T. C. Raymond

Heuristic Algorithm for the Traveling-salesman Problem

Abstract: The classical traveling-salesman problem is to determine a tour that will minimize the total distance or cost involved in visiting several cities and returning to the starting point. This paper describes a new heuristic algorithm that has been programmed for a digital computer and that obtains optimal or near-optimal solutions to the problem. The author's general approach was derived from an existing algorithm developed by Karg and Thompson in 1964. Computational results for five multi-city tours are presented and the algorithm is shown to be competitive with other existing heuristic techniques.

Introduction

One of the most intriguing problems encountered in operations research is the traveling-salesman problem. Simply stated, the problem is to find an optimal "tour" through n locations or cities that starts at one location, visits each of the $n - 1$ remaining locations once and only once and returns to the starting location. An optimal tour is defined to be a tour whose total distance or cost is a minimum.

To define the problem mathematically, we define a distance or cost matrix $D = (d_{ij})$. Each element d_{ij} represents the distance or cost of going from city i to city j. A solution to the problem is then a permutation $P(i_1, i_2, \cdots, i_n)$, where $i_1, i_2, \cdots, i_n$ are distinct integers with a range of 1 through n, that minimizes the quantity $d_{i_1 i_2} + d_{i_2 i_3} + \cdots + d_{i_{n-1} i_n} + d_{i_n i_1}$.

The difficulty in finding an optimal solution is evident from the fact that there are $(n - 1)!/2$ solutions to the symmetric ($d_{ij} = d_{ji}$) form of the problem. Nevertheless, there are several existing approaches to solving the traveling-salesman problem. A recent survey by Bellmore and Nemhauser[1] describes the state of the art.

Two exact solution techniques are known, dynamic programming and the branch-and-bound method. The primary disadvantages of dynamic programming are that it requires of the order of $(n - 1)(n - 2)2^{n-3}$ calculations and more than $(n - 1)2^{n-2}$ storage locations for an n-node problem.[2] Consequently, practical application of dynamic programming to the traveling-salesman problem is limited to tours with few cities. The branch-and-bound method can handle larger problems, but the amount of computing time is unpredictable and increases rapidly with the size of the problem. Sweeney et al.[3] stated that computer running time increases by approximately a factor of 10 for each increment of 10 in the number of nodes.

Many applications, however, do not demand optimal solutions. One trades optimality for reduced running time or storage. Several heuristic algorithms, which are fast and yield optimal or near-optimal solutions, exist for solving the traveling-salesman problem. Perhaps the best of these is the "3-opt" method developed by Lin.[4] Another, developed by Karg and Thompson,[5] obtains a single solution more rapidly, but the solutions in general are further from optimality. This paper describes extensions to the Karg-Thompson algorithm and computational results that reflect improvements in the algorithm. Additional related results are available in Ref. 6.

Basis for the algorithm

Karg and Thompson and also Lin developed two-phase algorithms. In each case the second phase uses the results of several first-phase solutions to obtain improved solutions. The same technique could be used with the author's algorithm.

A statement of the basic (first-phase) Karg-Thompson algorithm is the following:

1. Construct a permutation P of the nodes, e.g., $P = (i_1, i_2, i_3, \cdots, i_{n-1}, i_n)$.

2. Choose the first two nodes i_1 and i_2. The links represented by (i_1, i_2) and (i_2, i_1) form a tour or cyclic permutation. Set the number of links j equal to 2.

The author is located at the IBM Systems Development Division Laboratory, Poughkeepsie, New York 12602.

