A SET SEQUENCING HEURISTIC AND THE COMPUTER CODE FOR SOLVING THE TSP

By

O. E. Charles-Owaba and V.O. Oladokun
Department of Industrial and Production Engineering,
University of Ibadan, Ibadan, Nigeria.

ABSTRACT

This paper proposes a set sequencing heuristic solution for the travelling salesman problem (TSP). It attempts to first select, preferably a set of M smallest elements of the TSP matrix and then form a sequence. A computer code of the procedure was developed in Fortran 77 and used to examine its efficiency and relative effectiveness. It was found to be as effective as, but more efficient than the best of the nearest neighbour heuristics.

KEY WORDS: Heuristic solution, travelling salesman problem, computer code, nearest neighbour, set sequencing.

1. Introduction.

The problem of selecting the sequence of travelling through M cities to and from an origin in order to extremize some criteria has long received considerable attention [1, 2, 6, 7, 8, 10, 12, 13, 14]. Known as the travelling salesman problem (TSP), it naturally occurs in various forms in a number of industrial situations: the setting up of machines [2, 4], circuit assemply [3, 9], operations design [15], network distribution as well as the routing of transportation facilities [7], to mention just a few. In concise form, it may be expressed as:

$$\text{Minimize } Z = \sum_{k=2}^{M} CM_{[k-1],[k]} + CM_{[M],[1]} \tag{1}$$

where $[CM_{i,j}]$ or $[CM(i,j)]$ is between city cost, distance or time matrix and $CM_{[k-1],[k]}$, the value of travelling between the city in sequence positions $k-1$ and k.

Though easy to state, the TSP has long been recognized as one of the NP-hard problems [6, 13]. Simply put, NP-hard means a problem with no known clue for an efficient or polynomial time algorithm. An earlier claim of a polynomial time algorith for the TSP was recently proved wrong by Weixiong [19]. To date, only such implicit enumeration techniques as the branch and bound [2, 11], integer programming [2] and dynamic programming [2] have been used to optimally solved the TSP. However, they have been found impractical for the type of large size problems in industry [2, 4].

Consequently, practitioners resort to the use of heuristics for industrial problems. The earliest reported was the "nearest neighbour" family of heuristics [4, 10]. Because of the importance of the TSP in this era of computer integrated manufacturing (CIM), the list of heuristics continue to grow. More recently, the genetic [5, 16], tabu search [8] and simulated annealing [19] have been added.

Nevertheless, it is those whose relative efficiency and effectiveness are known with the availability of an easy-to-run computer code that may gain acceptance.

It is the main purpose of this paper to suggest a heuristic with an associated computer code. In particular, an attempt will be made to develop a set sequencing heuristic (SSH) coded in Fortran 77 language for easy application. Finally, its relative efficiency and effectiveness will be numerically examined. For now, the basis of the heuristic follows.

2. The Set Sequencing Heuristic (SSH)

Given a TSP matrix $CM(i,j)$ of size M, the SSH attempts to pack the set of, preferably, M smallest matrix elements (in relays) for constructing a complete sequence. Where partial sequences or cycles result, they are resolved into a complete sequence as presented in the following:

Resolution of Cycles

Consider the two partial cycles S_1^p, S_2^p to be joined into one, S

$$S_1^p = x_1 - x_2 - \ldots - x_i - x_{i+1} - \ldots - x_{n_1} - x_1 \tag{2}$$

$$S_2^p = y_1 - y_2 - \ldots - y_i - y_{i+1} - \ldots - y_{n_2} - y_1 \tag{3}$$

and

$$S = S_1^p + S_2^p \tag{4}$$

where
x_i = city in the first partial sequence position i and
y_i = city in second partial sequence position k.
Notice that, to obtain the sequence S, any pair of the elements $(x_i - x_{i+1},\ y_k - y_{k+1})$ have to be replaced with $(x_i - y_{k+1}, y_k - x_{i+1})$ as in the following:

$$S = x_i - y_{k+1} - \ldots - y_{n_2} - y_1 - y_2 - \ldots - y_k - S_{i+1} - \ldots - x_{n_1} - x_1 - x_2 - \ldots - x_i \tag{5}$$

