ZASTOSOWANIA MATEMATYKI
APPLICATIONES MATHEMATICAE
XII, 3 (1971)

ALGORITHM 14

M. SYSŁO (Wrocław)

INITIAL SOLUTION TO THE ZERO-ONE INTEGER LINEAR PROGRAMMING PROBLEM

1. Procedure declaration.

procedure *initsol* $(m, n, a, y, c, inf, opsol, x, z1)$;
value m, n, inf;
integer $m, n, opsol$;
real $z1, inf$;
integer array x;
array a, y, c;
comment *initsol* finds an initial solution to the zero-one linear programming problem:

$$\text{minimize } z = \sum_{j=1}^{n} c_j x_j,$$

$$\text{where } c_j \geqslant 0 \quad (j = 1, 2, \ldots, n),$$

$$\text{(1)} \quad \text{provided} \quad \sum_{j=1}^{n} a_{ij} x_j + y_i \geqslant 0 \quad (i = 1, 2, \ldots, m),$$

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

Data:

m — number of constraints,
n — number of variables,
$a[1:m, 1:n]$ — coefficient matrix of the constraints,
$y[1:m]$ — free terms of the constraints,
$c[1:n]$ — coefficients of the objective function,
inf — maximum positive number of type **real**.

Results:

opsol — integer number describing the type of the initial solution x found:

```
                    if opsol = 0, then x[j] = 0 (j = 1, 2, ..., n)
                    is a feasible, thus also an optimum, solution,
                    if opsol = 1, then an initial solution has
                    been found,
                    opsol = -1, in all other cases,
          x[1:n] -  initial solution (if opsol = 1),
          y[1:m] -  values of the left sides of the constraints
                    (1) for the initial solution,
              z1 -  the value of the objective function;
begin
 integer i, j, jmax, k, w, w1;
 real ej, max, p, q, r, sumyi;
 Boolean f, f1;
 procedure exchange(zw, fc, zs);
  integer zw;
  real fc, zs;
  begin
   for i := 1 step 1 until m do y[i] := y[i]+zs;
   z1 := z1+fc;
   x[zw] := 0;
   f1 := true
  end exchange;
 opsol := -1;
 f := true;
 z1 := sumyi := 0.0;
 for i := 1 step 1 until m do
  begin
   r := y[i];
   sumyi := sumyi+r;
   if r < 0.0
    then f := false
  end i;
 if f
  then begin
        opsol := 0;
        go to end
       end f;
 for j := 1 step 1 until n do x[j] := 0;
initpartsol: max := -inf;
 f := false;
 for j := 1 step 1 until n do
  if x[j] = 0
```

```
then begin
        r := 0.0;
        for i := 1 step 1 until m do
         begin
          q := y[i];
          p := a[i,j]+q;
          r := r+(if p < 0.0
                      then p
                      else 0.0)-(if q < 0.0
                                    then q
                                    else 0.0)
          end i;
         if r ≥ 0.0
          then begin
                q := c[j];
                ej := if q > 0.0
                        then r/q
                        else r-sumyi
               end r ≥ 0.0
          else ej := r;
         if ej > max
          then begin
                max := ej;
                jmax := j
               end ej > max
        end x[j] = 0,j;
if max ≥ 0.0
 then begin
        x[jmax] := 1;
        z1 := z1+c[jmax];
        for i := 1 step 1 until m do
         begin
          r := a[i,jmax];
          q := y[i] := y[i]+r;
          sumyi := sumyi+r;
          if q < 0.0
           then f := true
         end i;
        if f
         then go to initpartsol
         else begin
               opsol := 1;
```

```
            f := f1 := true;
mod2a : if f1
            then begin
                  f1 := false;
      mod2a1 : max := inf;
                  for k := 1 step 1 until n do
                   if x[k] = 0
                    then begin
                          r := c[k];
                          for j := 1 step 1 until n do
                           if x[j] = 1
                            then begin
                                  for i := 1 step 1 until m do
                                   if a[i, k] < a[i, j]
                                    then go to kk;
                                  if r < c[j] ∧ r < max
                                   then begin
                                         w := j;
                                         w1 := k;
                                         max := r
                                        end r < c[j] ...;
                             kk : end x[j] = 1, j;
                         end x[k] = 0, k;
                  if max < inf
                   then begin
                         exchange (w, c[w1] − c[w],
                                a[i, w1] − a[i, w]);
                         x[w1] := 1;
                         go to mod2a1
                        end max < inf;
                  if f ∨ f1
                   then begin
                         f1 := false;
                  mod2b : max := 0.0;
                         for j := 1 step 1 until n do
                          if x[j] = 1
                           then begin
                                 for i := 1 step 1 until m do
                                  if y[i] < a[i, j]
                                   then go to kkk;
                                 r := c[j];
                                 if r > max
```

```
                                                   then begin
                                                           w1 : = j;
                                                           max : = r
                                                           end r > max;
                                    kkk : end x[j] = 1, j;
                                      if max > 0.0
                                       then begin
                                              exchange (w1, −c[w1],
                                                           −a[i, w1]);
                                              go to mod2b
                                             end max > 0.0;
                                      f : = false;
                                      go to mod2a
                                     end f ∨ f1
                              end f1;
                         end ¬f
                 end max ≥ 0.0
            else if max = −inf
                 then opsol : = 1;
end: end initsol
```

