ZASTOSOWANIA MATEMATYKI
APPLICATIONES MATHEMATICAE
XVI, 1 (1977)

ALGORITHM 51

S. ZIELIŃSKI (Gdańsk)

SOLUTION OF LINEAR PROGRAMMING PROBLEMS IN ZERO-ONE VARIABLES

1. Procedure declaration. The procedure *PLB* solves the following linear zero-one programming problem:

Find the maximum of the objective function

$$f = a_{01}x_1 + a_{02}x_2 + \ldots + a_{0n}x_n$$

under the constraints

$$a_{i1}x_1 + a_{i2}x_2 + \ldots + a_{in}x_n \geqslant b_i \quad (i = 1, 2, \ldots, m),$$

$$x_j = 0 \text{ or } 1 \quad (j = 1, 2, \ldots, n),$$

where a_{ij} and b_i are given integer coefficients.

Data:

n — number of variables;

m — number of constraints;

$a[0:m, 1:n]$ — integer array containing the coefficients of the objective function in row number *0* and the coefficients of the constraints in rows from *1* to m;

$b[0:m]$ — integer array containing in $b[1]$ to $b[m]$ the right-hand sides of the constraints ($b[0]$ is arbitrary).

Results:

e — Boolean variable with the value **true** if an optimal solution to the problem exists and with the value **false** if there is no feasible solution;

$x[1:n]$ — optimal solution (if $e \equiv$ **true**);

f — maximum value of the objective function (if $e \equiv$ **false**);

d — number of iterations performed during the call of the procedure.

Arrays a and b are unchanged on return from the procedure.

2. Method used. The algorithm employed in the procedure *PLB* agrees generally with the method presented in papers [2]-[4]. However,

```
procedure PLB(n,m,a,b,x,e,d,f);
 value n,m;
 integer n,m,d,f;
 Boolean e;
 integer array a,b,x;
 begin
  integer c,g,h,i,j,k,l,r,t,v,y,z;
  integer array w[0:m],p,q,s[1:n];
  Boolean array u[0:m];
  procedure qq;
   begin
    k:=s[j];
    h:=h+1;
    s[j]:=s[h];
    s[h]:=k;
    if a[i,k]≤0
     then q[k]:=0
     else
     begin
      q[k]:=1;
      for v:=0 step 1 until m do
       b[v]:=b[v]-a[v,k]
     end a[i,k]>0
   end qq;
  f:=-1;
  for j:=1 step 1 until n do
   begin
    v:=a[0,j];
    if v<0
     then f:=f+v;
    s[j]:=j
   end j;
  b[0]:=f;
```

```
  e:=false;
  d:=g:=h:=t:=0;
l1:
  for i:=0 step 1 until m do
   u[i]:=false;
l2:
  d:=d+1;
  k:=1;
  v:=h+1;
  for i:=0 step 1 until m do
   begin
    r:=y:=0;
    for j:=v step 1 until n do
     begin
      z:=a[i,s[j]];
      if z>0
       then y:=y+z
       else r:=r+z
     end j;
    y:=w[i]:=y-b[i];
    if y<0
     then go to l7;
    if b[i]≤r
     then u[i]:=true
     else k:=0
   end i;
  if k=1
   then go to l6;
  for i:=0 step 1 until m do
   begin
    if u[i]
     then go to l5;
    v:=w[i];
```

```
        if v>0
         then
         begin
          j:=h;
   13:    j:=j+1;
          if j>n
           then go to 15;
          z:=a[i,s[j]];
          if z=0
           then go to 13;
          if z<0
           then z:=-z;
          r:=z;
          for k:=j+1 step 1 until n do
           begin
            y:=a[i,s[k]];
            if y≠0
             then
             begin
              if y<0
               then y:=-y;
              if y>z
               then
               begin
                z:=y;
                j:=k
               end y>z
               else
                if y<r
                 then r:=y
             end y≠0
           end k;
```

```
        if v<r
         then go to 14;
        if v≥z
         then go to 15;
        qq
       end v>0
       else
14:    begin
        z:=h;
        for j:=h+1 step 1 until n do
         if a[i,s[j]]≠0
          then qq;
        if z=h
         then go to 15
       end v≤0;
      go to 12;
15:end i;
   j:=h+1;
   z:=0;
   for i:=j step 1 until n do
    begin
     r:=a[0,s[i]];
     if r<0
      then r:=-r;
     if z<r
      then
      begin
       z:=r;
       j:=i
      end z<r
    end i;
   i:=0;
   qq;
```

```
   g:=g+1;
   p[g]:=h;
   go to 12;
16:
   i:=0;
   for j:=h+1 step 1 until n do
    qq;
   f:=l:=0;
   c:=n;
   t:=g;
   for j:=1 step 1 until n do
    begin
     x[j]:=y:=q[j];
     if y=1
      then f:=f+a[0,j]
    end j;
   b[0]:=1;
   e:=true;
17:
   if g=0
    then go to 18;
   v:=p[g];
   g:=g-1;
   if t=g+1
    then
    begin
     t:=t-1;
     for j:=v+1 step 1 until c do
      begin
       k:=s[j];
       r:=a[0,k];
       y:=x[k];
```

```
        if y=0∧r>0∨y=1∧r<0
         then l:=l+abs(r)
       end j;
      c:=v;
      if abs(a[0,s[v]])>l
       then go to 17
     end t=g+1;
    for j:=v step 1 until h do
     begin
      k:=s[j];
      if q[k]=1
       then
       for i:=0 step 1 until m do
        b[i]:=b[i]+a[i,k];
      if j=v
       then
       begin
        y:=q[k]:=1-q[k];
        if y=1
         then
         for i:=0 step 1 until m do
          b[i]:=b[i]-a[i,k]
       end j=v
     end j;
    h:=v;
    t:=0;
    go to 11;
  18:
   end PLB
```