To determine which pair to replace in order to join the partial cycles (S_1^p, S_2^p) to maximum advantage, one computes the value of replacing each pair with the function:

$$\begin{aligned} V(x_i, x_{i+1}; y_k, y_{k+1}) \;=\; & [CM(x_i, y_{k+1}) + CM(y_k, x_{i+1})] \\ & - [CM(x_i, x_{i+1}) + CM(y_k, y_{k+1})] \end{aligned}$$

For the $n_1 \times n_2$ such pairs, it is at the one with minimum value that S_1^p and S_2^p are joined to form S. Thus let $x_i^* - x_{i+1}^*$ and $y_k^* - y_{k+1}^*$ be such pair:

$$V(x_i^*, x_{i+1}^*; y_k^*, y_{k+1}^*) = \min\ V(x_i, x_{i+1}; y_k, y_{k+1}) \tag{7}$$

Now, joining at the links: $x_i^* - y_{k+1}^*$ and $y_k^* - x_{i+1}^*$,

$$S^* = x_i^* - y_{k+1}^* - y_{k+2} - \ldots - y_{n_2} - y_1 - y_2 - \ldots - y_k^* - x_{i+1}^* - x_{i+2} - \ldots - x_{n_1} - x_1 - x_2 - \ldots - x_i^* \quad (8)$$

Notice that S^* is the minimum out of the $n_1 \times n_2$ possible complete sequences. Hence, this approach to partial cycles resolution is a neighbourhood search procedure in its own right. For multiple cycles, they are joined in pairs until one complete sequence emerges.

The details of cycles resolution are presented in the flow chart of figure 1 while that of the complete SSH in figure 2.

To illustrate, consider the TSP matrix in table 1 taken from Baker [2]:

i \ j	1	2	3	4	5
1	-	4	8	6	8
2	5	-	7	11	13
3	11	6	-	8	4
4	5	7	2	-	2
5	10	9	7	5	-

Applying the procedure, the following elements were picked in the order:

$$4-3 \quad 1-2 \quad 3-5 \quad 5-4 \text{ and } 2-1$$

Attempting to form a sequence resulted in the partial cycles:

$$S_1^p = 3-5-4-3$$
$$S_2^p = 1-2-1$$

To resolve both into S, the following were computed:

$$V(3,5;1,2) = 6 \qquad V(3,5;2,1) = 17$$
$$V(5,4;1,2) = 6 \qquad V(5,4;2,1) = 11$$
$$V(4,3;1,2) = 9 \qquad V(4,3;2,1) = 5$$

$\min[V(x_i, x_{i+1}; y_k, y_{k+1})] = 5$
$\therefore \; x_i^* - x_{i+1}^* = 4-3; \;\; y_k^* - y_k^* - y_{k+1}^* = 2-1$
and
$S^* = 4-1-2-3-5-4; \;\; V(S^*) = 25$
In this case, the SSH also finds the optimal.

A computer code of the heuristic in Fortran 77 was written with the listing presented in the appendix. It consists of a main programme and a subroutine called Braker for partial cycles resolution using the principle of min $V(x_i, x_{i+1}; y_k, y_{k+1})$.

The main programme has three sections: input, a sorter and an output. The first section accepts the following inputs: problem size, M and TSP matrix, $[CM(i,j)]$. The sorter searches for the smallest M elements in relays and forms the sequence. The output

section computes the sequence value and then prints the sequence.

3. Computation Experience and Discussion

To examine the efficiency and relative effectiveness of the heuristic, two thousand problems with size ranging from 20 to 150 and between cities cost values: 5-50, 10-50 and 20-50 were generated randomly and solved first with the SSH, and then the nearest neighbour heuristic with variable origins (NNVB). The latter was the best reported by Gavet [10]; a PC computer, COMPAQ 486C2, was used.

