ZASTOSOWANIA MATEMATYKI
APPLICATIONES MATHEMATICAE
XV, 2 (1976)

ALGORITHM 45

A. ADRABIŃSKI (Wrocław)

A HEURISTIC ALGORITHM FOR THE TRAVELING-SALESMAN PROBLEM

1. Procedure declaration. Let us denote by $G_d = \langle X, U; d\rangle$ a network in which $G = \langle X, U\rangle$ is a complete graph (i.e., for all $x, y \in X$ we have $[x, y] \in U$) and d is a real function $d\colon U \to R$.

A closed path passing through each node exactly once is called a *Hamilton circuit of a graph G*. Procedure *TRAVEL* finds the shortest Hamilton cycle in a symmetric n-node network.

Data:

n — number of nodes of the network;

$P[1:n, 1:n]$ — symmetric array of distances between nodes of the network ($P[i,j] \geqslant 0$);

fh — integer number denoting the length of the route for a starting solution;

$RH[1:n]$ — array of node number of the starting solution.

Results:

fh — integer number denoting the length of the route of a heuristic solution;

$RH[1:n]$ — array of node numbers of a heuristic solution.

Remark. The starting solution ought to be chosen in such a manner to make it possible to finish the calculations as quick as possible with the solution nearly optimal. The best situation is if we can use many starting solutions. From a few local optima found in this way we choose the best one (for example, the shortest or best in the relation to the configuration of the route). Different methods using starting solutions are described in [14].

