

ПРОГРАММА АЛГОРИТМА РЕШЕНИЯ ЗАДАЧИ ВЫБОРА ОПТИМАЛЬНОГО НАБОРА

В.Л.Фереснев, Г.М.Заикина

В работе [1] описана общая схема алгоритма типа неявного перебора для решения так называемой задачи выбора оптимального набора

$$\min_{x_i \in \{0,1\}} S(x_1, \dots, x_m),$$

$$S(x_1, \dots, x_m) = \sum_{i=1}^m (1-x_i) g_i^0 + \sum_{j=1}^n \varphi_j \min_{i/x_i=0} g_{ij},$$

где $G = \{g_{ij}\}$, $i = 1, \dots, m$; $j = 1, \dots, n$ - матрица, а (φ_j) , $j = 1, \dots, n$; (g_i^0) , $i = 1, \dots, m$, - неотрицательные векторы.

Настоящая работа посвящена описанию машинной программы, реализующей ту часть алгоритма, в которой применяется схема одностороннего ветвления [1].

Напомним используемые в алгоритме способ задания подмножеств решений, способ вычисления нижней границы и способ ветвления.

Исследуемые на каждом шаге алгоритма множество решений и множество непросмотренных решений задаются частичными векторами (y_1, \dots, y_q) , (z_1, \dots, z_q) , $q < m$, $y_i \in \{0,1\}$, $z_i \in \{1,2\}$. Причем исследуемым множеством является множество продолжений

$P(y_1, \dots, y_q)$ частичного решения (y_1, \dots, y_q) , а множество непросмотренных решений - объединение $P(y_1, \dots, y_q)$ и множеств $P(y_1, \dots, y_{p-1}, 1-y_p)$, где $p \leq q$, $z_p = 1$.

В качестве нижней границы функции $S(x_1, \dots, x_m)$ на множестве $P(y_1, \dots, y_q)$ используется величина

$$b(y_1, \dots, y_q) = \sum_{i=1}^q (1-y_i) g_i^0 + \sum_{j=1}^n \min_{i=1, \dots, m} \{ \varphi_j g_{ij} + w_{ij} \}.$$

Здесь $W = (w_{ij})$, $w_{ij} \geq 0$, $i = 1, \dots, m$; $j = 1, \dots, n$, - матрица, удовлетворяющая условиям:

$$\sum_{j=1}^n w_{ij} \leq g_i, \quad i = 1, \dots, m,$$

где

$$g_i = \begin{cases} 0, & \text{если } i \leq q \text{ и } y_i = 0, \\ \infty, & \text{если } i \leq q \text{ и } y_i = 1, \\ g_i^0, & \text{если } i > q. \end{cases}$$

Кроме того, будем считать, что для каждого столбца j матрицы W выполняется:

- 1). если $J_j = \{i / \varphi_j g_{ij} + w_{ij} = \min_{k=1, \dots, m} \{\varphi_j g_{kj} + w_{kj}\}, \text{ то } w_{ij} = 0$
при $i \notin J_j$,
- 2). $\min_{i \in J_j} \{g_i - \sum_{l=1}^n w_{il}\} = 0$.

Матрица W , удовлетворяющая условиям 1,2, называется тупиковой и дает в некотором смысле наилучшую [3] нижнюю границу.

Иногда при помощи тупиковой матрицы W может быть найден оптимальный вектор функции $S(x_1, \dots, x_m)$ на множестве $P(y_1, \dots, y_q)$. В самом деле, пусть $J = \{i / g_i - \sum_{j=1}^n w_{ij} = 0\}$ и пусть для всякого $j, j = 1, \dots, n$, число элементов в множестве $\{i \in J / w_{ij} > 0\}$ не превосходит единицы. Тогда $\forall y_1, \dots, y_q, S(y_1, \dots, y_q, x_{q+1}, \dots, x_m)$, где $x_i = 0$ при $i \in J$ и $x_i = 1$ при $i \notin J$, и, следовательно, $(y_1, \dots, y_q, x_{q+1}, \dots, x_m)$ - оптимальный вектор на множестве $P(y_1, \dots, y_q)$.