Table 2: SSH Versus NNVB solution value (₦) and computational time (min)

	Problem		Solution value		Time (min)	
	Cost range	Size ( )	SSH	NNVB	SSH	NNVB
1	5-50	20	224	194	0.015	0.022
2	5-50	20	179	189	0.015	0.023
3	5-50	20	175	179	0.017	0.023
4	5-50	20	190	196	0.017	0.022
5	5-50	20	234	204	0.017	0.022
6	10-50	20	302	282	0.016	0.024
7	10-50	20	304	269	0.015	0.022
8	10-50	20	268	277	0.015	0.023
9	10-50	20	300	283	0.015	0.023
10	10-50	20	305	291	0.015	0.023
11	20-50	20	475	457	0.015	0.022
12	20-50	20	448	460	0.015	0.022
13	20-50	20	458	455	0.014	0.022
14	20-50	20	459	457	0.016	0.023
15	20-50	20	477	479	0.016	0.022
16	5-50	40	359	315	0.113	0.222
17	5-50	40	289	289	0.114	0.233
18	5-50	40	286	293	0.112	0.232
19	5-50	40	291	273	0.113	0.234
20	5-50	40	292	280	0.113	0.234
21	10-50	40	545	496	0.112	0.233
22	10-50	40	484	476	0.115	0.234
23	10-50	40	477	510	0.115	0.233
24	10-50	40	459	477	0.112	0.234
25	10-50	40	477	471	0.113	0.234
26	20-50	40	883	864	0.114	0.234
27	20-50	40	858	861	0.113	0.234
28	20-50	40	843	890	0.114	0.233

Table 2 (Contd.)

	Problem		Solution value		Time (min)	
	Cost range	Size ( )	SSH	NNVB	SSH	NNVB
29	20-50	40	851	867	0.112	0.234
30	20-50	40	850	862	0.113	0.234
31	5-50	60	443	428	0.470	1.080
32	5-50	60	419	388	0.472	1.081
33	5-50	60	407	390	0.472	1.080
34	5-50	60	411	384	0.473	1.081
35	5-50	60	385	394	0.470	1.081
36	20-50	60	1292	1281	0.472	1.081
37	20-50	60	1255	1256	0.480	1.081
38	20-50	60	1259	1269	0.472	1.081
39	20-50	60	1248	1260	0.476	1.082
40	5-50	80	503	531	1.376	3.291
41	5-50	80	502	511	1.376	3.290
42	5-50	80	489	515	1.376	3.291
43	5-50	80	494	520	1.376	3.291
44	5-50	80	525	519	1.374	3.293
45	10-50	80	895	908	1.376	3.292
46	10-50	80	910	882	1.371	3.292
47	10-50	80	890	902	1.370	3.292
48	10-50	80	880	898	1.377	3.289
49	10-50	80	949	891	1.370	3.290
50	20-50	100	2052	2066	3.216	7.888
51	5-50	100	629	629	3.209	7.882
52	5-50	100	617	617	3.208	7.879
53	5-50	100	609	610	3.208	7.882
54	5-50	100	611	594	3.210	7.883

The values of solutions and the computational times by the SSH and NNVB heuristics r a sample of fifty four problems are summarized in table 2. A plot of computational me (M) relative to problem size (N) for each heuristic is shown in figure 3.

Based on information from the 2000 solved problems, the SSH appears as good as e best nearest neighbour heuristics: 50% of the SSH solution values were smaller than ose of the NNVB and vice versa. However, on the computational time, the SSH heuris- : showed superiority over the NNVB (see table 2 and figure 3). The SSH time was nsistently smaller for all the problems solved.

Conclusion

In this paper a set sequencing heuristic was proposed and coded in Fortran computer nguage. To examine its efficiency and effectiveness, 2000 problems with size (M) rang- g between 20 and 150 were generated and solved with both the SSH and NNVB. The

former was found more efficient but was as effective as the latter. With an easy-to-run computer program, the heuristic may be easily adapted for a CIM situation.

References

1. Al-Haboub, Mohamad H and Selim, Shokrik, Z. "A sequencing problem in the weaving industry" European Journal of Operational Research (The Netherlands) vol.66, No 1 pp.65-71 April 1993.

2. Baker K.R. Introduction to Sequencing and Scheduling, John Wiley and Sons, Inc. New York, 1974.

3. Chan D. "Precedence constrained tsp applied to circuit board assemply and no wait flowshop". INT. Journal of Production Research Vol.31 no 9 pp.2171-2177, 1993.

4. Charles-Owaba O.E. and Lambert B.K "Sequence-dependent set-up times and similarity of parts: A mathematical model". IIE transaction Vol. 20 No. 1, March 1988.

5. Chartterjee S, Carea C and Lynch L.P. "Genetic Algorithm and tsp" European Journal of Operation Research Vol 93 No 3, Sept. 20, 1996.

