A program for the traveling-salesman problem, according to the heuristic algorithm of Lin-Kernighan

EINDHOVEN UNIVERSITY OF TECHNOLOGY

Department of Mathematics

PROBABILITY THEORY, STATISTICS AND OPERARIONS RESEARCH GROUP

Memorandum COSOR 77-12

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, look-ahead 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 i 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 sortingprocess 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 startingtour

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

closeup : a procedure which tests whether an actual tour is a better tour than the best tour found sofar in kernighanlin

tourimprovement: 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:

	Size	Source	Frequency of optimum		cpu-time/tour		*
			Pre-reduction	Post-reduction	Pre	Post	
1	33	[3]	0.80	0.90	0.38	0.38	2
2	42	[1]	1.00	1.00	2.14	0.99	1
3	57	[3]	0.23	0.23	7.3	5.3	6
4	100	[4]	0.50	0.50	8.2	7.1	4

* average number of distinct solutions

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

	Size	Source	Frequency of optimum		cpu-time/tour		*
			Pre-reduction	Post-reduction	Pre	Post	
1	33	[3]	1.00	-	0.8		1
2	42	[1]	1.00	-	2.6		1
3	57	[3]	0.27	0.43	8.2	3.8	3
4	100	[4]	0.56	0.63	17.0	5.5	3.5

* 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 check-out 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 startingtour 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 startingtour 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 startingtours for the post-reduction tours. The computing results for this alternative are:

	Size	Source	Frequency of optimum		cpu-time/tour		*
			Pre-reduction	Post-reduction	Pre	Post	
1	33	[3]					
2	42	[1]					
3	57	[3]	0.33	0.23	8.7	2.5	5
4	100	[4]	0.71	0.92	10.1	4.1	4

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

[1] G.B. Dantzig, D.R. Fulkerson, S.M. Johnson: "Solution of a Large-Scale Traveling-Salesman Problem", Opns. Res. 2, 393-410 (1954).

[2] B.W. Kernighan, S. Kin: "An Effective Heuristic Algorithm for the Traveling-Salesman Problem", Opns. Res. 21, 498-516 (1973).

[3] R.L. Karg, G.L. Thompson: "A Heuristic Approach to Solving Traveling-Salesman Problems", Management Sci. 10, 225-247 (1964).

