A program for the traveling-salesman problem, according to the heuristic algorithm of Lin-Kernighan
Keulemans, W.K.M.

Published: 01/01/1977

Document Version
Publisher’s PDF, also known as Version of Record (includes final page, issue and volume numbers)

Please check the document version of this publication:

• A submitted manuscript is the author’s version of the article upon submission and before peer-review. There can be important differences between the submitted version and the official published version of record. People interested in the research are advised to contact the author for the final version of the publication, or visit the DOI to the publisher’s website.
• The final author version and the galley proof are versions of the publication after peer review.
• The final published version features the final layout of the paper including the volume, issue and page numbers.

Link to publication

Citation for published version (APA):

General rights
Copyright and moral rights for the publications made accessible in the public portal are retained by the authors and/or other copyright owners and it is a condition of accessing publications that users recognise and abide by the legal requirements associated with these rights.

• Users may download and print one copy of any publication from the public portal for the purpose of private study or research.
• You may not further distribute the material or use it for any profit-making activity or commercial gain
• You may freely distribute the URL identifying the publication in the public portal ?

Take down policy
If you believe that this document breaches copyright please contact us providing details, and we will remove access to the work immediately and investigate your claim.

Download date: 20. Dec. 2018
A program for the Traveling-Salesman Problem,
according to the heuristic algorithm of
Lin-Kernighan

by

W.K.M. Keulemans

Eindhoven, May 1977

The Netherlands
A program for the Traveling-Salesman Problem, according to the heuristic algorithm of Lin-Kernighan

by

W.K.M. Keulemans

1. Introduction

The program described in this memorandum has been written in Burroughs Extended Algol (BEA). It is a fair transcription of the heuristic algorithm of Lin-Kernighan [2]. The terminology and notations of Lin-Kernighan will be used. The program follows the basic traveling-salesman algorithm of Lin-Kernighan conscientiously. The refinements of avoiding checkout time, lookahead and reduction have also been implemented. The program has been written as a procedure called linkernighan. Computational results are compared with the results of Lin-Kernighan.

2. The procedure linkernighan (n,x,dis,numtours,out)

The meaning of the formal parameters is as follows:

n : the number of cities minus one, minus one because the counting of the cities starts with zero
x : an integer variable which is used as an argument for the intrinsic random, which in its turn is used to generate a random starting-tour
dis : the distance matrix with dimensions 0:n,0:n
numtours: the number of tours that has to be generated
out : an output file

The important identifiers declared in linkernighan are:

m : the number of cities (m = n + 1)
numsol : the number of different locally optimal tours that has been found
posit : the positions of the cities in the locally optimal tour are stored in posit
sol : the locally optimal tour is stored in sol
pre,post: these arrays will contain the information about the links common to all locally optimal tours that have been found in linkernighan; when the first locally optimal tour has been found the predecessors of the cities are stored in pre, the successors in post.
nb: an array with the same dimensions as dis, which in every row will contain the numbers of the cities in order of increasing distance to city i; the distance of city i to city i is artificially made very large
solutions: the different locally optimal tours are stored in solutions
optimum: the values corresponding to the tours in solutions are stored in optimum
sortdistances: a procedure to sort the cities in order of increasing distances
kernighanlin: a procedure which forms the essential part of linkernighan, it delivers one locally optimal tour.

3. The procedure sortdistances (n,nb,dis)

The meaning of the formal parameters n,nb,dis in the same as the meaning of the corresponding actual parameters n,nb and dis. The time needed for the sorting process is proportional to n log n.

4. The procedure Kernighanlin (x,opt,posit,sol)

Again the meaning of the formal parameters x,posit,sol is the same as the meaning of the actual parameters x,posit,sol,opt will contain the value of the locally optimal tour.