the code of the procedure shows many differences when compared with the code of the Fortran subroutine given in [3].

3. Certification. The procedure *PLB* has been verified on both the K-202 and the ICL System 4 computers. This procedure shows improvements in saving both computer memory space and processor time as compared with Balas' procedure [1]. Table 1 contains the results of com-

TABLE 1. Results of comparison

No.	n	m	Balas' method		*PLB*	
			I	T	I	T
1	25	5	1482	21	1500	18
2	25	15	4814	144	1032	33
3	25	15	6302	150	5502	141
4	15	15	900	18	503	12
5	15	25	592	18	408	15

I — number of iterations, T — execution time in seconds.

parison of the algorithm *PLB* with the Balas method obtained on the ICL System 4 (programmed in Algol). The gain in processor time is particularly significant.

References

[1] Z. Cylkowski and J. Kucharczyk, *Algorithm 3: Solution of zero-one integer linear programming problems by Balas' method*, Zastosow. Matem. 11 (1969), p. 111-116.

[2] F. Fiala, *Computational experiences with a modification of an algorithm by Hammer and Rudeanu for 0-1 linear programming*, Proc. Nat. Conf. ACM (1971), p. 482-488.

[3] — *Algorithm 449: Solution of linear programming problems in 0-1 variables*, Comm. ACM 16 (1973), p. 445-447.

[4] P. L. Hammer and S. Rudeanu, *Pseudo-Boolean programming*, Operat. Res. 17 (1969), p. 233-261.

SHIP RESEARCH INSTITUTE
TECHNICAL UNIVERSITY
80-952 GDAŃSK

Received on 3. 8. 1975

ALGORYTM 51

S. ZIELIŃSKI (Gdańsk)

ROZWIĄZYWANIE ZADAŃ PROGRAMOWANIA LINIOWEGO ZE ZMIENNYMI ZERO-JEDYNKOWYMI

STRESZCZENIE

Procedura *PLB*, która jest ulepszoną wersją algolowską procedury przedstawionej w [3], służy do rozwiązywania zadań programowania liniowego ze zmiennymi zero-jedynkowymi.

Dane:

n — liczba zmiennych;
m — liczba ograniczeń;
$a[0:m, 1:n]$ — tablica zawierająca w wierszu zerowym współczynniki funkcji celu, a w pozostałych m wierszach — współczynniki ograniczeń;
$b[0:m]$ — tablica, której zerowy element jest dowolny, a pozostałe elementy są równe prawym stronom ograniczeń.

Wyniki:

e — równe jest **true**, jeżeli istnieje rozwiązanie optymalne, i równa się **false**, jeżeli nie ma żadnego rozwiązania dopuszczalnego;
$x[1:n]$ — rozwiązanie optymalne (jeżeli $e \equiv$ **true**);
f — maksymalna wartość funkcji celu (jeżeli $e \equiv$ **true**);
d — liczba wykonanych iteracji.

Zakłada się, że współczynniki ograniczeń i wartości prawych stron ograniczeń oraz współczynniki funkcji celu są liczbami całkowitymi.