[4] P. Krolak, W. Felts, G. Marble: "A Man-Machine Approach toward the Traveling-Salesman Problem", CACM 14, 327-334 (1971).

```
1000      PROCEDURE LINKERNIGHAN(N,X,DIS,NUMTOURS,OUT);
2000      VALUE N,NUMTOURS; INTEGER N,NUMTOURS,X; FILE OUT;
3000      REAL ARRAY DIS[0,0];
4000      BEGIN INTEGER I,J,K,L,M,R,NUMSOL,TOURS; REAL OPT;
5000          BOOLEAN OLDSOL,REDUCTION;
6000          INTEGER ARRAY POSIT,SOL,PRE,POST,SEQ[0:N], NB[0:N,0:N],
7000                        SOLUTIONS[1:NUMTOURS,0:N];
8000          REAL ARRAY OPTIMUM[1:NUMTOURS];
9000
10000
11000      PROCEDURE SORTDISTANCES(N,A,P);
12000      VALUE N; INTEGER N; INTEGER ARRAY P[0]; REAL ARRAY A[0];
13000      BEGIN INTEGER ARRAY C[0:N];
14000
15000          PROCEDURE SORT(L,R);
16000          VALUE L,R; INTEGER L,R;
17000          IF R-L > 1
18000          THEN BEGIN INTEGER I,K,IL,IR;
19000               K:=(L+R)/2; SORT(L,K); IR:=K+1; SORT(IR,R); I:=IL:=L;
20000               WHILE IL LEQ K AND IR LEQ R DO
21000               IF A[P[IL]] LEQ A[P[IR]]
22000               THEN BEGIN C[I]:=P[IL]; IL:=IL+1; I:=I+1 END
23000               ELSE BEGIN C[I]:=P[IR]; IR:=IR+1; I:=I+1 END;
24000               IF IR > R
25000               THEN WHILE I LEQ R DO
26000                    BEGIN C[I]:=P[IL]; IL:=IL+1; I:=I+1 END
27000               ELSE WHILE I LEQ R DO
28000                    BEGIN C[I]:=P[IR]; IR:=IR+1; I:=I+1 END;
29000               I:=L-1;
30000               WHILE I:=I+1 LEQ R DO P[I]:=C[I]
31000               END
32000          ELSE IF R = L
33000          THEN P[R]:=R
34000          ELSE IF A[L] > A[R]
35000          THEN BEGIN P[L]:=R; P[R]:=L END
36000          ELSE BEGIN P[L]:=L; P[R]:=R END;
37000
38000          SORT(0,N)
39000      END SORTDISTANCES;
40000
41000      PROCEDURE SORTSEQ(N,SEQ);
42000      INTEGER N; INTEGER ARRAY SEQ[0];
43000      BEGIN INTEGER I,K,L; REAL F; INTEGER ARRAY PLACE[0:N];
44000          K:=SOL[N]; L:=SOL[0];
45000          FOR I:=1 STEP 1 UNTIL N DO
46000          BEGIN F:=DIS[K,L]; K:=L; L:=SOL[I]; F:=MAX(F,DIS[K,L]);
47000              F:=F-DIS[K,NB[K,0]];
48000              PLACE[K]:=-F
49000          END;
50000          K:=SOL[N];
51000          PLACE[N]:=-MAX(DIS[K,SOL[N-1]],DIS[K,SOL[0]]);
52000          PLACE[N]:=*+DIS[K,NB[K,0]];
53000          SORTDISTANCES(N,PLACE,SEQ)
54000      END SORTSEQ;
```

```
56000      PROCEDURE KERNIGHANLIN(X,OPT,POSIT,SOL);
57000  % KERNIGHANLIN DETERMINES ONE LOCAL-OPTIMAL SOLUTION %
58000      INTEGER X; REAL OPT; INTEGER ARRAY POSIT,SOL[0];
59000      BEGIN INTEGER T1,T2,T3,T4,T5,T6,T7,T8,NT3,NT5,I1,I2,IT2,IT4,IT6,
60000                    NILL,NUM,PT,I,J,K,L,M,R,S,V,W,
61000                    PT1,PT2,PT3,PT4,PT5,PT6,PT7,PT8;
62000          REAL G,GG,HU,F,MAXGG;
63000          BOOLEAN FOUND,BOOL;
64000          INTEGER ARRAY POS1,POS2,NOD1,NOD2,POSIT1,SOL1[0:N], N3,N5[1:5];
65000
66000
67000      PROCEDURE STARTINGTOUR(X,POSIT,SOL,OPT);
68000      INTEGER X; REAL OPT; INTEGER ARRAY POSIT,SOL[0];
69000      BEGIN INTEGER I,K,L,M;
70000          INTEGER ARRAY TOUR[0:N];
71000          M:=N;
72000          FOR I:=0 STEP 1 UNTIL N DO TOUR[I]:=I;
73000          FOR I:=0 STEP 1 UNTIL N DO
74000          BEGIN K:=ENTIER(RANDOM(X)*M);
75000              SOL[I]:=TOUR[K]; POSIT[SOL[I]]:=I;
76000              TOUR[K]:=TOUR[M]; M:=M-1
77000          END;
78000          K:=SOL[N]; L:=SOL[0]; OPT:=DIS[K,L];
79000          FOR I:=1 STEP 1 UNTIL N DO