Таким образом, можно предположить, что компоненты оптимального вектора с номерами из множества J равны 0.

Это свойство элементов множества J используем для уменьшения размерности задачи. В самом деле, вычислим при $q=0$ нижнюю границу b_0 и множество J и далее выбор нулевых компонент оптимального вектора будем производить лишь среди элементов множества J . Такая предварительная отбраковка позволяет существенно уменьшить время решения задачи, что особенно важно в случае, когда число m велико. Однако при этом мы можем рассчитывать на получение лишь приближенного решения (x'_1, \dots, x'_m) задачи, но с оценкой $S(x'_1, \dots, x'_m) - b_0$ для величины возможного отклонения полученного решения от оптимального. В частности, если $S(x'_1, \dots, x'_m) = b_0$, то найденное решение (x'_1, \dots, x'_m) является оптимальным.

На указанном свойстве элементов множества J основан и способ ветвления. Пусть исследуемое множество и множество непросмотренных решений задаются векторами $(y_1, \dots, y_q), (z_1, \dots, z_q)$ и пусть ρ - наименьший элемент множества J . Тогда множество, полученное в результате ветвления, и множество непросмотренных решений будем задавать векторами $(y_1, \dots, y_q), (z_1, \dots, z_q)$, где $y_i = 1$ при $q < i < \rho$; $y_\rho = 0$; $z_i = 1$ при $q < i < \rho$.

Если исследуемое множество выбрасывается из рассмотрения, то множество непросмотренных решений и исследуемое на следующем шаге множество решений будем задавать частичными решениями $(y_1, \dots, y_{\rho-1}, 1 - y_\rho), (z_1, \dots, z_{\rho-1}, 2)$, где $\rho, 1 \leq \rho \leq q$, наибольший номер, для которого $z_\rho = 1$. Если же такого номера не существует, т.е. $z_i \neq 1$ для всех $i, 1 \leq i \leq q$, то множество непросмот-

ренных решений пусто и алгоритм заканчивает работу.

Перейдем к описанию программы алгоритма. При этом будем считать известным разбиение множества $\{1, 2, \dots, m\}$ на K групп: $\{m_k, \dots, m_{k+1} - 1\}$, $k = 1, \dots, K$; $m_1 = 1$; $m_{K+1} = m + 1$, и предполагать, что у оптимального вектора не более одной компоненты с номерами из одной группы равняются нулю. В частности, можно считать, что $K = m$ и в каждой группе содержится ровно один элемент.

Программа предполагает задание следующих числовых данных:

- 1-2) числа m , n - количество строк и столбцов матрицы G ;
- 3) число N - количество существенных элементов матрицы G , т.е. таких элементов, которые не равняются ∞ ;
- 4) массив $g[1:m]$ задает вектор (g_i^0) и разбиение множества $\{1, 2, \dots, m\}$ на группы. Элемент массива с номером i имеет знак "-", если номер i не является началом какой-либо группы;
- 5) массив $\varphi[1:n]$ задает вектор (φ_j) ;
- 6-7) массивы $g[1:N]$, $a[1:N]$ задают матрицу G . Существенные элементы матрицы, записанные подряд по столбцам, составляют массив $g[1:N]$, а каждый элемент массива $a[1:N]$ задает номер строки матрицы, в которой расположен соответствующий элемент массива $g[1:N]$. При этом первый существенный элемент каждого столбца матрицы G в массиве $a[1:N]$ помечен знаком "-".

8) Массив $R[1:2]$, $R[1] \in \{0, 1\}$, $R[2] \geq 1$, задает параметры алгоритма. Если $R[1] = 0$, то программа работает с блоком предварительной отбраковки, если $R[1] = 1$, то без этого блока. Параметр $R[2]$ используется для искусственного увеличения нижней границы. Это позволяет за сравнительно меньшее время получить приближенное решение задачи. Причем значение целевой функции для дальнейшего решения будет отличаться от оптимального значения не более чем в $R[2]$ раз. В частности, при $R[2] = 1$ отыскивается оптимальное решение задачи.