The important identifiers declared in Kernighanlin are:

nod1,nod2: nod1 and nod2 are used to store a sequence of the cities, this sequence is called an actual tour
pos1,pos2: pos1 and pos2 are used to store the positions of the cities corresponding to nod1 c.q. nod2
posit1,sol1: each time a better tour has been found this one is stored in sol1 and the positions of the cities in posit1. At the end of this iteration sol1 and posit1 are copied in sol and posit. This may not be done before, because the arrays sol and posit are used to test whether a link is a x-link.
startingtour: a procedure to generate the random starting tour
findlink: a procedure to determine the first y-link to be considered
findlinks: a procedure which chooses out of a maximum of 5 possible candidates the link which maximizes $|x_{i+1}| - |y_i|$, this procedure takes care of the lookahead refinement
reverse: a procedure to reverse a specified part of the actual tour, by this reversion a new actual tour is formed
A procedure which tests whether an actual tour is a better tour than the best tour found so far in KernighanLin.

Tour Improvement: this procedure takes care of the iteration process on levels > 3 for x-links and on levels > 2 for y-links.

5. Differences between BEA and Algol 60

BEA differs in some ways from Algol 60. The differences, that occur in the program, which need some explanation are:

1) In BEA specifications of formal parameters are obligatory. Specifications of arrays must be accompanied by the lowerbounds for each dimension. For reasons of efficiency these lowerbounds are preferred to be zero.

2) In BEA the conception of array-row makes it possible to treat a part of a more dimensional array, in which only the last index varies, as a one-dimensional array. For instance a row of a two-dimensional array can thus be an actual parameter.

3) In BEA there are not value arrays in the sense of Algol 60. This leads to the complication that the arrays sometimes have to be copied explicitly. There is however a fast way to copy arrays, this is done by means of the write statement. When a and b are one-dimensional arrays (arrayrows) of the same length the statement write (a,*,b) copies b in a. The write statement is also used to write output to an outputfile, in which case the identifier a must be a file identifier.

4) The construction: if boolean expression then if boolean expression then is a valid construction in BEA. It can easily be adapted to Algol 60 by the corresponding construction: if boolean expression and boolean expression then.

5) The intrinsics random and mod are implemented in BEA.

6. The implemented refinements

The following refinements have been implemented:

a) avoiding checkout time

Each time an improved tour has been found in the procedure KernighanLin the program checks whether this tour is one of the locally optimal tours found before. If this is the case the search for a better tour may be stopped, because this search already has been done in one of the earlier calls of KernighanLin.
b) lookahead
A restricted lookahead is added to the procedure: in all steps where a \( y_i \) is chosen the choice is not made on basis of minimum \( |y_i| \), but on basis of maximizing \( g_i = |x_{i+1}| - |y_i| \) over a maximum number of 5 possible candidates.

c) reduction
Once a number of locally optimal tours have been found by successive calls of kernighanlin, we observe that certain links are common to all of them. For reasons of efficiency the search for better tours is limited by the following reduction: links common to all locally optimal tours may be broken on a level 1, 2 or 3, but not on higher levels. This means that links of the actual tour common to all locally optimal tours will not be broken in the procedure tourimprovement.

7. Computational results

The following problems have been run with our program on a B7700:

<table>
<thead>
<tr>
<th>Size</th>
<th>Source</th>
<th>Frequency of optimum</th>
<th>cpu-time/tour</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Pre-reduction</td>
<td>Post-reduction</td>
</tr>
<tr>
<td>1</td>
<td>33</td>
<td>0.80</td>
<td>0.90</td>
</tr>
<tr>
<td>2</td>
<td>42</td>
<td>1.00</td>
<td>1.00</td>
</tr>
<tr>
<td>3</td>
<td>57</td>
<td>0.23</td>
<td>0.23</td>
</tr>
<tr>
<td>4</td>
<td>100</td>
<td>0.50</td>
<td>0.50</td>
</tr>
</tbody>
</table>

* average number of distinct solutions

The corresponding results of Lin-Kernighan on a GE635 are:

<table>
<thead>
<tr>
<th>Size</th>
<th>Source</th>
<th>Frequency of optimum</th>
<th>cpu-time/tour</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Pre-reduction</td>
<td>Post-reduction</td>
</tr>
<tr>
<td>1</td>
<td>33</td>
<td>1.00</td>
<td>-</td>
</tr>
<tr>
<td>2</td>
<td>42</td>
<td>1.00</td>
<td>-</td>
</tr>
<tr>
<td>3</td>
<td>57</td>
<td>0.27</td>
<td>0.43</td>
</tr>
<tr>
<td>4</td>
<td>100</td>
<td>0.56</td>
<td>0.63</td>
</tr>
</tbody>
</table>

* average number of distinct solutions
The results achieved by our program are considerably worse than those achieved by Lin-Kernighan. By their program the optimum is found more frequently; the difference in computing time between pre-reduction and post-reduction tours in our program can be explained by the gain achieved by avoiding checkout time, but a few of the locally optimal tours generated after reduction are new locally optimal tours. Lin-Kernighan have apparently implemented another form of reduction than we have done. In our program we start for the post-reduction tours also with a random starting tour and links common to all locally optimal tours may not be broken at a level 4 or deeper. Postreduction starts when either 5 distinct locally optimal tours have been found or so locally optimal tours have been generated. The program of Lin-Kernighan probably does not start with a random starting tour but with a tour in which all the links common to the locally optimal tours are present and in which only the links not common to all locally optimal tours are chosen at random. This alternative has also been implemented. To the program the procedure reducttour is added, which for generates the starting tours for the post-reduction tours. The computing results for this alternative are:

<table>
<thead>
<tr>
<th>Size</th>
<th>Source</th>
<th>Frequency of optimum</th>
<th>cpu-time/tour</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Pre-reduction</td>
<td>Post-reduction</td>
</tr>
<tr>
<td>1</td>
<td>33</td>
<td>[3]</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>42</td>
<td>[1]</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>57</td>
<td>[3]</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>100</td>
<td>[4]</td>
<td></td>
</tr>
</tbody>
</table>

The results fit moderately to those achieved by Lin-Kernighan, the frequency of optimum for pre-reduction tours may be biased by the fact that post-reduction tours have been generated once three different locally optimal tours had been found.

7. References


PROCEDURE LINKERNIGHAN(N,X,DIS,NUMTOURS,OUT);
VALUE N,NUMTOURS; INTEGER N,NUMTOURS,X; FILE OUT;
BEGIN INTEGER I,J,K,L,M,R,NMSOL,TOURS; REAL OPT;
BOOLEAN OLDSOL,REDUCTION;
INTEGER ARRAY POSIT,SOL,PRE,POST,SEQ[O:N], NC[O:N,O:N],
SOLUTIONS[1:NUMTOURS,O:N];
REAL ARRAY OPTIMUM[1:NUMTOURS];

PROCEDURE SORTDISTANCES(N,A,P);
VALUE N; INTEGER N;
FILE P;
REAL ARRAY A[O;O];
BEGIN INTEGER I,K,L,M,R,NUMSOL,TOURS;
REAL OPT;
BOOLEAN OLDSOL,REDUCTION;
INTEGER ARRAY POSIT,SOL,PRE,POST,SEQ[O:N], NC[O:N,O:N],
SOLUTIONS[1:NUMTOURS,O:N];
REAL ARRAY OPTIMUM[1:NUMTOURS];