80000          BEGIN K:=L; L:=SOL[I]; OPT:=OPT+DIS[K,L] END
81000      END STARTINGTOUR;
82000
83000      PROCEDURE REDUCTTOUR(X,POSIT,SOL,OPT);
84000      INTEGER X; REAL OPT; INTEGER ARRAY POSIT,SOL[0];
85000      BEGIN INTEGER I,J,K,L,M,CITY;
86000          INTEGER ARRAY LINKS[0:N];
87000          K:=-1;
88000          FOR I:=0 STEP 1 UNTIL N DO
89000          IF POST[I] = -1 THEN BEGIN K:=K+1; LINKS[K]:=I END;
90000          M:=K; J:=0;
91000          FOR I:=0 STEP 1 UNTIL K DO
92000          BEGIN L:=ENTIER(RANDOM(X)*M);
93000              CITY:=LINKS[L]; SOL[J]:=CITY; POSIT[CITY]:=J; J:=J+1;
94000              LINKS[L]:=LINKS[M]; M:=M-1;
95000              WHILE PRE[CITY] NEQ -1 DO
96000              BEGIN CITY:=PRE[CITY]; SOL[J]:=CITY;
97000                  POSIT[CITY]:=J; J:=J+1
98000              END;
99000          END;
100000          K:=SOL[N]; L:=SOL[0]; OPT:=DIS[K,L];
101000          FOR I:=1 STEP 1 UNTIL N DO
102000          BEGIN K:=L; L:=SOL[I]; OPT:=OPT+DIS[K,L] END;
103000       END REDUCTTOUR;
```

```
164000      PROCEDURE CLOSEUP(GG,G,DIS,POS,NOD);
165000 % CLOSINGUP CHECKS WHETHER CLOSING UP RESULTS IN A BETTER TOUR %
166000      REAL GG,G,DIS; INTEGER ARRAY POS,NOD[0];
167000      BEGIN IF GG-DIS > G THEN
168000          BEGIN G:=GG-DIS;
169000              WRITE(POSIT1,*,POS); WRITE(SOL1,*,NOD)
170000          END
171000      END CLOSEUP;
172000
173000      PROCEDURE TOURIMPROVEMENT(T1,T2I,G,GG,POS,NOD);
174000 % TOURIMPROVEMENT TAKES CARE OF THE ITERATIONPROCESS ON LEVELS > 3 FOR X
175000 % AND ON LEVELS > 2 FOR Y %
176000      INTEGER T1,T2I; REAL G,GG; INTEGER ARRAY POS,NOD[0];
177000      BEGIN INTEGER I,C,D,R,S,PTC,PTD,NUMCAND,CANDIDATE;
178000          REAL MAXGG;
179000          CANDIDATE:=1;
180000          WHILE CANDIDATE > 0 AND GG > G DO
181000          BEGIN NUMCAND:=0; MAXGG:=-@50; CANDIDATE:=I:=-1;
182000              WHILE I < N-1 AND NUMCAND < 5 DO
183000              BEGIN I:=I+1; D:=NB[T2I,I];
184000                  IF DIS[T2I,D] < GG-G
185000                  THEN BEGIN R:=ABS(POSIT[T2I]-POSIT[D]);
186000                       S:=ABS(POS[T2I]-POS[D]);
187000                       IF D NEQ T1 THEN
188000 % R=1 OR R= IMPLIES B-D IS A LINK OF THE TOUR IN SOL,
189000 % THEREFORE B-D MAY NOT BE AN Y-LINK
190000                       IF R NEQ 1 AND R NEQ N THEN
191000 % S=1 OR S=N IMPLIES B-D IS A Y-LINK %
192000                       IF S NEQ 1 AND S NEQ N THEN
193000                       BEGIN PTD:=POS[D]; PTC:=(PTD+W) MOD M; C:=NOD[PTC];
194000                           R:=ABS(POSIT[C]-POSIT[D]);
195000 % R=1 OR R=N IMPLIES CD IS A LINK IN SOL AND THEREFORE C-D MAY BE BROKEN
196000                           IF R = 1 OR R = N THEN
197000                           BEGIN NUMCAND:=NUMCAND+1;
198000                               IF NOT REDUCTION OR
199000                               (PRE[D] NEQ C AND POST[D] NEQ C) THEN
200000                               IF DIS[D,C]-DIS[D,T2I] > MAXGG THEN
201000                               BEGIN MAXGG:=DIS[D,C]-DIS[D,T2I];
202000                                   CANDIDATE:=D
203000                               END
204000                           END
205000                       END
206000                       END
207000                  ELSE I:=N
208000              END;
209000              IF CANDIDATE GEQ 0 THEN
210000              BEGIN D:=CANDIDATE; PTD:=POS[D];