2. Method used. In procedure *TRAVEL* a shortened version of the algorithm of Lin and Kernighan ([11] and [12]) has been used. This algorithm is a peculiar case of a general heuristic method used for solving many optimization problems on a discrete set.

```
integer procedure TRAVEL(n,P,fh,RH);
 value n;
 integer n,fh;
 integer array P,RH;
 begin
  integer a,a1,a2,b,b1,b2,g,g0,g1,g2,g3,gp,i,j,j1,k,l,m,p,
  po,p1,s,t,z,x,y,y1,y2;
  Boolean I1,I2,I3;
  m:=n-1;
  begin
   integer array K[1:n,1:m],H,U[1:n],V,X[1:m],T[0:n+1];
   Boolean procedure TOUR(P,H);
    integer array P,H;
    begin
     integer a,b,i,j,k,l,m,p,s;
     Boolean B;
     B:=false;
     s:=n;
     H[1]:=i:=k:=l:=m:=1;
L1:  for j:=m step 1 until k-1,k+1 step 1 until s do
      begin
       a:=if B then j else i;
       b:=if B then i else j;
       p:=P[a,b];
       if p=1Vp=3
        then
        begin
         l:=l+1;
         k:=i;
         H[l]:=i:=j;
```

```
        if l≠n∧j≠1
         then
         begin
          if B
           then
           begin
            m:=j+1;
            s:=n
           end B
           else
           begin
            m:=1;
            s:=j-1
           end not B;
          B:=¬B;
          go to L1
         end l≠n ∧ j≠1;
        TOUR:=l=n;
        go to FINE
       end p=1 ∨ p=3
      end j;
     if B
      then
      begin
       m:=i+1;
       k:=i;
       s:=n
      end B
      else
      begin
```

```
          m:=s:=1;
          k:=i
         end not B;
        B:=¬B;
        go to L1;
FINE:
       end TOUR;
      for k:=1 step 1 until n do
       begin
        for i:=1 step 1 until n do
         U[i]:=P[k,i];
        for i:=1 step 1 until m do
         begin
          p:=-1;
          l:=n-i;
          for j:=1 step 1 until n do
           if U[j]>p
            then
            begin
             p:=U[j];
             V[l]:=j
            end U[j]>p,j;
          U[V[l]]:=-1
         end i;
        for i:=1 step 1 until m do
         K[k,i]:=V[i]
       end k;
      z:=0;
ROT:
      T[0]:=RH[n];
      T[n+1]:=RH[1];
      for i:=1 step 1 until n do
       T[i]:=RH[i];
      I2:=false;
```

```
PER:
   z:=z+1;
   for i:=1 step 1 until n do
    if T[i]=z
     then go to E1;
E1:j:=i-1;
   l:=n-j;
   for k:=1 step 1 until n do
    H[k]:=T[k];
   for k:=1 step 1 until l do
    T[k]:=RH[k]:=H[i+k-1];
   for k:=1 step 1 until j do
    T[l+k]:=RH[l+k]:=H[k];
   T[n+1]:=T[1];
   T[0]:=T[n];
   for i:=1 step 1 until n do
    for j:=i+1 step 1 until n do
     P[i,j]:=0;
   for i:=1 step 1 until n do
    begin
     k:=T[i];
     l:=T[i+1];
     if k<l
      then P[k,l]:=1
      else P[l,k]:=1
    end i;
   I1:=I3:=false;
   t:=z;
E: m:=T[2];
   a:=if t<m then t else m;
   b:=if t>m then t else m;
   x:=P[b,a];
   P[a,b]:=2;
   X[1]:=m;
   i:=k:=0;
```

```
    p1:=1;
B:  l:=K[m,p1];
    a:=if m<l then m else l;
    b:=if m>l then m else l;
    if P[a,b]=0
     then
     begin
      y:=P[b,a];
      if x-y>0
       then go to A
       else go to P6d
     end P[a,b]=0
     else
     begin
      p1:=p1+1;
      if p1>5
       then go to P6d;
      go to B
     end P[a,b]≠0;
A:  P[1,1]:=l;
    g:=g1:=gp:=x-y;
    P[a,b]:=3;
    for j:=4 step 1 until n do
     if T[j]=l
      then go to C;
C:  m:=T[j-1];
    j1:=j;
    a:=if m<l then m else l;
    b:=if m>l then m else l;
    x:=P[b,a];
```

```
    P[a,b]:=2;
    y1:=P[if m>t then m else t,if m<t then m else t]:
    X[2]:=m;
    g3:=x-y1;
    if g1+g3>0
     then
     begin
      g0:=g1+g3;
      k:=2
     end g1+g3>0
     else g0:=0;
    p:=1;
A1:l:=K[m,p];
    a:=if m<l then m else l;
    b:=if m>l then m else l;
    if P[a,b]=0∧l≠t
     then y:=P[b,a]
     else
     begin
      p:=p+1;
      if p>5
       then
       begin
        if g0>0
         then
         begin
          a:=if m<t then m else t;
          b:=if m>t then m else t;
          go to CB
         end g0>0
```

```
         else go to if I1 then P1 else P2
        end p>5;
       go to A1
      end P[a,b]=0V1=t;
     P[2,2]:=1;
     g2:=x-y;
     gp:=g1+g2;
     if gp≤0
      then
      begin
       if g0>0
        then
        begin
         a:=if m<t then m else t;
         b:=if m>t then m else t;
         go to CB
        end g0>0
        else go to if I1 then P1 else P2
      end gp≤0;
     if gp≤g0
      then
      begin
       a:=if m<t then m else t;
       b:=if m>t then m else t;
       go to CB
      end gp≤g0;
     P[a,b]:=3;
     s:=2;
     for j:=1 step 1 until n do
      if T[j]=1
```

```
      then go to B1;
B1:s:=s+1;
   m:=T[j-1];
   a1:=if m<l then m else l;
   b1:=if m<l then l else m;
   if P[a1,b1]=2∨j=3
    then go to A2;
   a:=if m<t then m else t;
   b:=if m>t then m else t;
   P[a,b]:=3;
   P[a1,b1]:=2;
   if TOUR(P,H)
    then
    begin
     y2:=P[b,a];
     x:=P[b1,a1];
     X[s]:=m;
     P[a,b]:=0;
     go to B2
    end TOUR(P,H);
   P[a,b]:=0;
   P[a1,b1]:=1;
A2:m:=T[j+1];
   a1:=if m<l then m else l;
   b1:=if m>l then m else l;
   if P[a1,b1]=2∨j=n∨j=n-1
    then go to CA;
   a:=if m<t then m else t;
   b:=if m>t then m else t;
   P[a,b]:=3;
```

```
P[a1,b1]:=2;
if TOUR(P,H)
 then
 begin
  y2:=P[b,a];
  x:=P[b1,a1];
  X[s]:=m;
  P[a,b]:=0;
  go to B2
 end TOUR(P,H)
 else
 begin
    P[a,b]:=0;
    P[a1,b1]:=1;
    go to CA