6. Crowder, H. and Padberg M.W. "Solving large scale symmetric tsp to optimality". Management Science, Vol. 26 No. 5 pp 495-509.

7. Ferreir, J.V. "A travelling salesman model for the sequencing of duties in bus crew rotas". Journal of Operational Research Society (UK) Vol 46 No 4, pp 415-426 April 1995.

8. Fiechter, C.N. "A Parallel tabu search for algorithm for large travelling salesman problems". Discrete Applied Mathematics (The Netherlands) vol 51, No. 3, pp 243-267, July 1994.

9. Foulds, L.R. and Hamacher, H.W. "Optimal bin location and sequencing in printed circuit board assemply". European Journal of Operational Research (Netherland) 66, 3, 279-290, May 1993.

10. Gavett J. W. "Three Heuristic Rules for Sequencing Jobs to a Single Production Facility", Management Science, 11, 8, 1, 1965.

11. Little J.D.C., Murty K.G., Sweeney D.W. and Karel C. "An algorithm for the tssp" Operation Research Vol.11 pp 972-989 1963.

12. Madsen O.B.G. "An application of travelling salesman routine to solve pattern allocation problem in the glass industry" Journal of Operational Research Society (U.K) vol 39 No 3, pp249-256 March, 1988.

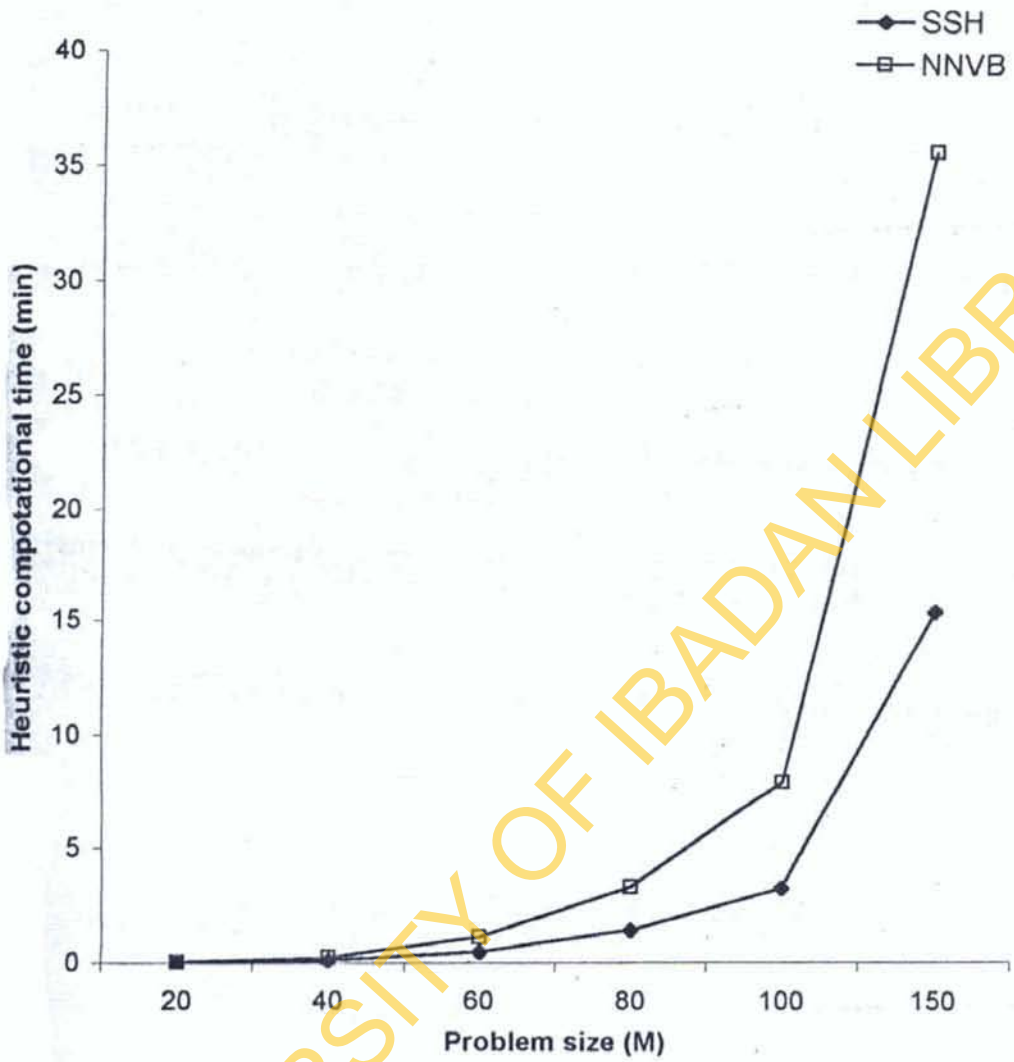

Figure 3: Heuristic computational time versus problem size (M)