211000                  PTC:=(PTD+W) MOD M; C:=NOD[PTC];
212000                  GG:=GG+MAXGG;
213000                  IF GG > G THEN
214000                  BEGIN REVERSE(POS,NOD,PT2,PTC,V);
215000                      CLOSEUP(GG,G,DIS[T1,C],POS,NOD);
216000                  END;
217000                  T2I:=C
218000              END
219000          END
220000      END TOURIMPROVEMENT;
```

```
222000      IF REDUCTION THEN REDUCTTOUR(X,POSIT,SOL,OPT)
223000                   ELSE STARTINGTOUR(X,POSIT,SOL,OPT);
224000      SORTSEQ(N,SEQ);
225000      NILL:=NUM:=PT1:=-1; M:=N+1;
226000      WHILE NUM < N DO
227000      BEGIN NUM:=NUM+1;
228000          T1:=SEQ[NUM]; PT1:=POSIT[T1];
229000 % V = 1 CORRESPONDS TO AN ORIENTATION TO THE RIGHT %
230000 % V = N CORRESPONDS TO AN ORIENTATION TO THE LEFT %
231000          FOR V:=1,N DO
232000          BEGIN IT2:=NT3:=0; W:=M-V;
233000              PT2:=(PT1+V) MOD M; T2:=SOL[PT2];
234000              DO BEGIN G:=0; GG:=DIS[T1,T2];
235000                 T3:=FINDLINKS(IT2,POSIT,T1,T2,GG,N3,NT3,SOL);
236000                 IF T3 NEQ NILL THEN
237000                 BEGIN PT4:=(POSIT[T3]+W) MOD M; T4:=SOL[PT4];
238000                     WRITE(POS1,*,POSIT); WRITE(NOD1,*,SOL);
239000                     GG:=GG-DIS[T2,T3]+DIS[T3,T4];
240000                     REVERSE(POS1,NOD1,PT2,PT4,V);
241000                     CLOSEUP(GG,G,DIS[T1,T4],POS1,NOD1);
242000                     IT4:=NT5:=0;
243000                    DO BEGIN T5:=FINDLINKS(IT4,POS1,T1,T4,GG,N5,NT5,NOD1);
244000                        IF T5 NEQ NILL THEN
245000                        BEGIN PT6:=(POS1[T5]+W) MOD M; T6:=NOD1[PT6];
246000                            HU:=GG-DIS[T4,T5]+DIS[T5,T6];
247000                            IF HU GEQ G THEN
248000                            BEGIN WRITE(POS2,*,POS1); WRITE(NOD2,*,NOD1);
249000                                REVERSE(POS2,NOD2,PT2,PT6,V);
250000                                CLOSEUP(HU,G,DIS[T6,T1],POS2,NOD2);
251000                                TOURIMPROVEMENT(T1,T6,G,HU,POS2,NOD2);
252000                            END
253000                        END
254000                        END
255000                     UNTIL T5 = NILL OR G > 0;
256000 % THE ALTERNATE CHOICE FOR T4 %
257000      IF G = 0 THEN
258000      BEGIN PT3:=POSIT[T3]; PT4:=(PT3+V) MOD M; T4:=SOL[PT4];
259000          IT4:=NT5:=0;
260000          DO BEGIN GG:=DIS[T1,T2]-DIS[T2,T3]+DIS[T3,T4];
261000             T5:=FINDLINKS(IT4,POSIT,T1,T4,GG,N5,NT5,SOL);
262000             IF T5 NEQ NILL THEN
263000             BEGIN PT5:=POSIT[T5];
264000                 IF T1 = T4 THEN BOOL:= V=N ELSE
265000                 IF (PT4 < PT1 AND PT5 > PT4 AND PT5 LEQ PT1)
266000                 THEN BOOL:=TRUE
267000                 ELSE IF (PT4 > PT1 AND (PT5 > PT4 OR PT5 LEQ PT1))
268000                 THEN BOOL:=TRUE ELSE BOOL:=FALSE;
```

```
269000                  IF BOOL EQV V = 1
270000                  THEN BEGIN PT6:=(PT5+W) MOD M; T6:=SOL[PT6];
271000 % T5 LIES BETWEEN T1 AND T4, DETERMINATION OF T7
272000                       GG:=GG-DIS[T4,T5]+DIS[T5,T6];
273000                       IT6:=T7:=0; FOUND:=FALSE;
274000                       WHILE NOT FOUND AND T7 NEQ NILL DO
275000                       BEGIN T7:=FINDLINK(N,IT6,NB[T6,*],DIS[T6,*],
276000                                          POSIT,POSIT,T1,T6,GG);
277000                           IF T7 NEQ NILL THEN