PROCEDURE SORT(L,R);
VALUE L,R; INTEGER L,R;
BEGIN IF R-L &gt; 1 THEN BEGIN INTEGER I,K,IL,IR;
K:=(L+R)/2; SORT(L,K); IR:=K+1; SORT(IR,R); I:=IL:=L;
WHILE IL LEQ K AND IR LEQ R DO
AC[IL]:=P[I]; IL:=IL+1; I:=I+1 END
ELSE BEGIN
AC[IR]:=P[I]; IR:=IR+1; I:=I+1 END;
IF IR &gt; R THEN WHILE I LEQ R DO
BEGIN
AC[I]:=P[I]; IL:=IL+1; I:=I+1 END
ELSE WHILE I LEQ R DO
BEGIN
AC[I]:=P[I]; IR:=IR+1; I:=I+1 END;
I:=L-1;
WHILE I:=I+1 LEQ R DO P[I]:=C[I]
END
ELSE IF R = L THEN P[R]:=R
ELSE IF ALL &gt; A[R] THEN BEGIN P[L]:=R
END
ELSE BEGIN P[L]:=L; P[R]:=R END;
SORT(O,N)
END SORTDISTANCES;

PROCEDURE SORTSEQ(N,SEQ);
BEGIN INTEGER N; INTEGER ARRAY SEQ[O];
BEGIN INTEGER I,K,L; REAL F; INTEGER ARRAY PLACE[O:N];
REAL K:=SOL[N]; L:=SOL[O];
FOR I:=1 STEP 1 UNTIL N DO
BEGIN F:=DISK[L]; K:=L; L:=SOL[I]; F:=MAX(F,DISK[L]);
PLACE[K]:=F;
END;
PLACE[K]:=-F
END SORTSEQ;

PROCEDURE KERNIGHANLIN(X, OPT, POSIT, SOL);

KERNIGHANLIN DETERMINES ONE LOCAL-OPTIMAL SOLUTION X

INTEGER X; REAL OPT; INTEGER ARRAY POSIT, SOL[

BEGIN INTEGER T1, T2, T3, T4, T5, T6, T7, T8, NT3, NT5, I1, I2, IT2, IT4, IT6,

NULL, NUM, PT, I, J, K, L, M, R, S, V, W,

PT1, PT2, PT3, PT4, PT5, PT6, PT7, PT8;

REAL GG, HH, FF, MAXGG;

BOOLEAN FOUND, BOOL;

INTEGER ARRAY POS1, POS2, NOD1, NOD2, POSIT1, SOL1[C: N], N3, N5[C: 5];

56000 PROCEDURE STARTINGTOUR(X, POSIT, SOL, OPT);

INTEGER X; REAL OPT; INTEGER ARRAY POSIT, SOL[C];

BEGIN INTEGER I, K, L, M;

INTEGER ARRAY TOUR[C];

MI = N;

FOR I := 0 STEP 1 UNTIL N DO TOUR[I]: = I;

FOR I := 0 STEP 1 UNTIL N DO;

BEGIN K := ENTIER(RANDOM(X)*M);

SOL[I]: = TOUR[K]; POSIT[SOL[I]]: = I;

TOUR[K]: = TOUR[I]; MI := M-1

END;

80000 BEGIN K := SOL[N]; L := SOL[O]; OPT := DIS[K, L];

90000 FOR I := 1 STEP 1 UNTIL N DO;

BEGIN K := L; L := SOL[I]; OPT := OPT + DIS[K, L] END;

91000 END STARTINGTOUR;

PROCEDURE REDUCTTOUR(X, POSIT, SOL, OPT);

INTEGER X; REAL OPT; INTEGER ARRAY POSIT, SOL[C];

BEGIN INTEGER I, J, K, L, M, CITY;

INTEGER ARRAY LINKS[C];

K := -1;

FOR I := 0 STEP 1 UNTIL N DO;

IF POST[I] = -1 THEN BEGIN K := K+1; LINKS[K] := I END;

92000 M := K; J := O;

FOR I := 0 STEP 1 UNTIL K DO;

BEGIN L := ENTIER(RANDOM(X)*M);

CITY := LINKS[L]; SOL[I] := CITY; POSIT[CITY] := J; J := J+1;

93000 LINKS[L] := LINKS[M]; M := M-1;

WHILE PRE[CITY] NEQ -1 DO;

