,h_1,\dots,h_K)$ (здесь h_0 - номер прута, $h_1,\dots,h_K$ -номера деталей)допустим, если I) $\sum_{i=1}^{K} c[h_i] \leqslant c[h_0]$; 2) $K \leqslant t$; 3) среди номеров $h_1,\dots,h_K$ – не более чем $t1$ разных. Обозначим через a_{iH} кратность элемента i в раскрое H, т.е.

$$a_{iH}=|\{h_\alpha|(h_\alpha\in H)\wedge(h_\alpha=i)\}|,$$

а через A_H m - мерный вектор ($a_{1H}, a_{2H}, \dots, a_{mH}$). Таким образом, каждому допустимому (технологичному)раскрою $H=(h_0,h_1,\dots,h_K)$ сопоставлен m -компонентный целочисленный вектор A_H, в котором последние $m2$ компоненты составляют (h_0-m1)- й орт ,первые $m1$ компонент дают в сумме не более t, а всего среди них не более чем $t1$ отлично от нуля и выполнено

$$\sum_{i=1}^{m1} c[i]\ a_{iH} \leqslant \sum_{i=m1+1}^{m} c[i]\cdot a_{iH}\ .$$

Верно и обратное: каждый вектор с указанными свойствами (множество таких векторов обозначим через $\mathcal{H}$) определяет допустимый раскрой.

Для определения наилучшей системы раскроев напишем задачу "ного программирования.

пайти числа X_H, $H\in\mathcal{H}$ (X_H -интенсивность использования раскроя H),минимизирующие μ - количество комплектов прутьев, идущих на изготовление одного комплекта деталей при условиях:

$$\sum_H a_{iH}X_H \geqslant K_i \qquad \text{для любого } i\in\mathcal{J}$$

($\mathcal{J}=\{1,\dots,m1\}$ - множество номеров деталей);

$$\sum_H a_{iH}X_H-\mu K_i\leqslant 0 \text{ для любого } i\in\mathcal{J}$$

$\mathcal{J}=\{m1+1,\dots,m\}$ - множество номеров прутьев);

i -тое из первых ограничений требует выпуска i -й детали в количестве, достаточном для одного комплекта.

i -е из вторых ограничений предполагает использование i -го прута в количестве,даваемом не более чем μ комплектами прутьев.

Приведем текст программы:

```
begin integer  it,m,m1,m2,t,t1,i1; real e,gz;
      ввод  (m1,m2,t,t1,e,gz); m:=m1+m2;
      begin real a1,a2; array b[1:m,1:m],d,c,k,X,p[1:m];
            integer array bh[1:m-1,0:t1],h[0:t1],w[1:m],
                          sbh[1:m-1,1:t1],sh[1:t1];
```

```
procedure start;
   begin integer i,j,u;real z,zI,y,yI;ввод (c,k);
         zI:=I/k[m];d[m]:=b[m,m]:=-zI;yI:=0;
         for i:=I step I until mI do
             begin for j:=I step I until m do
                                    b[i,j]:=0;
                   if c[i]>c[m]then
                      begin вывод (i,c);stop end;
                   bh[i,0]:=m;bh[i,I]:=i;
                   if tI > I then bh[i,2]:=0;
                   u:=entier(c[m]/c[i]);w[i]:= i;
                   sbh[i,i]:=u:=if u ≤ t then u else t;
                   b[i,i]:=y:=I/u;X[i]:=k[i]×y;
                   d[i]:=b[m,I]:=z:=y × zI;yI:=yI+X
                   for j:=mI+I step I until m-I do
                       b[j,i]:=z × k[j]
             end;
         y:=X[m]:=zI × yI;w[m]:=m;it:=0;
         for j:=mI+I step I until m-I do
             begin for j:=mI+I step I until m do
                                    b[i,j]:=0;
                   b[j,j]:=I;d[j]:=bh[j,i]:=0;
                   bh[j,0]:=j;X[j]:=y × k[j];
                   b[j,m]:=-zI × k[j];w[j]:=[j]
             end
   end start;
```

comment В начальном плане все детали делаются из последнего (самого длинного !) прута. Базисная матрица начального плана имеет вид:

u_i … u_{mi}	0	0
0	1 … 1	$-K_{mi+1}$ … $-K_{m-1}$
1 . . . 1	0	$-K_m$

Здесь U_i - максимальное количество деталей типа i, которые можно изготовить из одного прута.
Интенсивностями начального плана будут

$$X_1 = \frac{K_1}{U_1}, \ldots, X_{m_1} = \frac{K_{m_1}}{U_{m_1}},$$

$$X_{m_1+1} = \frac{K_{m_1+1}}{K_m} \sum_{j=1}^{m_1} \frac{K_j}{U_j}, \ldots, X_{m-1} = \frac{K_{m-1}}{K_m} \sum_{j=1}^{m_1} \frac{K_j}{U_j},$$

$$X_m = \frac{1}{K_m} \sum_{j=1}^{m_1} \frac{K_j}{U_j}$$

При этом матрица, обратная к базисной, такова:

$\frac{1}{U_1}$ ... $\frac{1}{U_{m_1}}$	0	0
$\frac{K_{m_1+1}}{K_m U_1}$ ... $\frac{K_{m_1+1}}{K_m U_{m_1}}$ ... ... $\frac{K_{m-1}}{K_m U_1}$ ... $\frac{K_{m-1}}{K_m U_{m_1}}$	1 ... 1	$-\frac{K_{m_1+1}}{K_m}$ ... $-\frac{K_{m-1}}{K_m}$
$\frac{1}{K_m U_1}$ ... $\frac{1}{K_m U_{m_1}}$	0	$-\frac{1}{K_m}$

Двойственные переменные совпадают с последней строкой обратной матрицы.
Информация о способах, занятых в базисе, хранится в массивах bh и sbh ; точнее, об i - м способе в i - х строках указанных массивов.
Если $h_0 = bh[i,0] > 0$, то h_0 указывает номер прута ; $h_\alpha = bh[i,\alpha]$ указывает в случае $h_\alpha > 0$ номер детали, а $sbh[i,\alpha]$ - её кратность. Отметим, что в каждой строке массива bh следует рассматривать только элементы, предшествующие первому нулю. В частности, $bh[i,1] = 0$ говорит о том, что прут $bh[i,0]$ остается неиспользованным.

$h_0 < 0$ соответствует производству лишних деталей типа „$-h_0$".

```
procedure new;
   begin integer i,iI,j,jI,s,sI,u,v;real ch,zh,y;
         integer array hI[0:tI],shI,nh[I:tI];aI:=0;
         for i:=1 step I until mI do
             begin if -d[i] > aI then
                       begin aI:=-d[i];u:=-i end
             end;
         for i:=mI+I step I until m do
             begin if d[i]>aI then
                       begin aI:=d[i];u:=i end
             end;
             h[0]:=u;h[I]:=0;s:=2;sI:=mI+2;
1 10:        for i:=mI step -I until s do
                 begin u:=w[i];v:=w[i-I];
                       if d[u]×c[v]>d[v]×c[u] then
                          begin w[i]:=v;w[i-I]:=u end
                 end;
             if s≠2 then go to 1I2;
1 II:        for i:=m step -I until sI do
                 begin u:=w[i];v:=w[i-I];
                       if d[u]×c[v]>d[v]×c[u] then
                          begin w[i]:=v;w[i-I]:=u end
                 end;
             u:=hI[0]:=w[sI-I];j:=jI:=0;iI:=I;
             ch:=c[u];zh:=d[u];
1 12:        u:=w[iI];y:=ch/c[u];
             if y×d[u]+zh≤aI then go to 1 I4;
             if y<1 then go to 1 13 else jI:=jI+I;
             y:=entier(y);j:=j+y;
             nI[jI]:=u;nh[jI]:=iI;
             if j >t then begin y:=y-j+t;j:=t end;
             shI[jI]:=y;ch:=ch-c[u]×y;zh:=zh+d[u]×y;
             if zh>aI then
                begin if jI<tI then h[jI+I]:=0;
                      aI:=zh;h[u]:=hI[0];
                      for i:=I step I until jI do
                          begin h[i]:=hI[i];sh[i]:=shI[i] end
                end;
             if jI=tI then go to 1 I5;if j=t then go to 1 I6;
1 13:        iI:=iI+I;if iI>mI then go to 1 I4;
             if iI≠s then go to 1 I2;s:=s+I;go to 1 I0;
```

```
1 14: if j=0 then
        begin s1:=s1+1;
              go to if s1=m+2 then 1 17 else 1 11
        end;
      if u≠h1[j1] then
        begin u:=h1[j1];i1:=nh[j1];go to 116 end;
1 15: y:=sn1[j1];j:=j-y;j1:=j1-1;
      ch:=ch+c[u]×y;zh:=zh-q[u]×y;go to 1 13;
1 16: j:=j-1;ch:=ch+c[u];zh:=zh-q[u];
      sn1[j1]:=sh1[j1]-1;
      if sh1[j1]=0 then j1:=j1-1;go to 1 13;
1 17: if a1<e then go to 1 2
  end new;

procedure place;
  begin integer i,j,u; real a2,a3;array q[1:m];
        for i:=1 step 1 until m do q[i]:=0;
        u:=h[0];
        if u<0 then
          begin q[-u]:=-1;go to 1 end;
        q[u]:=1;
        for j:=1 step 1 until t1 do
            begin i1:=h[j];
                  if i1=0 then go to 1;
                  q[i1]:=sh[j]
            end;
1:      i1:=0;
        for i:=1 step 1 until m do
            begin p[i]:=0;
                  for j:=1 step 1 until m do
                      p[i]:=p[i]+b[i,j]×q[j];
                  if p[i]>e then
                     begin a3:=x[i]/p[i];
                           if i1=0 ∨ a3<a2 then
                              begin a2:=a3;i1:=i end
                     end
            end
  end place;
```

```
procedure work;
   begin integer i,j;real y;bh[iI,0]:=h[0];
         bh[iI,0]:=h[0];
         for j:=I step I until tI do
             begin bn[iI,j]:=n[j];sbh[iI,j]:=sh[j] end;
         y:=X[iI]:=X[iI]/p[iI];
         for j:=I step I until m do b[iI,j]:=b[iI,j]/p[iI];
         for i:=I step I until m do
             if i≠iI then
                begin X[i]:=X[i]-y× p[i];
                      if X[i]<0 then X[i]:=0;
                      for j:=I step I until m do
                          b[i,j]:=b[i,j]-b[iI,j]×p[i]
                end;
         for j:=I step I until  do d[j]:=b[m,j];
         it:=it+I
   end work;

procedure fin;
   begin integer i,j,k;array q[I:m];
         begin вывод (it,X,bh,sbh,d);
                for i:=I step I until m do q[i]:=0;
                for i:=I step I until m-I do
                    begin k:=bh[i,0];if k<0 then go to 1;
                          q[k]:=X[i]+q[k];
                          for j:=I step I until tI do
                              begin k:=bn[i,j];
                                    if k=0 then go to 1;
                                    q[k]:=q[k]+sbn[i,j]×X[i]
                              end;
1:                  end;
                    вывод (q)
         end
   end fin;

start; l I: new; place; work; go to l I; l 2: fin

end end программы.
```