2. Method used. The algorithm is based on paper [1].

3. Application. The results of this procedure may be used as an initial solution for any algorithm solving the 0-1 integer linear programming problem, e.g. [2]. In this case it is necessary to form the vectors s and v, and arrange the elements of vector s in terms of the second measure of feasibility contribution [1]. To do this, it is necessary to replace in [2] the procedure body from the beginning to the label *L0* by the following instructions (Remark: Enter procedure with $nosoln := \neg\, api$, $A[0, 0] = z1$, $A[i, 0] = y[i]$ $(i = 1, 2, \ldots, m)$ and exclude *api* from formal parameter list):

```
e : = 0;
if nosoln
 then begin
        for j : = 1 step 1 until n do
         s[j] : = v[j] : = 0;
        z : = 0.0
       end nosoln
 else begin
        z : = A[0, 0];
        null : = true;
        for i : = 1 step 1 until m do
```

```
  null := null ∧ A[i, 0] ≥ 0.0;
if null
 then begin
       k := 1;
       for j := 1 step 1 until n do
        if x[j] = 1
         then begin
               r := 0.0;
               for i := 1 step 1 until m do
                begin
                 q := A[i, j] - A[i, 0];
                 r := r + (if q < 0.0
                           then 0.0
                           else q)
                end i;
               for i := 1 step 1 until k-1 do
                if v[i] < r
                 then begin
                       for d := k step -1 until i+1 do
                        begin
                         s[d] := s[d-1];
                         v[d] := v[d-1]
                        end d;
                       s[i] := j;
                       v[i] := r;
                       go to con
                      end v[i] < r, i;
               s[k] := j;
               v[k] := r;
          con: k := k+1
              end x[j] = 1, j;
        e := k-1
      end null
 else begin
       k := 1;
       for j := 1 step 1 until n do
        if x[j] = 1
         then begin
               s[k] := j;
               k := k+1
              end x[j] = 1, j
      end ¬ null;
```

```
for j := 1 step 1 until n do
  v[j] := if x[j] = 1
          then 3
          else 0;
if null
  then go to L4
end ⌝ nosoln;
```

4. Certification. The procedure *initsol* has been verified on the examples from [3] and [4] on the Odra 1204 computer. The following table gives also the results obtained by using the solution of *initsol* in the modified procedure IMPLEN [2].

m	n	Optimum value of the objective function	*initsol*		*initsol*+IMPLEN		IMPLEN		Ref.
			time in sec.	*z1*	time in sec.	*count*	time in sec.	*count*	
15	15	10	26	12	1035	688			[3]
31	31	18	199	20	28800+				[3]
50	15	9	61	9	5452	2367			[3]
30	60	7643-7700	602	7675					[4]
30	60	8685-8698	500	8700					[4]
23	3	352	17	351	1348	1270	1569	1449	
3	4	14	1	14	1	2	2	7	

A (+) means that the problem was unsolved after the indicated time.

In the last 4 examples the function was maximized. In column 3 of examples 4 and 5 are given the intervals containing the approximate solutions obtained by applying the method of [4].

References

[1] J. L. Byrne and L. G. Proll, *Initialising Geoffrion's implicit enumeration algorithm for the zero-one linear programming problem*, Computer Journ. 12 (1969), p. 381-384.

[2] — Algorithm 341, Comm. ACM 11 (1968), p. 782.

[3] J. Haldi, *25 integer programming test problems*, Working paper No. 43, Graduate School of Business, Stanford University 1964.

[4] Schizuo Senju, Yoshiaki Toyoda, *An approach to linear programming with 0-1 variables*, Manag. Sci. B15 (1968), p. 196-207.

DEPARTMENT OF NUMERICAL METHODS
MATHEMATICAL INSTITUTE
UNIVERSITY OF WROCŁAW

Received on 29. 7. 1970

M. SYSŁO (Wrocław) ALGORYTM 14

ROZWIĄZANIE POCZĄTKOWE ZERO-JEDYNKOWEGO ZAGADNIENIA PROGRAMOWANIA LINIOWEGO

STRESZCZENIE

Procedura *initsol* znajduje rozwiązanie początkowe zero-jedynkowego zagadnienia programowania liniowego metodą opublikowaną w [1].

Dane:

m	—	liczba ograniczeń,
n	—	liczba zmiennych decyzyjnych,
$a[1:m, 1:n]$	—	tablica współczynników w ograniczeniach(1),
$y[1:m]$	—	tablica wyrazów wolnych w ograniczeniach (1),
$c[1:n]$	—	tablica wspóczynników funkcji celu,
inf	—	maksymalna liczba typu **real**.

Wyniki:

opsol	—	liczba całkowita, określająca rodzaj znalezionego rozwiązania x: jeśli $opsol = 0$, to $x[j] = 0$ $(j = 1, 2, \ldots, n)$ jest rozwiązaniem dopuszczalnym, a więc i optymalnym, jeśli $opsol = 1$, to znaleziono rozwiązanie początkowe, $opsol = -1$, w pozostałych przypadkach,
$x[1:n]$	—	tablica wartości rozwiązania początkowego,
$y[1:m]$	—	tablica wartości lewych stron ograniczeń (1) obliczonych dla rozwiązania początkowego,
$z1$	—	wartość funkcji celu.

Wyniki tej procedury mogą służyć jako rozwiązanie wstępne dla algorytmu znajdującego rozwiązanie optymalne, np. [2]. Podana została także konieczna w tym celu modyfikacja algorytmu [2].

Procedurę przetestowano na przykładach zaczerpniętych z [3] i [4].