94000 BEGIN CITY := PRE[CITY]; SOL[I] := CITY;

95000 POSIT[CITY] := J; J := J+1

96000 END;

97000 END;

98000 END REDUCTTOUR;
PROCEDURE CLOSEUP(GG,G,DIS,POS,NOD);

% CLOSINGUP CHECKS WHETHER CLOSING UP RESULTS IN A BETTER TOUR
REAL GG,G,DIS; INTEGER ARRAY POS,NOD[0];
BEGIN IF GG-DIS > G THEN
  BEGIN G:=GG-DIS;
  WRITE(POS[1],*,POS); WRITE(SOL[1],*,NOD)
END
END CLOSEUP;

PROCEDURE TOURIMPROVEMENT(T1,T2I,G,GG,POS,NOD);

TOUR IMPROVEMENT TAKES CARE OF THE ITERATION PROCESS ON LEVELS > 3 FOR X
% AND ON LEVELS > 2 FOR Y
INTEGER T1,T2I; REAL G,GG;
INTEGER ARRAY POS,NOD[0];
BEGIN INTEGER I,C,D,R,S,PTC,PTD,NUMCAND,CANDIDATE;
REAL MAXGG;
CANDIDATE:=1;
WHILE CANDIDATE > 0 AND GG > G DO
BEGIN NUMCAND:=0; MAXGG:=-50; CANDIDATE:=I:=-1;
WHILE I < N-1 AND NUMCAND < 5 DO
BEGIN I:=I+1; D:=NBCT2I,IJ;
IF DISCT2I,DJ < GG-G THEN BEGIN
R:=ABS(POSITCT2IJ-POSITCDJ);
S:=ABS(POSCT2IJ-POSCDJ);
IF D NEG THEN % R=1 OR R=N IMPLIES C-D IS A LINK IN SOL,
S=1 OR S=N IMPLIES B-D MAY NOT BE AN Y-LINK
IF R NEG 1 AND R NEG N THEN % S=1 OR S=N IMPLIES B-D IS A Y-LINK
BEGIN PTD:=POSCDJ; PTC:=(PTD+W) MOD M; C:=NODCPTCJ;
R:=ABS(POSITCCJ-POSITCDJ);
% R=1 OR R=N IMPLIES C-D IS A LINK IN SOL AND THEREFORE C-D MAY BE BROKEN
IF R = 1 OR R = N THEN
BEGIN NUMCAND:=NUMCAND+1;
END
IF GG:::G THEN
BEGIN REVERSE(POS,NOD,PT2,PTC,V);
CLOSEUP(GG,G,DISCT1,CJ,POS,NOD);
T2I:=C
END
END
END
END
ELSE I:=N
END;
IF CANDIDATE GEO 0 THEN
BEGIN D:=CANDIDATE; PTD:=POSCDJ;
PTC:=(PTD+W) MOD M; C:=NODCPTCJ;
GG:=GG+MAXGG;
IF GG > G THEN
BEGIN REVERSE(POS,NOD,PT2,PTC,V);
CLOSEUP(GG,G,DISCT1,CJ,POS,NOD);
END;
T2I:=C
END
END
END TOURIMPROVEMENT;
IF REDUCTION THEN REDUCTTOUR(X, POSIT, SOL, OPT)
ELSE STARTINGTOUR(X, POSIT, SOL, OPT);
SORTSEQ(N, SEQ);
NULL:=NUM:=PT1:=-1; M:=N+1;
WHILE NUM < N DO
BEGIN NUM:=NUM+1;
T1:=SEQ[NUM]; PT1:=POSIT[T1];
BEGIN IT2:=NT3:=0; W:=M-V;
PT2:=(PT1+V) MOD M; T2:=SOL[PT2];
DO BEGIN G1:=0; GG:=DIS[T1, T2];
T3:=FINDLINKS(IT2, POSIT, T1, T2, GG, N3, NT3, SOL);
IF T3 NEQ NULL THEN
BEGIN PT4:=(POSIT[T3]+W) MOD M; T4:=SOL[PT4];
WRITE(POS1, *, POSIT); WRITE(NOD1, *, SOL);
GG1:=GG-DIS[T2, T3]+DIS[T3, T4];
REVERSE(POS1, NOD1, PT2, PT4, V);