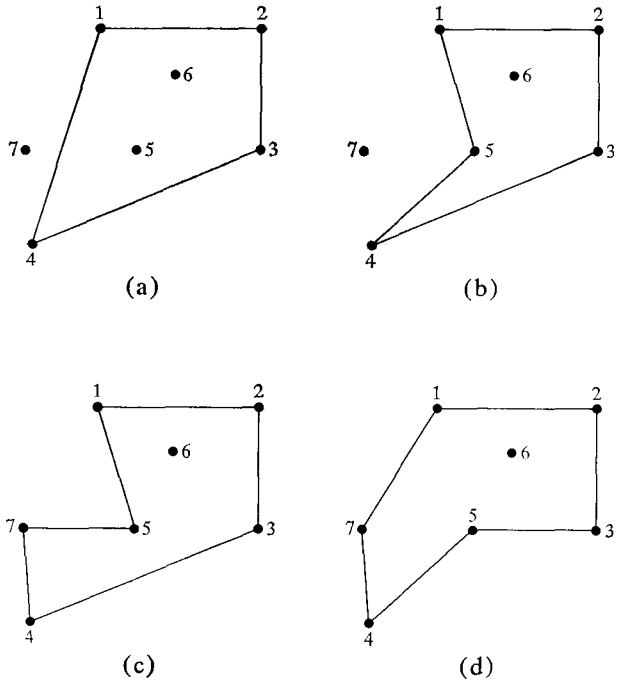

Figure 1 Seven-city example of the traveling-salesman problem: (a) incomplete tour, (b) starting mistake, (c) non-optimal tour and (d) corrected tour. Node 6 inserted between Nodes 1 and 2 completes the optimal tour.

3. Choose the next [the $(j + 1)$th] node k from the permutation P. For each link (p, q) in the tour, calculate using the distance matrix the quantity $I = d_{pk} + d_{kq} - d_{pq}$. This is the increment to the tour length that results if node k is inserted between nodes p and q.

4. Find the minimum I over all existing links (p, q). Call this increment I^* and call the link that produced the increment (p^*, q^*). Remove link (p^*, q^*) from the existing tour and produce a new tour by adding links (p^*, k) and (k, q^*). Increase j by 1.

5. If $j = n$, a solution exists; if $j < n$, repeat Steps 3 and 4.

Each distinct permutation chosen in the first step leads to a locally optimal solution. The number of operations required for a single solution is proportional to n^2 and many solutions can be obtained for large problems in a short time. By obtaining many solutions, each of which is locally optimal, one achieves a good probability of obtaining a globally optimal solution.

The author has found that better solutions can be obtained by deterministically choosing the next node to insert and by improving the existing tour at intermediate stages. The details can be understood by viewing the Karg-Thompson algorithm as follows:

1. An initial state is defined to be a tour through two nodes.

2. A transition is made to an intermediate state by including another node in the tour.

3. After $n - 2$ transitions are made, the resulting state represents a feasible solution.

The decision involved in the state-to-state transition requires two choices, which node to include and where to insert the chosen node in the existing tour. Karg and Thompson chose the next node from a randomly generated permutation and inserted it between the two nodes in the existing tour for which the smallest increment to the tour length resulted. The author, however, considered that a better decision could be made if all remaining nodes were evaluated in some way before making a transition.

By making increment calculations for all remaining nodes, one can in effect look ahead. One of several decision rules can then be applied. For example, one could choose the minimum of the best (minimum) increments for each node; that is, for the k nodes evaluated in each link (p, q) choose

$$\min_k [\min_{(p,q)} (d_{pk} + d_{kq} - d_{pq})].$$

Figure 1a shows four links of a seven-node tour; three decisions are yet to be made. The corresponding distance matrix $D = (d_{ij})$ is given in Fig. 2 and Table 1 lists the tour length increments. In this example the choice would be $\min\{4, 7, 5\} = 4$, which corresponds to inserting Node 5 between Nodes 4 and 1 in the existing tour. Another possibility would be to choose the node that produces the maximum of the minimum increments,

$$\max_k [\min_{(p,q)} (d_{pk} + d_{kq} - d_{pq})].$$

Applying this rule, we would choose Node 6 and insert it between Nodes 1 and 2.

What one really must seek is a criterion of "sureness" before making a decision. In other words, the question emerges: How can one be most sure that the chosen node is the right node? One answer is to look at not only the "best" increment for each node, but also the "next-best" increment. If these two criteria are nearly equal, one is not sure which link should be broken to insert the node. Referring to the example, we see that the best increment for Node 5 is 4 and the next-best is 6, yielding a difference of 2. Contrast this with Node 6 (a difference of $13 - 7 = 6$) and Node 7 (a difference of $26 - 5 = 21$). In qualitative terms, one draws the conclusion that he is surest about Node 7 and least sure about Node 5. Thus one chooses Node 7 to insert next and for this example the best final solution results.