8^0. До сих пор мы рассматривали симплекс-процедуру, не используя специфику базисной матрицы. Однако для задач больших размеров подобного рода рассмотрения становятся весьма полезными. Мы будем иметь дело с так называемой узкоблочной задачей [3].

Требуется найти X_i, $i \in \mathcal{J}$, такие, что при выполнении

$$\sum_{j \in \mathcal{J}} X_j A_j = B \qquad (B \text{ и } A_j - \text{S- компонентные векторы})$$

и $\sum_{j \in \mathcal{J}_i} X_j \alpha_j = \beta_i$, где $\mathcal{J}_i \cap \mathcal{J}_\kappa = \Lambda$ $(\mathcal{J}_i, \mathcal{J}_\kappa \subset \mathcal{J})$

при любых неравных i, $\kappa = 1, 2, \ldots, t$
переменная X_j принимает минимальное значение.
Симплекс-таблица приведенной задачи имеет вид:

$A_1 \ldots A_{n_0}$	$A_{n_0+1} \ldots A_{n_1}$	$A_{n_1+1} \ldots A_{n_2}$	. . .	$A_{n_{t-1}+1} \ldots A_{n_t}$		B
	$\alpha_{n_0+1} \ldots \alpha_{n_1}$					β_1
		$\alpha_{n_1+1} \ldots \alpha_{n_2}$				β_2
						.
					=	.
						.
				$\alpha_{n_{t-1}+1} \ldots \alpha_{n_t}$		β_t

Здесь

$$\mathcal{J} = \{1, \ldots, n_t\}; \quad \mathcal{J}_1 = \{n_0+1, \ldots, n_1\}; \ldots; \mathcal{J}_t = \{n_{t-1}+1, \ldots, n_t\}.$$

Переменную X_j будем называть переменной κ - ой полосы (κ -го узкого блока), если $j \in \mathcal{J}_\kappa$.
Отберем по одному представителю из каждой полосы и в базисной матрице поместим представителя полосы "κ" на $(S+\kappa)$-ое место.

Тогда базисная матрица будет иметь вид:

$$D = \begin{array}{|c|c|} \hline \delta & \gamma \\ \hline \omega & \alpha \\ \hline \end{array}$$

Здесь α - диагональная матрица. Мы будем помнить только диагональные элементы.