CLOSEUP(GG1, G, DIS[T1, T4], POS1, NOD1);
END
END
DO BEGIN IT4:=NT5:=0; W:=M-V;
PT3:=(PT1+V) MOD M; T3:=SOL[PT3];
DO BEGIN G1:=DIS[T1, T2]-DIS[T2, T3]+DIS[T3, T4];
T5:=FINDLINKS(IT4, POSIT, T1, T4, GG, NS, NT5, NOD1);
IF T5 NEQ NULL THEN
BEGIN PT6:=NEG NILL THEN
BEGIN T6:=NEG NOD1[PT6];
HU:=GG-DIS[T4, T5]+DIS[T5, T6];
IF HU NEQ G THEN
BEGIN WRITE(POS2, *, POS1); WRITE(NOD2, *, NOD1);
REVERSE(POS2, NOD2, PT2, PT6, V);
CLOSEUP(HU, G, DIS[T6, T1], POS2, NOD2);
TOURIMPROVEMENT(T1, T6, G, HU, POS2, NOD2);
END
END
UNTIL T5 = NULL OR G > 0;
THE ALTERNATE CHOICE FOR T4 %
IF G = 0 THEN
BEGIN PT3:=POSIT[PT3]; PT4:=(PT3+V) MOD M; T4:=SOL[PT4];
DO BEGIN GG:=DIS[T1, T2]-DIS[T2, T3]+DIS[T3, T4];
T5:=FINDLINKS(IT4, POSIT, T1, T4, GG, N5, NT5, SOL);
IF T5 NEQ NULL THEN
BEGIN PT5:=POSIT[PT5];
IF T4 THEN BOOL:= V=N ELSE IF (PT4 < PT1 AND PT5 > PT4 AND PT5 LEQ PT1)
THEN BOOL:=TRUE
END
ELSE IF (PT4 > PT1 AND (PT5 > PT4 OR PT5 LEQ PT1))
THEN BOOL:=TRUE ELSE BOOL:=FALSE;
IF \( B \) \( = \) 1 THEN 
BEGIN 
\( PT6 := (PT5 + W) \mod M \)
\( T6 := \text{SOL}(\text{PT6}) \)
\( X \)
\( T5 \) LIES BETWEEN \( T1 \) AND \( T4 \), DETERMINATION OF \( T7 \)
\( T7 := \text{FIND}(\text{NOD1, POS1, POSIT, POSIT, T1, T6, GG}) \)
IF \( T7 \) \( \neq \) NULL THEN 
BEGIN 
\( PT7 := \text{POS1}(T7) \)
IF \( (PT2 < PT3 \text{ AND } PT7 \geq PT2 \text{ AND } PT7 \leq PT3) \) THEN \( BOOLl := \text{TRUE} \)
ELSE IF \( (PT2 > PT3 \text{ AND } (PT7 \geq PT2 \text{ OR } PT7 \leq PT3)) \) THEN \( BOOLl := \text{TRUE} \)
ELSE \( BOOLl := \text{FALSE} \)
ENDIF 
\( \text{FOUND} := BOOLl = 1 \)
ENDIF 
\( \text{IF} \) \( \text{FOUND} \) \( \text{THEN} \)
BEGIN 
\( T1 := \text{SOL}(\text{PT7} + V) \mod M \)
\( T2 := \text{SOL}(\text{PT7} + W) \mod M \)
\( \text{X} \)
\( \text{DETERMINATION OF} \ T8 \)
\( \text{IF} \) \( (PT2 \leq PT3 \text{ AND } (PT7 \geq PT2 \text{ OR } PT7 \leq PT3)) \) \( \text{THEN} \) \( 
\text{IF} \) \( (T1 \leq T7 \text{ OR } T1 \leq T2) \) \( \text{AND} \) \( (T7 \leq T3 \text{ OR } T7 \leq T4) \) \( \text{THEN} \)
\( \text{BOOL} := \text{TRUE} \)
ELSE \( \text{BOOL} := \text{FALSE} \)
ENDIF 