   end TOUR(P,H);
B2:g3:=x-y2;
   if gp+g3>g0
    then
    begin
     g0:=gp+g3;
     k:=s;
     i:=0
    end gp+g3>g0
    else i:=1;
   po:=1;
B3:l:=K[m,po];
   a2:=if m<l then m else l;
   b2:=if m>l then m else l;
   if P[a2,b2]=0∧l≠t
```

```
        then y:=P[b2,a2]
        else
        begin
         po:=po+1;
         if po>5
          then go to CB;
         go to B3
        end P[a2,b2]≠0∨1=t;
       P[s,s]:=1;
       g:=x-y;
       gp:=gp+g;
       if gp≤0∨gp≤g0
        then go to CB;
       P[a2,b2]:=3;
       for j:=1 step 1 until n do
        if T[j]=1
         then go to B1;
    CA:I3:=true;
    CB:if z≥n
        then go to LOOP;
       if g0>0
        then
        begin
         fh:=fh-g0;
         if i=0
          then
          begin
           if I3
            then
            begin
```

```
      x:=X[k];
      P[a2,b2]:=0;
      P[if t<x then t else x,if t>x then t else x]:=3;
      TOUR(P,T);
      go to PER
     end I3;
    P[a,b]:=3;
    TOUR(P,T);
    go to PER
   end i=0
   else
    if I3
     then
     begin
      s:=s-1;
      for p:=k step 1 until s do
       begin
        a:=X[p];
        b:=P[p,p];
        P[if a<b then a else b,if a>b then a else b]:=0
       end p;
      s:=s-1;
      for p:=k step 1 until s do
       begin
        m:=p+1;
        a:=X[m];
        b:=P[p,p];
        P[if a<b then a else b,if a>b then a else b]:=1;
       end p;
      a:=X[k];
```

```
      P[if a<t then a else t,if a>t then a else t]:=3;
      TOUR(P,T);
      go to PER
     end I3
     else
     begin
      s:=s-1;
      for p:=k step 1 until s do
       begin
        a:=X[p];
        b:=P[p,p];
        m:=p+1;
        P[if a<b then a else b,if a>b then a else b]:=0;
        a:=X[m];
        P[if a<b then a else b,if a>b then a else b]:=1
       end k;
      a:=X[k];
      P[if a<t then a else t,if a>t then a else t]:=3;
      TOUR(P,T);
      go to PER
     end not I3,i≠0
  end g0>0;
 if i=0
  then
  begin
   p:=p+1;
   m:=X[2];
   P[if m<1 then m else 1,if m>1 then m else 1]:=0;
   I3:=false;
   if p>5
```

```
  then go to if I1 then P1 else P2;
 go to A1
end i=0
else
 if I3
  then
  begin
   s:=s-1;
   for po:=2 step 1 until s do
    begin
     x:=X[po];
     y:=P[po,po];
     P[if x<y then x else y,if x>y then x else y]:=0
    end po;
   s:=s-1;
   for po:=2 step 1 until s do
    begin
     m:=po+1;
     x:=X[m];
     y:=P[po,po];
     P[if x<y then x else y,if x>y then x else y]:=1
    end po;
   m:=X[2];
   y:=P[1,1];
   x:=P[if m>y then m else y,if m<y then m else y];
   p:=p+1;
   I3:=false;
   if p>5
    then go to if I1 then P1 else P2;
   go to A1
```

```
      end I3
      else
      begin
       s:=s-1;
       for po:=2 step 1 until s do
        begin
         x:=X[po];
         y:=P[po,po];
         P[if x<y then x else y,if x>y then x else y]:=0;
         m:=po+1;
         x:=X[m];
         P[if x<y then x else y,if x>y then x else y]:=1
        end po;
       m:=X[2];
       y:=P[1,1];
       x:=P[if m>y then m else y,if m<y then m else y];
       p:=p+1;
       if p>5
        then go to if I1 then P1 else P2;
       go to A1
      end not I3,i≠0;
P2:I1:=true;
   if j1=n
    then go to P1;
   a:=l:=P[1,1];
   b:=X[2];
   P[if a<b then a else b,if a>b then a else b]:=1;
   X[2]:=m:=T[j1+1];
   a:=if l<m then l else m;
   b:=if l>m then l else m:
```

```
    x:=P[b,a];
    P[a,b]:=2;
    gp:=g1;
    po:=1;
A1P2:
    l:=K[m,po];
    a:=if m<l then m else l;
    b:=if m>l then m else l;
    if P[a,b]=0∧l≠t
     then y:=P[b,a]
     else
     begin
      po:=po+1;
      if po>5
       then go to P1;
      go to A1P2
     end P[a,b]≠0∨l=t;
    P[2,2]:=1;
    g:=x-y;
    gp:=gp+g;
    if gp≤0
     then go to P1;
    P[a,b]:=3;
    s:=2;
    for j:=1 step 1 until n do
     if T[j]=1
      then go to B1P2;
B1P2:
    if j>j1+1
     then go to B2P2
```

```
   else go to B1;
B2P2:
  s:=s+1;
  m:=T[j-1];
  a1:=if l<m then l else m;
  b1:=if l>m then l else m;
  P[a1,b1]:=2;
  x:=P[b1,a1];
  X[3]:=m;
  po:=i:=1;
A2P2:
  l:=K[m,po];
  for j:=1 step 1 until n do
   if T[j]=l
    then go to A3P2;
A3P2:
  if j≥j1+1
   then
   begin
    po:=po+1;
    if po>5
     then go to P11;
    go to A2P2
   end j>j1+1;
  if l≠t
   then
   begin
    a2:=if m<l then m else l;
    b2:=if m>l then m else l;
    y:=P[b2,a2]
```

```
      end l≠t
      else
      begin
       po:=po+1;
       if po>5
        then go to P11;
       go to A2P2
      end l=t;
     P[3,3]:=1;
     g:=x-y;
     gp:=gp+g;
     if gp≤0
      then go to P11;
     P[a2,b2]:=3;
     for j:=1 step 1 until n do
      if T[j]=1
       then go to B1;
P11:
     P[a1,b1]:=1;
     P[a,b]:=0;
P1:a:=X[2];
     b:=P[1,1];
     P[if a<b then a else b,if a>b then a else b]:=1;
     m:=a:=X[1];
     P[if a<b then a else b,if a>b then a else b]:=0;
     p1:=p1+1;
     if p1>5
      then go to P6d;
     x:=P[if m<t then t else m,if m>t then t else m];
     go to B;
```

```
P6d:
   if I2
    then
    begin
     if z=n
      then go to LOOP;
     go to ROT
    end I2;
   I2:=true;
   a:=t;
   b:=T[2];
   P[if a<b then a else b,if a>b then a else b]:=1;
   m:=n-1;
   for i:=1 step 1 until m do
    T[i+1]:=RH[n-i+1];
   T[0]:=T[n];
   go to E;
LOOP:
   for i:=1 step 1 until n do
    RH[i]:=T[i]
  end
 end TRAVEL
```