В результате работы алгоритма вычисляются:

- 1) число B - значение целевой функции на оптимальном или приближенном решении (x_1, \dots, x_m) ;
- 2) массив $y[1:m]$ - номера нулевых компонент оптимального или приближенного решения (x_1, \dots, x_m) ;
- 3) число $B - b_0$ - оценка для величины возможного отклонения значения целевой функции на решении (x_1, \dots, x_m) от оптимального;
- 4) число cx - счетчик числа просматриваемых в процессе работы алгоритма подмножеств.

Ниже приводится программа алгоритма, написанная на языке АЛГОЛ.

```

BEGIN_INTEGER_Q, Q, I, J, L, R,  $\bar{R}$ , H, M, N, PP,  $\bar{N}$ , K, II, C4;
  REAL_B,  $\bar{B}$ ,  $\bar{F}$ ,  $\bar{H}$ ,  $\bar{G}$ ,  $\bar{A}$ , D, B0;
  INPUT(M, N,  $\bar{N}$ ); D := 10 * 10;
BEGIN_ARRAY_GO, X, Z, Y, P, GG[1:M], НАЧ[1:(N+1)],
  G, A[1:N],  $\Phi$ ,  $\bar{Q}$ , Q, Q1, I1[1:N],  $\bar{R}$ [1:2];
PROCEDURE_ГРАНИЦА:

```

COMMENT Процедура начинается с проверки, содержит ли множество $P(y_1, \dots, y_q)$ оптимальное решение. Для этого, во-первых, проверяется, содержит ли частичное решение (y_1, \dots, y_q) две нулевые компоненты из одной группы, и, во-вторых, осуществляется проверка, аналогичная проверке в методе последовательных расчетов [4]. Если в результате этих проверок удается установить, что множество $P(y_1, \dots, y_q)$ не содержит оптимального вектора, то полагаем нижнюю границу b равной 10^{10} . В противном случае начинаем процесс вычисления нижней границы b функции $S(x_1, \dots, x_n)$ на множестве $P(y_1, \dots, y_q)$;

```

BEGIN_IF_QQ=0∨Q≤1 THEN_GOTO_M1;
  IF_Y[Q]=0∨GO[Q]>0 THEN_GOTO_M1;
  I:=Q-1;
MO: IF_Y[I]=0∧Z[I]>0 THEN
  BEGIN_B:=0; GOTO_MKEND;
  IF_GO[I]<0 THEN
  BEGIN_I:=I-1; GOTO_MOEND;

```

COMMENT Компонента массива Z помечается знаком "-", если соответствующий элемент множества $\{1, 2, \dots, n\}$ выбрасывается из рассмотрения при предварительной отбраковке;

```

M1: FOR_L:=1 STEP_1 UNTIL_NDO
  BEGIN_QQ[L]:=-D; I1[L]:=-НАЧ[L]-1;
  I:=НАЧ[L];
M2: IF_I<НАЧ[L+1] THEN
  BEGIN_R:=A[I];
  IF_R<Q∧Y[R]=Q∧G[I]< $\bar{Q}$ [L] THEN
   $\bar{Q}$ [L]:=G[I];
  IF_R≤Q∧Z[I]>0 THEN
  I1[L]:=I; I:=I+1; GOTO_M2END;
END;

```

```

IF_Q=0VY[Q]=0THEN_GOTO_M3;
F:=ABS(GO[Q]);
FOR_J:=1STEP_1UNTIL_NDO_
IF_Q=A[I1[J]]THEN_
BEGIN_IF_QO[J]>G[I1[J]]THEN_
    BEGIN_F:=F-QO[J]+G[I1[J]];
    QO[J]:=G[I1[J]]END_
END_;
IF_F>0THEN_
BEGIN_B:=D;GOTO_MKEND_

