ZASTOSOWANIA MATEMATYKI
APPLICATIONES MATHEMATICAE
XI, 1 (1969)

ALGORITHM 3

Z. CYLKOWSKI and J. KUCHARCZYK (Wrocław)

SOLUTION OF ZERO-ONE INTEGER LINEAR PROGRAMMING PROBLEMS BY BALAS' METHOD

1. Procedure declaration.

```
procedure Balas (m, n, a, b, c, x, Fval, exist, max);
  value m, n, max;
  integer m, n, Fval, max;
  Boolean exist;
  integer array a, b, c, x;
  comment Balas solves the zero-one integer linear programming problem:
    cx = min subject to ax ≤ b, where c ≥ 0 and x = 0, 1.
    Data:
    m — number of constraints,
    n — number of variables,
    a[1:m, 1:n] — coefficient matrix of the constraints,
    b[1:m] — right sides of the constraints,
    c[1:n] — coefficients of the objective function.
    Results:
    exist — true if an optimum solution exists and false if there is no
    feasible solution,
    x[1:n] — optimum solution (if exist = true only, otherwise the pro-
    cedure Balas does not change the table x),
    Fval — the value of the objective function (if exist = true only,
    otherwise Fval = 2×(1+c[1]+c[2]+...+c[n])).
    Other parameters:
    max — maximum positive number of type integer;
  begin
    integer p, mnr, i, gamma, alfa, beta, j, z, t, nr, s, rob, rob1, rob2;
    integer array w, y, zr[1:m], xx, jj, ii[1:n], kk[1:n+1];
    for i := 1 step 1 until m do y[i] := b[i];
    z := 1;
```

```
    for j := 1 step 1 until n do
      begin
        xx[j] := 0;
        z := z+c[j]
      end j;
    Fval := z+z;
    s := t := kk[1] := z := 0;
    exist := false;
beg:p := mnr := 0;
    for i := 1 step 1 until m do
      begin
        rob := y[i];
        if rob < 0
          then
          begin
            p := p+1;
            gamma := 0;
            alfa := rob;
            beta := —max;
            for j := 1 step 1 until n do
              if xx[j] ≤ 0
                then begin
                       if c[j]+z ≥ Fval
                         then begin
                                xx[j] := 2;
                                kk[s+1] := kk[s+1]+1;
                                t := t+1;
                                jj[t] := j
                              end c[j]+z ≥ Fval
                         else begin
                                rob1 := a[i,j];
                                if rob1 < 0
                                  then begin
                                         alfa := alfa — rob1;
                                         gamma := gamma+c[j];
                                         if beta < rob1 then
                                           beta := rob1
                                       end a[i,j] < 0
                              end c[j]+z < Fval
                     end xx[j] ≤ 0, j;
```

```
if alfa < 0 then go to backtr;
if alfa+beta < 0
 then
 begin
   if gamma+z ⩾ Fval then go to backtr;
   for j := 1 step 1 until n do
     begin
       rob1 := a[i, j];
       rob2 := xx[j];
       if rob1 < 0
        then
        begin
          if rob2 = 0
            then
            begin
              xx[j] := -2;
              for nr := 1 step 1 until mnr do
               begin
                 zr[nr] :=zr[nr]-a[w [nr], j];
                 if zr[nr]< 0 then go to backtr
               end nr
            end rob2 = 0
       end a[i, j] < 0 ∧ xx[j] = 0
       else if rob2 < 0
         then
         begin
          alfa := alfa-rob1;
          if alfa < 0 then go to backtr;
          gamma := gamma+c[j];
          if gamma+z ⩾ Fval then go to backtr
         end a[i, j] ⩾ 0 ∧ xx[j] < 0;
     end j;
```

```
                              mnr := mnr+1;
                              w[mnr] := i;
                              zr[mnr] := alfa
                            end alfa+beta < 0

                 end y[i] < 0
      end i;
   if p = 0 then go to sol;
   if mnr = 0
     then
     begin
       p := 0;
       gamma := -max;
       for j := 1 step 1 until n do
         if xx[j] = 0

           then
           begin
             beta := 0;
             for i := 1 step 1 until m do

               begin
                 rob := y[i];
                 rob1 := a[i,j];
                 if rob < rob1 then beta := beta+rob-rob1;
               end i;
             rob := c[j];
             if beta > gamma ∨ beta = gamma ∧ rob < alfa

               then begin
                        alfa := rob;
                        gamma := beta;
                        p := j
                      end beta > gamma ∨ beta = gamma ∧ c[j] < alfa

           end xx[j] = 0, j;
       if p = 0 then go to backtr;
       s := s+1;
       kk[s+1] := 0;
       t := t+1;
       jj[t] := p;
       ii[s] := xx[p] := 1;
       z := z+c[p];
```

```
      for i := 1 step 1 until m do y[i] := y[i]-a[i, p];
      go to beg
    end mnr = 0;
  s := s+1;
  ii[s] := kk[s+1] := 0;
  for j := 1 step 1 until n do
    if xx[j] < 0
    then
    begin
      t := t+1;
      jj[t] := j;
      ii[s] := ii[s]-1;
      z := z+c[j];
      xx[j] := 1;
      for i := 1 step 1 until m do y[i] := y[i]-a[i, j]
    end xx[j] < 0, j;
  go to beg;
sol: Fval := z;
  exist := true;
  for j := 1 step 1 until n do x[j] := if xx[j] = 1 then 1 else 0;
  go to finish;
backtr: for j := 1 step 1 until n do if xx[j] < 0 then xx[j] := 0;
btr: if s = 0 then go to finish;
  p := t;
  t := t-kk[s+1];
  for j := t+1 step 1 until p do xx[jj[j]] := 0;
  p := abs(ii[s]);
  kk[s] := kk[s]+p;
  for j := t-p+1 step 1 until t do
    begin
      p := jj[j];
      xx[p] := 2;
      z := z-c[p];
      for i := 1 step 1 until m do y[i] := y[i]+a[i, p]
    end j;
  s := s-1;
  go to if ii[s+1] < 0 then btr else beg;
finish: end Balas;
```