\( \text{IF} \) \( \text{BOOL} = \text{TRUE} \) \( \text{THEN} \)
BEGIN 
\( T5 := \text{SOL}(\text{PT7} + V) \mod M \)
\( T6 := \text{SOL}(\text{PT7} + W) \mod M \)
\( \text{X} \)
\( \text{IF} \) \( \text{DIS}(T7, T1) \text{ - DIS}(T1, T2) > \text{DIS}(T7, T2) \text{ - DIS}(T2, T1) \text{ THEN} \)
\( T5 := T1 \text{ ELSE} \)
\( T5 := T2 \text{ IF} \)
\( T5 \) \( \neq \) NULL \( \text{THEN} \)
BEGIN 
\( \text{WRITE}(\text{POS1}, *, \text{POSIT}) \text{ WRITE}(\text{NOD1}, *, \text{SOL}) \)
\( \text{IF} \) \( T5 = T1 \) \( \text{THEN} \)
\( T5 := T2 \text{ ELSE} \)
\( T5 := T1 \)
ENDIF 
\( \text{CLOSEUP}(GG, \text{DIS}(T5, T1), T1 \text{ POS1, NOD1, POSIT), \text{DIS}(T5, T2), T2 \text{ POS1, NOD1, POSIT}) \)
\( \text{TOUR}(\text{PROV}(T1, T5, GG, GG, POS1, NOD1)) \)
375000 \text{M:=N+1;}
376000 \text{FOR I:=0 STEP 1 UNTIL N DO SORTDISTANCES(N,DISC[I,\cdot],NB[I,\cdot]);}
377000 \text{OLDSOL:=REDUCTION:=FALSE;}
378000 \text{FOR TOURS:=1 STEP 1 UNTIL NUMTOURS DO}
379000 \text{BEGIN KERNIGHANLIN(X,OPT,POSIT,SOL);}
380000 \text{\quad IF NOT OLDSOL THEN}
381000 \text{\quad BEGIN NUMSOL:=NUMSOL+1; OPTIMUM[NUMSOL]:=OPT;}
382000 \text{\quad \quad IF NUMSOL > 2 THEN REDUCTION:=TRUE;}
383000 \text{\quad \quad WRITE(SOLUTIONS[NUMSOL,\cdot,\cdot,SOL]);}
384000 \text{\quad IF NUMSOL = 1}
385000 \text{\quad \quad THEN BEGIN K:=SOL[0]; L:=SOL[N];}
386000 \text{\quad \quad PRE[K]:=L; POST[L]:=K;}
387000 \text{\quad \quad FOR I:=1 STEP 1 UNTIL N DO}
388000 \text{\quad \quad BEGIN L:=SOL[I]; PRE[L]:=K; POST[K]:=L; K:=L END}
389000 \text{\quad END}
390000 \text{\quad ELSE FOR I:=1 STEP 1 UNTIL NUMSOL-1 DO}
391000 \text{\quad \quad FOR R:=0 STEP 1 UNTIL N DO}
392000 \text{\quad \quad BEGIN K:=POSIT[R];}
393000 \text{\quad \quad \quad J:=(K+N) MOD M; L:=(K+1) MOD M;}
394000 \text{\quad \quad \quad PRE[K]:=J; POST[L]:=K;}
395000 \text{\quad \quad \quad IF PRE[K] NEQ J AND PRE[K] NEQ L THEN PRE[K]:=1;}
396000 \text{\quad \quad \quad IF POST[K] NEQ J AND POST[K] NEQ L THEN POST[K]:=1;}
397000 \text{\quad END;}
398000 \text{\quad END;}
399000 \text{\quad END;}
400000 \text{\quad END \quad LINKERNIGHAN;}

\text{\textcircled{}}