```

COMMENT Далее начинается итеративный процесс вычисления массивов $\bar{Q}[l]$, $q_1[l]$, $l = 1, \dots, n$, где $\bar{Q}[l]$ - величина минимального элемента столбца l матрицы $(g_{ij} + w_{ij})$, а $q_1[l]$ - число минимальных элементов в столбце l . На каждой итерации для номера l с наименьшим значением $q_1[l]$ происходят, во-первых, увеличение $\bar{Q}[l]$ на некоторую величину A и, во-вторых, пересчет значения $q_1[l]$;

```

M3:   FOR_I:=Q+1STEP_1UNTIL_MDO_
      GG[I]:=ABS(GO[I]);
      FOR_L:=1STEP_1UNTIL_NDO_
      BEGIN_Q[L]:=D;
        FOR_I:=I1[L]+1STEP_1UNTIL_HA9[L+1]-1DO_
          IF_G[I]<Q[L]^Z[A[I]]>0THEN...
          Q[L]:=G[I];
      Q1[L]:=0;
      FOR_I:=I1[L]+1STEP_1UNTIL_HA9[L+1]-1DO_
        IF_G[I]=Q[L]^Z[A[I]]>0THEN_
          Q1[L]:=Q1[L]+1;
        IF_Q[L]>QO[L]THEN_
          BEGIN_Q[L]:=QO[L];Q1[L]:=M+1END_
      END_;

```

```

M4:      R:=D;
          FOR L:=1 STEP 1 UNTIL NDO
            IF L>Q1[L] THEN R:=Q1[L];
          R:=0;
          FOR L:=1 STEP 1 UNTIL NDO
            IF L=Q1[L] THEN R:=R+1;
            IF L>M THEN GOTO M10;

M7:      L:=1;
M8:      IF Q1[L]=R THEN GOTO M5;
M6:      IF L=N THEN GOTO M7; L:=L+1; GOTO M8;
M5:      I:=I1[L]; G:=D; H:=Q0[L];
          FOR K:=I+1 STEP 1 UNTIL HA4[L+1]-1 DO
            IF L2[A[K]]>0 THEN
              BEGIN IF G[K]≤Q[L] THEN
                G:=MIN(G, GG[A[K]]) ELSE
                H:=MIN(H, G[K]) END;
              A:=MIN(G, H-Q[L]);

          IF A=0 THEN
            BEGIN Q1[L]:=M+1; GOTO M9 END;
          I:=I1[L]; H:=0;
          FOR K:=I+1 STEP 1 UNTIL HA4[L+1]-1 DO
            IF L2[A[K]]>0 THEN
              BEGIN IF G[K]≤Q[L] THEN
                GG[A[K]]:=GG[A[K]]-A ELSE
                BEGIN IF G[K]≤Q[L]+A THEN
                  H:=H+1 END;
              END; Q[L]:=Q[L]+A;
              IF A=G+Q[L]>Q0[L] THEN
                BEGIN Q1[L]:=M+1; GOTO M9 END;
              IF L=0 THEN GOTO M6; Q1[L]:=Q1[L]+H;

```

```

M9:   R:=R-1;
      IF R=0 THEN GOTO M4;
      GOTO M6;
M10:  B:=0;
      FOR I:=1 STEP 1 UNTIL QDO
      IF V[I]=0 THEN B:=B+ABS(GO[I]);
      FOR L:=1 STEP 1 UNTIL NDO
      B:=B+Q[L]; PP:=0;
      FOR I:=Q+1 STEP 1 UNTIL MDO
      IF CG[I]=0 THEN
      BEGIN PP:=PP+1; P[PP]:=IEND;