13. Padberg M.W. and Hong, S. "On the Symmetric tsp: A computation study" Mathematical programming study 12 April 1980. North Holland Pub. Co. Amsterdam.

14. Padberg M.W. and Rinaldi G. "A branch Cut algorithm for the solution of large scale symmetric tsp" SIAM Review vol. 33, No. 1, pp 60-120, 1991.

15. Plante, R.D., Lowe, T.J. and Chandransekaren, R. "The Product matrix travelling salesman problem: An application and solution heuristic". Operations Research Vol. 35, No. 5, pp. 771-783, Oct. 1987.

16. Tian, P. and Yang, S. "An Improved Simulated Annealing Algorithm with generic characteristics and travelling salesman problem". Journal of information and Optimization Science, Vol.14, No.3, pp. 241-254, 1993.

17. Van Der Crusyssein, P. and Rijckaet, M.G. "Heuristic for the Asymmetric travelling salesman problem" Journal of Operational Research Society. Vol 29, No. 7, pp.677-701, 1978.

18. Webb, M.H.J. Some methods of producing solutions to travelling salesman problems with hundreds or thousand of cities. Operational Research Quarterly Vol. 22, No. 49-66.

19. Weixiong, Z. A note on the complexity of the Asymmetric TSP. Operations Research letters 20 (1997).

APPENDIX

Computer Program

```
PROGRAM HER1
    REAL CM(150,150),CM1(150,150)
    INTEGER TE(300),KIR(150),KIC(150),MR1(152),MV(4)
C    FILES  INPUT/OUTPUT
    OPEN (6,FILE= 'INP2.VIC')
     OPEN (7,FILE= 'H.V')
C
C          INPUT PROGRAM
C
    READ(6,01) M
 01  FORMAT(I3)
    DO 07 I=1,M
         DO 06 J=1,M
          READ(6,05) CM(I,J)
 05      FORMAT(F8.3)
       CM1(I,J)=CM(I,J)
 06    CONTINUE
 07   CONTINUE
C
C         MAIN PROGRAM
C
    K=1
    K4=0
    M2=M*2
    M7=1+M
    M8=M-2
C    SEARCHING   FOR THE LEAST ELT.
C
 03   DO 13 K3=1,M8
    KV=10000.0
    DO 12 I=1,M
    K5=0
    DO 8 L=1,K4
     IF(KIR(L).EQ.I)GO TO 12
 8    CONTINUE
    DO 10 J=1,M
    DO 9 L=1,K4
     IF(KIC(L).EQ.J)GO TO 10
 9    CONTINUE
    IF(CM(I,J).GE.KV)GO TO 10
    KV=CM(I,J)
    K1=J
    K5=1
 10   CONTINUE
    IF(K5.NE.1)GO TO 12
 11   K2=I
```

```
12  CONTINUE
    TE(K)=K2
    TE(K+1)=K1
    KIC(K3)=K1
    KIR(K3)=K2
    K=K+2
    K4=K3
    CM(K1,K2)=10000.0
13  CONTINUE
C
C     TO AVOID r      ITITION
    N=0
    DO 18 I=1
    DO 17 L=1,4
    IF(KIR(L).EQ.I)GO TO 18
17  CONTINUE
    N=N+1
    MV(N)=I
    IF(N.EQ.2) GO TO 19
18  CONTINUE
19  DO 21 I=1,M
    DO 20 L=1,K4
    IF(KIC(L).EQ.I) GO TO 21
20  CONTINUE
    N=N+1
    MV(N)=I
    IF(N.EQ.4)GO TO 22
21  CONTINUE
22  IF(MV(1).EQ.MV(3)) GO TO 26
    IF(MV(1).EQ.MV(4))GO TO 27
    IF(MV(2).EQ.MV(3))GO TO 27