В каждом столбце матрицы ω только один элемент может быть отличен от нуля, поэтому достаточно помнить только его величину и номер строки, в которой он находится.

Обратная матрица базиса такова

$$D^{-1} = \begin{array}{|c|c|} \hline \beta & -\beta\gamma\alpha^{-1} \\ \hline -\alpha^{-1}\omega\beta & \alpha^{-1}+\alpha^{-1}\omega\beta\gamma\alpha^{-1} \\ \hline \end{array}$$

где $\beta = (\delta - \gamma\alpha^{-1}\omega)^{-1}$.

Мы будем хранить в качестве информации о D^{-1} только матрицу β.

Обратная матрица используется для решения двух систем. Первая из них $DP = A$ дает коэффициенты $P = D^{-1}A$ разложения вектора A по базису D.

Ясно, что

$$\left.\begin{aligned} \bar{P} &= \beta(\bar{A} - \gamma\alpha^{-1}\underline{A}), \\ \underline{P} &= \alpha^{-1}(\underline{A} - \omega\bar{P}). \end{aligned}\right\} \quad (*)$$

Здесь первые s координат вектора P образуют вектор $\bar{P}$, а последние t - вектор $\underline{P}$. Аналогично для вектора A.

При вычислениях полезно учесть, что вектор $\underline{A}$ либо нулевой, либо отличен от нулевого одной компонентой (k - ой, если A из k - ой полосы).

Вторая система $dD=c$ дает двойственные переменные $d=cD^{-1}$. Здесь c - вектор базисных цен. Причём $\bar{c}$ - есть sI-й орт (sI - это номер места, которое занимает переменная X_s в базисе), а $\underline{c}=\Phi$. Таким образом:

$$\left.\begin{array}{l}\bar{d}=b_s, \quad \text{(sI - я строка матрицы b)},\\ \underline{d}=-\bar{d}_\gamma \alpha^{-1}.\end{array}\right\}(**)$$

Использование формул (*) и (* *) исключает необходимость восстановления обратной матрицы D^{-1}, а специфика матриц ω и α позволяет легко производить умножение на ω и α^{-1}. Элементы матрицы γ, как и вообще векторов A_j, мы считаем алгорифмически восстанавливаемыми процедурой a(i,j) по номеру строки (i) и столбца (j), на пересечении которых они стоят.

Так же считается описанной процедура max(j,y,d,h,zh,ah), определяющая по двойственным оценкам общих ограничений d и двойственной оценке y полосы j способ h, имеющий максимальную оценку zh среди векторов полосы j. Попутно запоминается элемент ah полосы найденного способа h. Теперь приведем текст программы.

