ZASTOSOWANIA MATEMATYKI
APPLICATIONES MATHEMATICAE
XI, 2 (1970)

ALGORITHM 5

M. SYSŁO (Wroclaw)

SOLUTION OF THE TRANSPORTATION PROBLEM BY BRIGG'S DUAL LABELING METHOD

1. Procedure declaration.

procedure *Briggstransport* (*n*, *m*, *a*, *b*, *D*, *MAX*, *aa*, *bb*, *o*, *p*, *alfa*, *beta*, *z*);
value *n*, *m*, *MAX*;
integer *n*, *m*, *MAX*, *z*;
integer array *a*, *b*, *D*, *aa*, *bb*, *o*, *p*, *alfa*, *beta*;
comment *Briggstransport* finds the primal and dual solutions of the transportation problem:

$$\text{minimize } z = \sum_{i=1}^{n}\sum_{j=1}^{m} D_{ij}x_{ij},$$

$$\text{provided } \sum_{j=1}^{m} x_{ij} = a_i \quad (i = 1, 2, \ldots, n), \quad \sum_{i=1}^{n} x_{ij} = b_j \quad (j = 1, 2, \ldots, m),$$

$$\sum_{i=1}^{n} a_i = \sum_{j=1}^{m} b_j,$$

$$x_{ij} \geqslant 0 \qquad (i = 1, 2, \ldots, n,\ j = 1, 2, \ldots, m).$$

Data:

n	— number of origins,
m	— number of destinations,
$a[1:n]$	— amounts of goods available at the origins,
$b[1:m]$	— amounts of goods wanted at the destinations,
$D[1:n, 1:m]$	— matrix of unit transportation costs,
MAX	— maximum positive number of type **integer** available in the computer.

Results:

$aa, o[1:n], bb, p[1:m]$	— vectors in which the solution is located,

```
    as follows: If x_ij is nonzero in the solution, either aa[i] := j and
    o[i] := x_ij, or bb[j] := i and p[j] := x_ij, are assigned,
  alfa[1:n], beta[1:m] — the dual solution,
                     z — the value of the objective function;
begin
  integer i, j, h, io, jo, y, kk, b2, c2, av;
  Boolean f;
  integer array lam, eps, mi[1:n], delta, ka, ni[1:m];
  procedure X(k, l, w);
    integer k, l, w;
    comment Procedure X displaces the elements of the solution
    in the vectors aa, o[1:n], bb, p[1:m];
    begin
      integer a1, b1, c, d1, e, f1, s;
      if bb[l] = k
        then begin
               av := p[l] := p[l]+w;
               if av = 0 then bb[l] := 0
             end bb[l] = k
        else if aa[k] = l
             then begin
                    av := o[k] := o[k]+w;
                    if av = 0 then aa[k] := 0
                  end aa[k] = l
             else if bb[l] = 0
                    then begin
                           bb[l] := k;
                           p[l] := w
                         end bb[l] = 0
                  else if aa[k] = 0
                         then begin
                                aa[k] = l;
                                o[k] := w
                              end aa[k] = 0
                         else begin
                                d1 := k;
                                e := l;
                                f1 := w;
                           L1 : s := 0;
                                av := bb[e];
```

```
                    if aa[av] = 0
                      then begin
                             aa[av] := e;
                             o[av] := p[e];
                             s := 1
                           end aa[av] = 0
                      else begin
                             a1 := av;
                             b1 := e;
                             c := p[e]
                           end aa[av] ≠ 0;
                    bb[e] := d1;
                    p[e] := f1;
                    if s ≠ 1
                      then begin
                             d1 := a1;
                             e := aa[a1];
                             f1 := o[a1];
                             aa[a1] := b1;
                             o[a1] := c;
                             if bb[e] ≠ 0 then go to L1;
                             bb[e] := d1;
                             p[e] := f1
                           end s ≠ 1
                  end aa[k] ≠ 0
    end X;
  for i := 1 step 1 until n do alfa[i] := aa[i] := 0;
  for j := 1 step 1 until m do beta[j] := bb[j] := 0;
  z := 0;
i1: begin
    integer s, s1, ai, bj, MM;
    comment This block calculates the initial solution by the method
      of minimum cost matrix elements;
    MM := MAX;
    s := 1;
    for i := 1 step 1 until n do
      if a[i] ≠ 0
        then begin
               s := s1 := 0;
               for j := 1 step 1 until m do
                 if b[j] ≠ 0
```

```
                    then begin
                            s1 : = 1;
                            av : = D[i, j];
                            if av < MM
                              then begin
                                     MM : = av;
                                     io : = i;
                                     jo : = j
                                   end av < MM
                          end b[j] ≠ 0, j;
                  if s1 = 0 then go to fin
                end a[i] ≠ 0, i;
      if s = 1 then go to fin;
      ai : = a[io];
      bj : = b[jo];
      y : = if ai > bj then bj else ai;
      a[io] : = ai − y;
      b[jo] : = bj − y
    end of block which calculates the initial solution;
i2 : for i : = 1 step 1 until n do eps[i] : = mi[i] : = lam[i] : = 0;
    for j : = 1 step 1 until m do delta[j] : = ni[j] : = ka[j] : = 0;
    lam[io] : = − jo;
    eps[io] : = y;
    mi[io] : = D[io, jo] − alfa[io] − beta[jo];
ii : f : = false;
    for i : = 1 step 1 until n do
      begin
        av : = lam[i];
        if av < 0
          then begin
                 lam[i] : = − av;
                 for j : = 1 step 1 until m do
                   begin
                     b2 : = D[i, j] − alfa[i] − beta[j] − mi[i];
                     c2 : = b2 + ni[j];
                       if c2 < 0
                         then ni[j] : = − b2
                         else if c2 ≠ 0 ∨ b2 ⩾ 0 ∨ eps[i] ⩽ delta[j]
                                then go to e1;
                       delta[j] : = eps[i];
                       ka[j] : = − i;
                       f : = true;
```

```
            e1 : end j
          end av < 0
  end i;
if f then
    begin
    for j : = 1 step 1 until m do
          begin
            av : = ka[j];
            if av < 0
              then begin
                     ka[j] : = −av;
                     for i : = 1 step 1 until n do
                       begin
                         b2 : = mi[i]−ni[j];
                         kk : = if bb[j] = i
                                then p[j]
                                else if aa[i] = j
                                     then o[i]
                                     else 0;
                         if kk > 0 ∧ b2 ⩽ 0
                           then begin
                                  c2 : = delta[j];
                                  if kk ⩽ c2 then c2 : = kk;
                                  if b2 < 0
                                   then mi[i] : = ni[j]
                                   else if mi[i] ⩽ 0 ∨ c2 ⩽ eps[i]
                                        then go to ee1;
                                  eps[i] : = c2;
                                  lam[i] : = −j;
                                  f : = false
                                end kk > 0 ∧ b2 ⩽ 0;
                     ee1 : end i
                      end av < 0
            end j;
          if ¬ f then go to ii
        end f;
for i : = 1 step 1 until n do alfa[i] : = alfa[i]+mi[i];
for j : = 1 step 1 until m do beta[j] : = beta[j]−ni[j];
h : = delta[jo];
if h = 0
  then begin
```

```
            z : = z+y × D[io, jo];
            X(io, jo, y);
            go to i1
          end h = 0
        else begin
            y : = y−h;
            z : = z+h × (alfa[io]+beta[jo]);
            j : = jo;
            for i : = ka[j] while i ≠ io do
              begin
                kk : = lam[i];
                X(i, kk, −h);
                X(i, j, h);
                j : = kk
              end i;
            X(i, j, h);
            go to if y = 0 then i1 else i2
          end h ≠ 0;
fin : end Briggstransport
```