The class of these problems can be formulated shortly as follows:

find a subset $T \subset S$ which satisfies some criteria Z and minimizes the function F.

For example, a traveling-salesman problem can be formulated as follows:

from a set U of all edges of a complete directed graph G find a Hamilton circuit H of minimal length.

Generally, the method mentioned above can be described by the following steps:

1. Generate a pseudo-random solution T satisfying some criterion Z.
2. Improve the solution T by some transformation.
3. If the found solution T' is better than T, i.e. if $F(T') < F(T)$, replace T by T' and repeat from step 2.
4. If no improved solution can be found, T is a locally optimum solution. Repeat from step 1 until the computation time runs out or the answers are satisfactory.

The most important part of this procedure is the choice of the initial solution and transformation. The method of generation of the initial solution has been given in the Remark. In general, however, the transformation in step 2 relies on finding a number k and sets

$$\{x_1, x_2, \ldots, x_k\} \subset T \quad \text{and} \quad \{y_1, y_2, \ldots, y_k\} \subset S-T$$

to be the best in the given iteration. These sets are generated "element by element", i.e. we choose $x_1 \in T$ and $y_1 \in S-T$ as the "most-out-of-place". Then we choose elements $x_2 \in T-\{x_1\}$ and $y_2 \in S-T-\{y_1\}$ again in such a way that the replacement of $\{x_1, x_2\}$ by $\{y_1, y_2\}$ gives the greatest possible decrease of the Hamilton cycle length. We continue the process taking

TABLE 1

Source of example	Dimensions of problem	Finding the starting solutions		Finding the optimum solutions		Frequency of optimum solution occurrence	Optimum value of objective function
		value of objective function	time	value of objective function	time		
Karg [8]	5	148	1	148	1	1	148
Flood A [6]	5	32	1	32	1	1	32
Flood B [6]	6	22	4	22	2	1	22
Little [13]	6	70	1	70	2	0.8	70
Dantzig [5]	10	425	2	387	7	0.25	378
Christofides [2]	10	229	2	212	6	1	212
Lawler and Wood [9]	10	159	2	152	3	0.43	150
Croes [3]	20	398	13	253	49	0.11	246
Held and Karp [7]	25	1772	24	1711	162	0.42	1711
Polish cities (1)	27	3901	30	3336	126	0.36	3336
Polish cities (2)	27	4074	29	3757	121	0.5	3757
Karg [8]	33	12218	53	10861	228		10861
Dantzig [4]	42	1591	123	1399	572		1398
Held and Karp [7]	48	13055	150	12294	1186		11470
Karg [8]	57	16538	236	13459	1741		12955

(1) Railway distances.
(2) Road distances.
(Data taken from the Dom Książki Calendar for 1974.)

care that a possible replacement of $\{x_1, x_2, \ldots, x_k\}$ by $\{y_1, y_2, \ldots, y_k\}$ gives a solution that satisfies the criterion Z.