2. Method used. The algorithm is based on the paper [1] with some minor modifications for programming purposes. These were made to save computer memory space and may easily be located by inspection of the algorithm itself. No attempts have been made to include into the algorithm any of the later proposals given in the literature.

3. Certification. The algorithm Balas has been verified both on the Elliott 803 and on the Odra 1204 computers. All examples given in [1] have been solved and the same results obtained.

Reference

[1] E. Balas, *An additive algorithm for solving linear programs with zero-one variables,* Operations Research 13 (1965), pp. 517-546.

DEPT. OF NUMERICAL METHODS AND
DEPT. OF STATISTICS,
UNIVERSITY OF WROCŁAW

Received on 6. 12. 1968

ALGORYTM 3

Z. CYLKOWSKI i J. KUCHARCZYK (Wrocław)

ROZWIĄZYWANIE ZAGADNIEŃ ZERO-JEDYNKOWEGO CAŁKOWITEGO PROGRAMOWANIA LINIOWEGO METODĄ BALASA

STRESZCZENIE

Procedura *Balas* znajduje rozwiązanie zero-jedynkowego całkowitego programu liniowego metodą, opublikowaną przez Balasa w [1].

Dane:

m — liczba warunków liniowych,

n — liczba zmiennych zero-jedynkowych,

$a[1:m, 1:n]$ — macierz współczynników przy ograniczeniach,

$b[1:m]$ — prawe strony (wyrazy wolne) ograniczeń,

$c[1:n]$ — współczynniki funkcji celu.

Wyniki:

exist — **true**, jeżeli istnieje rozwiązanie optymalne, i **false**, jeżeli nie ma rozwiązania dopuszczalnego,

$x[1:n]$ — rozwiązanie optymalne (tylko, gdy *exist* = **true**, w przeciwnym wypadku procedura *Balas* nie zmienia tablicy *x*),

Fval — wartość funkcji celu (tylko, gdy *exist* = **true**, w przeciwnym wypadku $Fval = 2\times(1+c[1]+c[2]+\ldots+c[n])$).

Inne parametry:

max — największa dodatnia liczba typu **integer**.

Obliczenia kontrolne, wykonane na maszynach cyfrowych Elliott 803 i Odra 1204, wykazały poprawność procedury.

АЛГОРИТМ 3

З. ЦЫЛЬКОВСКИ и Й. КУХАРЧИК (Вроцлав)

РЕШЕНИЕ ЗАДАЧИ 0,1-ЦЕЛОЧИСЛЕННОГО ЛИНЕЙНОГО ПРОГРАММИРОВАНИЯ МЕТОДОМ БАЛАСА

РЕЗЮМЕ

Процедура *Balas* решает 0,1-целочисленную линейную задачу методом, опубликованным Баласом в [1].

Данные:

m — число линейных ограничений,

n — число переменных со значениями 0 или 1,

$a[1:m,\ 1:n]$ — матрица коэффициентов ограничений,

$b[1:m]$ — правые части (свободные члены) ограничений,

$c[1:n]$ — коэффициенты целевой функции.

Результаты:

exist — **true**, если оптимальное решение существует и **false**, если нет допустимого решения,

$x[1:n]$ — оптимальное решение (только при *exist* = **true**, в противном случае процедура *Balas* массива x не изменяет),

Fval — значение целевой функции (только при *exist* = **true**, в противном случае $Fval = 2\times(1+c[1]+c[2]+\ldots+c[n])$).

Другие параметры:

max — максимальное положительное число типа **integer**.

Контрольные вычисления выполнены на ЭВМ Эллиотт 803 и Одра 1204.