    IF(MV(2).EQ.MV(4))GO TO 26
    KV=10000.0
    IK=0
    DO 31 I=1,2
    DO 31 J=3,4
    IK=IK+1
    IF(CM(MV(I),MV(J)).GE.KV) GO TO 31
    KK=IK
31  CONTINUE
    IF(KK.EQ.1.OR.KK.EQ.4)THEN
    TE(K)=MV(1)
    TE(K+1)=MV(4)
    TE(K+2)=MV(2)
    TE(K+3)=MV(3)
    ELSE
    TE(K)=MV(1)
    TE(K+1)=MV(3)
    TE(K+2)=MV(2)
    TE(K+3)=MV(4)
```

```
      ENDIF
      GO TO 25
   26  DO 32 I=1,2
      TE(K)=MV(I)
      TE(K+1)=MV(5-I)
      K=K+2
   32  CONTINUE
      GO TO 25
   27  DO 33 I=1,2
      TE(K)=MV(I)
      TE(K+1)=MV(I+2)
      K=K+2
   33  CONTINUE
   25  CALL BRAKER(TE,M2,M7,M,CM1,MR1)
      C3=0.0
      DO 2700 J=1,M
       C3=CM1(MR1(J),MR1(J+1))+C3
   2700  CONTINUE
      WRITE(7,34) (MR1(J),J=1,M)
   34  FORMAT(1X,25I3)
      WRITE(7,55) C3
   55  FORMAT(3X,'COST=',F7.2)
      STOP
      END

C
C          PROGRAME TO BREAK CYCLE
C
      SUBROUTINE BRAKER(TE,M2,M7,M,CM,MR1)
      INTEGER TE(M2),TER(300),MR1(M7),ITF(20),NE(6500),NY(6500)
      REAL CM(150,150),NCM(6500)
C
C    BRAKER
C
   1008 TER(1)=TE(1)
      TER(2)=TE(2)
      LF=0
      M2=M*2
      M3=M2-1
   1009 DO 2110 L = 2,M3,2
       DO2015 J = 3,M2,2
       IF(TER(L).NE.TE(J)) GO TO 2015
       TER(L+1)=TE(J)
       TER(L+2)=TE(J+1)
       K1=L+2
       GO TO 2110
   2015  CONTINUE
       GO TO 2120
   2110  CONTINUE
   2120 IF(K1.EQ.M2) GO TO 2130
```

```
      DO 2118 L=4,M2,2
        DO 2025 J=2,K1,2
        IF(TE(L).EQ.TER(J)) GO TO 2118
2025   CONTINUE
        KZ=L-1
        TER(K1+2)=TE(L)
        TER(K1+1)=TE(L-1)
        LF = LF+1
        ITF(LF)=K1
        KT=K1+2
        GO TO 2250
2118 CONTINUE
2250  DO 2210 L=KT,M2,2
        DO 2215 J=3,M2,2
          IF(J.EQ.KZ) GO TO 2215
          IF(TER(L).NE.TE(J)) GO TO 2215
          TER(L+1) = TE(J)
          TER(L+2) = TE(J+1)
          K1=L+2
          GO TO 2210
2215   CONTINUE
        GO TO 2120
2210  CONTINUE
2130  IF(LF.EQ.0) GO TO 3500
      ITF(LF+1)=M2
       DO 3015 LM = 1, LF
       K9 = 0
       IF = ITF(LM)
       IG = ITF(LM+1)
       IH = IF+1
       DO 3010 L = 2, IF, 2
        IZ = L-1
        DM1=CM(TER(IZ),TER(L))
         DO 3008 J = IH, IG, 2
          K9 = K9+1
          CM2=CM(TER(IZ),TER(J+1))+CM(TER(J),TER(L))
          NCM(K9)=CM2-DM1-CM(TER(J),TER(J+1))
          NE(K9)=L
          NY(K9)=J+1
3008   CONTINUE
3010   CONTINUE
C              BREAKING POINTS (LINKS)
        SM = 1000.00
        DO 3102 J=1, K9
          IF(NCM(J).GE.SM) GO TO 3102
          SM = NCM(J)
          K8 = J
3102   CONTINUE
        NEP = TER(NE(K8))
        TER(NE(K8))=TER(NY(K8))
```