As is easily seen, the number k may be different in each iteration.

3. Certification. Procedure *TRAVEL* has been verified on the ODRA 1204 computer for many examples found in the literature. The method of "the nearest neighbour with a starting point at the greatest gradient" [14] was used to obtain the starting solutions. The computational results are listed in Table 1 (time in secs.).

It is seen from Table 1 that in most cases the optimum solution has been obtained by using only one starting solution. Optimum solutions for almost all examples have been obtained when more random starting solutions were used.

References

[1] M. Bellmore and G. L. Nemhauser, *The traveling salesman problem (A survey)*, Opns. Res. 16 (1968), p. 538-558.

[2] N. Christofides, *Bounds for the traveling-salesman problem*, ibidem 20 (1972), p. 1044-1056.

[3] G. A. Croes, *A method for solving traveling-salesman problem*, ibidem 6 (1958), p. 791-812.

[4] G. B. Dantzig, D. R. Fulkerson and S. M. Johnson, *Solution of a large-scale traveling-salesman problem*, ibidem 2 (1954), p. 393-410.

[5] — *On a linear-programming, Combinatorial approach to the traveling-salesman problem*, ibidem 7 (1959), p. 58-66.

[6] M. M. Flood, *The traveling-salesman problem*, ibidem 4 (1956), p. 61-75.

[7] M. Held and R. M. Karp, *A dynamic programming approach to sequencing problems*, J. Soc. Ind. Math. 10 (1962), p. 196-210.

[8] R. L. Karg and G. L. Thompson, *A heuristic approach to solving traveling salesman problems*, Manag. Sci. 10 (1964), p. 225-248.

[9] E. L. Lawler and D. E. Wood, *Branch-and-bound methods (A survey)*, Opns. Res. 14 (1966), p. 699-719.

[10] S. Lin, *Computer solutions of the traveling-salesman problem*, B.S.J. 44 (1965), p. 2245-2269.

[11] — and B. W. Kernighan, *A heuristic algorithm for the traveling salesman problem*, Comp. Sci. Rep. 1 (1972), p. 1-40.

[12] — *An effective heuristic algorithm for the traveling-salesman problem*, Opns. Res. 21 (1973), p. 498-516.

[13] J. D. C. Little, K. G. Murty, D. W. Sweeney and C. Karel, *An algorithm for the traveling-salesman problem*, ibidem 11 (1963), p. 972-989.

[14] Z. Skupień and M. M. Sysło, *Stosowana teoria grafów III. Grafy Eulera i Hamiltona. Zagadnienie komiwojażera*, Matem. Stosow. (w druku).

INSTITUTE OF INFORMATICS
UNIVERSITY OF WROCŁAW
50-384 WROCŁAW

Received on 17. 1. 1975

ALGORYTM 45

A. ADRABIŃSKI (Wrocław)

HEURYSTYCZNY ALGORYTM DLA ZAGADNIENIA KOMIWOJAŻERA

STRESZCZENIE

Procedura *TRAVEL*, oparta na metodzie Lina i Kernighana ([11] i [12]), poszukuje możliwie najkrótszego cyklu Hamiltona w symetrycznej n-węzłowej sieci.

Dane:

n — liczba wierzchołków sieci;
$P[1:n, 1:n]$ — symetryczna tablica odległości między wierzchołkami sieci ($P[i, j] \geqslant 0$);
fh — liczba całkowita, oznaczająca długość drogi rozwiązania początkowego;
$RH[1:n]$ — tablica numerów wierzchołków rozwiązania początkowego.

Wyniki:

fh — liczba całkowita oznaczająca długość drogi rozwiązania heurystycznego;
$RH[1:n]$ — tablica numerów wierzchołków rozwiązania heurystycznego.

Uwaga. Różne metody otrzymywania rozwiązania początkowego zawarte są w [14].

Obliczenia kontrolne, wykonane na maszynie cyfrowej ODRA 1204, wykazały poprawność procedury *TRAVEL*. W większości przykładów otrzymano rozwiązanie optymalne, startując z jednego tylko rozwiązania początkowego.