2. Method used. The procedure *Briggstransport* finds the primal and dual solutions of a transportation problem by the dual labeling method given by F. E. A. Briggs in [1].

The initial solution is found in the following way:
1° Find $x_{i_0j_0} = \min(a_{i_0}, b_{j_0})$, where i_0, j_0 are such that $D_{i_0j_0} = \min_{\substack{a_i>0 \\ b_j>0}}\{D_{ij}\}$,
2° adjust new a_{i_0}, b_{j_0} as follows: $a_{i_0} : = a_{i_0} - x_{i_0j_0}$, $b_{j_0} : = b_{j_0} - x_{i_0j_0}$,
3° do 1° and 2° as long as there are any positive a_i, b_j.

Instead of the usual flow matrix $\{x_{ij}\}$ a new method of remembering the solution is used; based on the knowledge that at most $m+n-1$ nonzero flows may occur, the solution is packed into the four vectors *aa*, $o[1 : n]$, *bb*, $p[1 : m]$, as described in the comment to the procedure. If necessary, the procedure X exchanges the elements, to avoid packing of two solution elements in one place.

3. Certification. The example given in [1] has been solved and correct results obtained, the computer being the Elliott 803. Table 1 gives relative computing times for various dimensions of solved problems.

TABLE 1. Relative computing times for *Briggstransport*

Dimension $n \times m$	5×5	10×10	10×20	20×20	30×30	50×50	20×200	10×400
Execution time	1	4	8	28	75	287	63	87

Reference

[1] F. E. A. Briggs, *A dual labeling method for the Hitchcock problem*, Operat. Res. 10 (1962), pp. 507-517.

DEPT. OF NUMERICAL METHODS
UNIVERSITY OF WROCŁAW

Received on 5. 12. 1968

M. SYSŁO (Wrocław)

ALGORYTM 5

ROZWIĄZANIE ZAGADNIENIA TRANSPORTOWEGO DUALNĄ METODĄ OZNACZANIA BRIGGSA

STRESZCZENIE

Procedura *Briggstransport* znajduje rozwiązanie pierwotne i dualne zagadnienia Hitchcocka dualną metodą oznaczania [1].

Kolejne elementy rozwiązania wstępnego wyznaczane są sukcesywnie metodą minimalnego elementu macierzy.

Ponieważ niezerowych elementów rozwiązania optymalnego może być najwyżej $m+n-1$, więc do zapamiętania ich użyto dwóch par wektorów *aa*, *o*[1 : *n*], *bb*, *p*[1 : *m*]. Procedura *X* zapewnia, że nowe elementy nie niszczą wcześniej obliczonych składowych wektorów rozwiązania.

Procedura *Briggstransport* była sprawdzona na maszynie cyfrowej Elliott 803.

АЛГОРИТМ 5

М. СЫСЛО (Вроцлав)

РЕШЕНИЕ ТРАНСПОРТНОЙ ЗАДАЧИ ДВОЙСТВЕННЫМ МЕТОДОМ РАССТАНОВКИ ПОМЕТОК БРИГГСА

РЕЗЮМЕ

Процедура *Briggstransport* находит прямое и двойственное решение задачи Хичкока двойственным методом расстановки пометок [1].

Очередные элементы исходного решения определяются последовательно методом минимального элемента матрицы.

Так как ненулевых элементов оптимального решения может быть не больше $m+n-1$ для запоминания их используются две пары векторов *aa*, *o*[1 : *n*], *bb*, *p*[1 : *m*]. Благодаря процедуре *X*, новые элементы не портят раньше вычисленных компонент векторов решения.

Процедура *Briggstransport* была проверена на вычислительной машине Эллиотт 803,

---