278000                           BEGIN PT7:=POSIT[T7];
279000                           IF (PT2 < PT3 AND PT7 GEQ PT2 AND PT7 LEQ PT3)
280000                               THEN BOOL:=TRUE
281000                               ELSE
282000                           IF (PT2>PT3 AND (PT7 GEQ PT2 OR PT7 LEQ PT3))
283000                               THEN BOOL:=TRUE ELSE BOOL:=FALSE;
284000                               FOUND:= BOOL EQV V = 1
285000                           END
286000                       END;
287000
288000                       IF FOUND THEN
289000                       BEGIN I1:=SOL[(PT7+V) MOD M];I2:=SOL[(PT7+W) MOD M];
290000 %DETERMINATIONOF T8 %
291000                           IF DIS[T7,I1]-DIS[T1,I1] > DIS[T7,I2]-DIS[T1,I2]
292000                           THEN T8:=I1 ELSE T8:=I2;
293000                           IF T8 = T1 OR T8 = T2 OR T8 = T4 THEN
294000                           T8:= IF T8 = I1 THEN I2 ELSE I1;
295000                           GG:=GG-DIS[T6,T7]+DIS[T7,T8];
296000                           IF GG > 0 THEN
297000                           BEGIN WRITE(POS1,*,POSIT); WRITE(NOD1,*,SOL);
298000                               IF T8 = I1
299000                               THEN BEGIN PT8:=(PT7+V) MOD M;
300000                                    REVERSE(POS1,NOD1,PT4,PT6,V);
301000                                    REVERSE(POS1,NOD1,PT2,PT7,V);
302000                                    REVERSE(POS1,NOD1,PT8,PT3,V);
303000                                    REVERSE(POS1,NOD1,PT2,PT3,V);
304000                                    END
305000                               ELSE BEGIN PT8:=(PT7+W) MOD M;
306000                                    REVERSE(POS1,NOD1,PT2,PT8,V);
307000                                    REVERSE(POS1,NOD1,PT7,PT3,V);
308000                                    REVERSE(POS1,NOD1,PT4,PT6,V);
309000                                    END;
310000                               CLOSEUP(GG,G,DIS[T8,T1],POS1,NOD1);
311000                               TOURIMPROVEMENT(T1,T8,G,GG,POS1,NOD1);
312000                           END
313000                       END
314000                       END
```

```
375000      M:=N+1;
376000      FOR I:=0 STEP 1 UNTIL N DO SORTDISTANCES(N,DIS[I,*],NB[I,*]);
377000      OLDSOL:=REDUCTION:=FALSE;
378000      FOR TOURS:=1 STEP 1 UNTIL NUMTOURS DO
379000      BEGIN KERNIGHANLIN(X,OPT,POSIT,SOL);
380000         IF NOT OLDSOL THEN
381000         BEGIN NUMSOL:=NUMSOL+1; OPTIMUM[NUMSOL]:=OPT;
382000               IF NUMSOL > 2 THEN REDUCTION:=TRUE;
383000               WRITE(SOLUTIONS[NUMSOL,*],*,SOL);
384000               IF NUMSOL = 1
385000               THEN BEGIN K:=SOL[0]; L:=SOL[N];
386000                    PRE[K]:=L; POST[L]:=K;
387000                    FOR I:=1 STEP 1 UNTIL N DO
388000                    BEGIN L:=SOL[I]; PRE[L]:=K; POST[K]:=L; K:=L END
389000                    END
390000               ELSE FOR I:=1 STEP 1 UNTIL NUMSOL-1 DO
391000                    FOR R:=0 STEP 1 UNTIL N DO
392000                    BEGIN K:=POSIT[R];
393000                        J:=(K+N) MOD M; L:=(K+1) MOD M;
394000                        J:=SOL[J]; K:=SOL[K]; L:=SOL[L];
395000                        IF PRE[K] NEQ J AND PRE[K] NEQ L THEN PRE[K]:=-1;
396000                        IF POST[K] NEQ J AND POST[K] NEQ L THEN POST[K]:=-1
397000                    END;
398000         END
399000      END;
400000 END      LINKERNIGHAN;
```