Процедуры a и max приводятся для случая полного задания матрицы ограничений.

```
begin integer m,s,t,n; ввод (s,t,n);m:=s+t;
      begin integer h,it,ii,jI,jh;
            real bar,gz,kz,ah,z,zh;
            array b[1:s,I:s],p,bp,X,d[I:s],
                  aI[I:s+1,I:n];
            integer array nI[0:t],w[I:s],bh[I:m];

comment
      s -количество ограничений общего вида,
      t -количество полос,
      m -всего ограничений,
      n -количество переменных,
      it-счётчик итерации,
      iI-освобождаемое в базисе место,
      jI-рабочая переменная,
      jh-номер полосы вводимого в базис способа,
     bar-барьер,
      gz-большое число ,
      kz-малое число,
```

```
      h   - информация о вводимом  в базис способе,
      ah  - элемент полосы способа  h  ,
      z   - главный элемент,
      zh  - оценка способа  h  ,
      b   - квадрант обратной матрицы,
      p   - коэффициенты разложения способа  h  по базису,
    bp[k] - элемент полосы  k  - го базисного способа,
      X   - план,
      bh  - информация о базисных способах,
      d   - двойственные переменные,
      w   - номера полос основных базисных способов,
      aI  - матрица общих ограничений,
    nI[k] - номер последней переменной полосы k  ;
          -
procedure a(i,j);value i,j;integer i,j;a:=aI[i,j];

procedure max(j,y,d) result:(h,zh,ah);
          integer j,h;real zh,ah,y;array d;
          begin integer i,k,v; real yI;
                v:=if j=0 then I else nI[j-I]+I;zh:=0;
                for k:=nI[j]step -I until v do
                    begin yI:=y × a(s+I,k);
                          for i:=I step I until s do
                              yI:=yI+d[i] × a(i,k);
                    if yI >zh then begin zh:=yI;h:=k;
                                         ah:=a(s+I,h)
                                   end
                    end
          end max;

procedure start;
          begin integer i,j; ВВОД (X,aI,nI);
                it:=0; ВВОД (kz,gz);bar:=gz;jI:=t;
                for i:=I step I until s do
                    begin for j:=I step I until s do b[i,j]:=0;
                          b[i,i]:=I;d[i]:=gz;bp[i]:=0;
                          w[i]:=bh[i]:=0
                    end;
               for i:=s+I step I until m do
                   begin bp[i]:=I; bh[i]:=0 end;
          end start;
```

```
procedure new;
   begin integer hI,i,j,u,k; real y,z,ahI; zh:=0;j:=jI;
1 I0:    j:=if j=t tnen 0 else j+I; y:=0;
         if j≠0 then begin
            begin k:=s+j;u:=bh[k];
                  if u=0 then y:=gz
                         else begin for i:=I step I until s do
                                          y:=y-d[i]×a(i,u);
                                    y:=y/bp[k]
                              end
            end;
         max(j,y,d,hl,z,ahI);
         if z>zh then begin zh:=z;h:=hI;jh:=j;ah:=ahI end;
        if z>bar then go to 1 20; if j≠jI then go to 1 I0;
         if zh<kz then go to 1 I2;bar:=zh/2;
120: end new;
procedure place;
   begin Boolean net;integer i,j,u;real y,yI;
         for i:=I step I until m do p[i]:=0; u:=s+jh;y:=0;
         if jh>0 then y:=p[u]:=ah/bp[u]; net:=true;
         for j:=1 step 1 until s do
              begin y1:=a(j,h); i:=bh[u];
                    if i≠0 then y1:=y1-y×a(j,i);
                    for i:=1 step 1 until s do
                          p[i]:=p[i]+b[i,j]×y1
              end;
         for i:=1 step 1 until s do
            begin u:=w[i];
                  if u>0 then
                     p[s+u]:=p[s+u]-p[i]×bp[i]/bp[s+u]
            end;
         for i:=1 step 1 until m do if p[i]>kz then
             begin y:=X[i]/p[i];
                   if net then begin z:=y; i1:=i;
                                     net:=false    end
                          else if y<z  then
                                    begin z:=y; i1:=i end
             end;
         if net then begin  вывод (h,jh,X,bh); stop end
   end place;
```

```
procedure work;
  begin Boolean da; integer i,j,u,i2,j2;
        real y,y1,; array q[1:s];
        it:=it+1; i2:=i1; y1:=y:=1/p[i1]; da:=false;
        for i:=1 step 1 until m do
            begin X[i]:=X[i]-z × p[i]; if X[i]< 0 then X[i]:=
            end;
        if i1 ≤ s
           then for i:=1 step 1 until s do q[i]:=b[i1,i]
           else begin j2:=i1-s; da:=true;
                      for i:=1 step 1 until s do q[i]:=0;
                      for i:=1 step 1 until s do
                      if w[i]=j2 then
                         begin u:=i;
                               for j:=1 step 1 until s do
                                   q[j]:=q[j]-b[i,j]× bp[i]
                         end;
                      y1:=y/bp[i1];
                      if jh≠j2 then
                      begin X[i1]:=X[i2];i2:=u;da:=false;
                           bp[i1]:=bp[i2];bh[i1]:=bh[i2];
                      end
                end;
        for i:=1 step 1 until s do q[i]:= q[i]× y1;
        for i:=1 step 1 until s do
            begin d[i]:=d[i]-q[i] × zh;
                  if p[i]≠0 ∧ i≠i2 then
                      for j:=1 step 1 until s do
                          b[i,j]:=b[i,j] -p[i] × q[j]
            end;
        bh[i2]:=h; if da then go to 1; w[i2]:=jh;
        for i:=1 step 1 until s do b[i2,i]:=q[i];
     1: X[i2]:=z; bp[i2]:=ah
  end work;

procedure fin; ВЫВОД (X,bh);

start; l1:new; place; work; go to l1;l2:fin

end  end     программы
```

Литература

I. Дж.Данциг. Линейное программирование.Его применение и обобщения. М. Издательство "Прогресс", 1966.

2. Сб. "Линейные неравенства и смежные вопросы". М.ИЛ.1959.

3. Р.А. Звягина . Программа реализации на„М-20" модифицированного метода с узкоблочной матрицей .В сб. "Оптимальное планирование".Вып. 4. Новосибирск.1966.

Поступила в редакцию
27.XI.1967 г.