```

COMMENT На этом процесс вычисления нижней границы заканчивается. Далее проверяем, можно ли с помощью неявным образом построенной тупиковой матрицы W найти оптимальный вектор функции $S(x_1, \dots, x_m)$ на множестве $P(y_1, \dots, y_q)$;

```

      IF PP<1 THEN GOTO MK;
      IF PP>4 THEN
      BEGIN B:=RO[2]*B; GOTO MKEND;
      FOR L:=1 STEP 1 UNTIL NDO
      BEGIN II:=0;
        FOR I:=I1[L]+1 STEP 1 UNTIL HAЧ[L+1]-1 DO
        IF CG[A[I]]-OAG[I]<Q[L] THEN
        BEGIN II:=II+1;
          IF II=2 THEN
          BEGIN B:=B*RO[2]; GOTO MKEND
        END
      END;
      PP:=-PP;
MK:   C4:=C4+1 END ПРОЦЕДУРЫ;
      INPUT(GO, Φ, G, A, RO); C4:=0;
      OUTPUT('T', 'G=', 'E', G, '/'. 'T', 'A=', 'E', A, '/'. 'E', GO,
        Φ, RO, '/');
      I:=1; J:=1;
M11:  IF A[I]<0 THEN
      BEGIN HAЧ[J]:=I; J:=J+1; A[I]:=-A[I] END;
      G[I]:=G[I]*Φ[J-1]; I:=I+1;
      IF I<N THEN GOTO M11;
      HAЧ[N+1]:=N+1;

```

COMMENT Далее следует программа основной части алгоритма:
процесс ветвления;

```

      Q:=0;QQ:=0;B:=D;
      FOR I:=1 STEP 1 UNTIL MDQ
      Z[I]:=1;
      ГРАНИЦА;
      BO:=B;
      IF RO[1]=0 THEN GOTO MO1;
      II:=1;
      FOR I:=1 STEP 1 UNTIL MDQ
      IF I=P[II] THEN II:=II+1 ELSE Z[I]:=-Z[I];
MO1:   IF B<BO GOTO MO2;

MO5:   FOR I:=Q STEP 1 UNTIL 1 DO
      IF Z[I]=1 THEN
      BEGIN Q:=I; GOTO MO3 END;
      GOTO KOH;
MO3:   Y[Q]:=-Y[Q]; Z[Q]:=-Z[Q]; QQ:=1;
      ГРАНИЦА; GOTO MO1;
MO2:   IF PP>1 THEN GOTO MO4;
      B:=B;
      FOR I:=1 STEP 1 UNTIL Q DO
      X[I]:=-Y[I];
      FOR I:=Q+1 STEP 1 UNTIL MDQ
      X[I]:=1;
      IF PP=1 THEN X[P[1]]:=0;
      IF PP<0 THEN
      BEGIN FOR I:=1 STEP 1 UNTIL -PP DO
      X[P[I]]:=0 END;
      GOTO MO5;
MO4:   FOR I:=Q+1 STEP 1 UNTIL P[1] DO
      BEGIN Y[I]:=1;
      IF Z[I]>0 THEN Z[I]:=-1 END;
      Y[P[1]]:=0; Q:=P[1]; QQ:=0;
      ГРАНИЦА;
      GOTO MO1;
KOH:   BO:=B-BO;
      OUTPUT('T', 'B', 'E', B, '/');
      K:=1;
      FOR I:=1 STEP 1 UNTIL MDQ
      IF X[I]=0 THEN

```



```
BEGIN_Y[K]:=I;K:=K+1ENDL;  
OUTPUT('T','НАБОР=','E',Y,'/',  
       'T','B-B0=','E',B0,'/',  
       'T','CЧ=','E',CЧ)
```

ENDL

ENDL

Поступила в ред.-изд.отдел
6 октября 1975 г.

Л и т е р а т у р а

1. Береснев В.Л. Алгоритм неявного перебора для задачи типа размещения и стандартизации. - В кн.: Управляемые системы. Вып. 12 Новосибирск, 1974, с.24-34.

2. Романовский И.В. Методы неявного перебора для решения задач целочисленного программирования с бивалентными переменными. - "Изв. вузов. Матем", 1970, №4, с.17-29.

3. Гимади Э.Х., Глебов Н.И., Дементьев В.Т. Об одном методе построения нижней оценки и приближенного решения с апостериорной оценкой точности для задачи стандартизации. - В кн.: Управляемые системы. Вып. 13, Новосибирск, 1974, с.26-31.

4. Корбут А.А., Финкельштейн Ю.Ю. Дискретное программирование. М., "Наука", 1969.

5. Лавров С.С. Универсальный язык программирования, М., "Наука", 1967.