The author has programmed several decision rules with varying degrees of success. The traveling-salesman problem is quite complex and, by looking ahead only one stage, one gets results analogous to those of the chess

	1	2	3	4	5	6	7
1	0	30	40	55	28	17	33
2	30	0	29	68	35	20	56
3	40	29	0	47	22	24	46
4	55	68	47	0	31	51	27
5	28	35	22	31	0	20	25
6	17	20	24	51	20	0	36
7	33	56	46	27	25	36	0

Figure 2 Distance matrix for the seven-city problem (arbitrary units).

Table 1 Tour-length increments for the seven-city problem (arbitrary units).

	Node		
Link	*5*	*6*	*7*
(1, 2)	33	7	59
(2, 3)	28	15	73
(3, 4)	6	28	26
(4, 1)	4	13	5
(4, 5)	—	40	21
(5, 1)	—	9	30

player who looks ahead only one move. To look ahead more than one stage, one must increase the number of calculations by an order of magnitude. The analogy with the chess player can be extended to show the difference between Karg and Thompson's decision rule and the author's. In the former case, one randomly chooses a piece to move, evaluates all (one-stage) moves for that piece and makes the best one. In the latter case, one evaluates all (one-stage) moves for all pieces and then makes a move.

The cost of making such evaluations in terms of computing time can be anticipated by determining the number of increment calculations. For a problem of n nodes, one makes $2(n-2)+3(n-3)+\cdots+(n-1)[n-(n-1)]$ calculations. Each term represents the number of links in the tour times the number of nodes that are not in the tour. The sum is $(n^3-7n+6)/6$. One can avoid redundant calculations by saving the increments for each node-link pair in a matrix. When two new links are created at each stage, the matrix can be updated by performing increment calculations involving each remaining node with only the two new links. Thus, instead of recalculating the same increments many times for nodes inserted at later stages, one simply scans the increment matrix. Running time remains proportional to n^3, but the number of calculations is reduced to $2(n-2)+2(n-3)+\cdots+2+[n-(n-1)]=(n-1)(n-2)$. This number is slightly less than twice the number of calculations made by Karg and Thompson.

Before describing tour-correction techniques, it should be pointed out that one can change decision rules between stages. Thus in the early stages of a solution one can use one rule and in later stages use another. This flexibility is important because the final solution is highly dependent on the early stages. From a practical standpoint the tour should take on in an early stage the general form of the final tour. This implies making an intelligent choice of the first several nodes, particularly the initial two. The author had little success in predicting which two initial nodes would lead to the best solution. Given the first two nodes, however, the author visualized that by using the *maximum* best-increment criterion for the first few nodes, the tour would include nodes located at corners of a polyhedron that encloses many nodes. The shape of this tour often approximates that of the final tour. On the other hand, if the *minimum* best-increment criterion is used for the first few nodes, the resulting tour is very short and encloses only a small area. Thus it cannot approximate the final tour and generally produces an inferior result. After several nodes are in the tour the transition decision can be based on the sureness criteria.

Correction techniques

With any decision rule based on a one-stage "look ahead," mistakes are made and nonoptimal solutions result. Ideally one should be able to examine a tour at any stage, determine whether it is the optimal tour and, if not, correct it. Obviously, if this could be accomplished one could completely solve the traveling-salesman problem by starting with any random tour through all nodes. Because this is not possible with currently known methods, the author sought partial correction techniques with the idea of correcting glaring mistakes.

The example introduced previously will suffice to illustrate a "mistake." Let us assume that Node 5 was inserted before Nodes 6 and 7. The resulting next stage is shown in Fig. 1b and the increment calculations for Nodes 6 and 7 are given in Table 1. Now let us assume that Node 7 is inserted next (see Fig. 1c). At this stage the tour is not optimal and without correction would lead to a nonoptimal solution. It is evident that Node 5 is in the wrong place in the tour; thus a mistake has been made that should be corrected before proceeding to the next stage.

The first correction technique developed by the author involves taking each node in turn out of the present tour, reevaluating its insertion increments and inserting the node

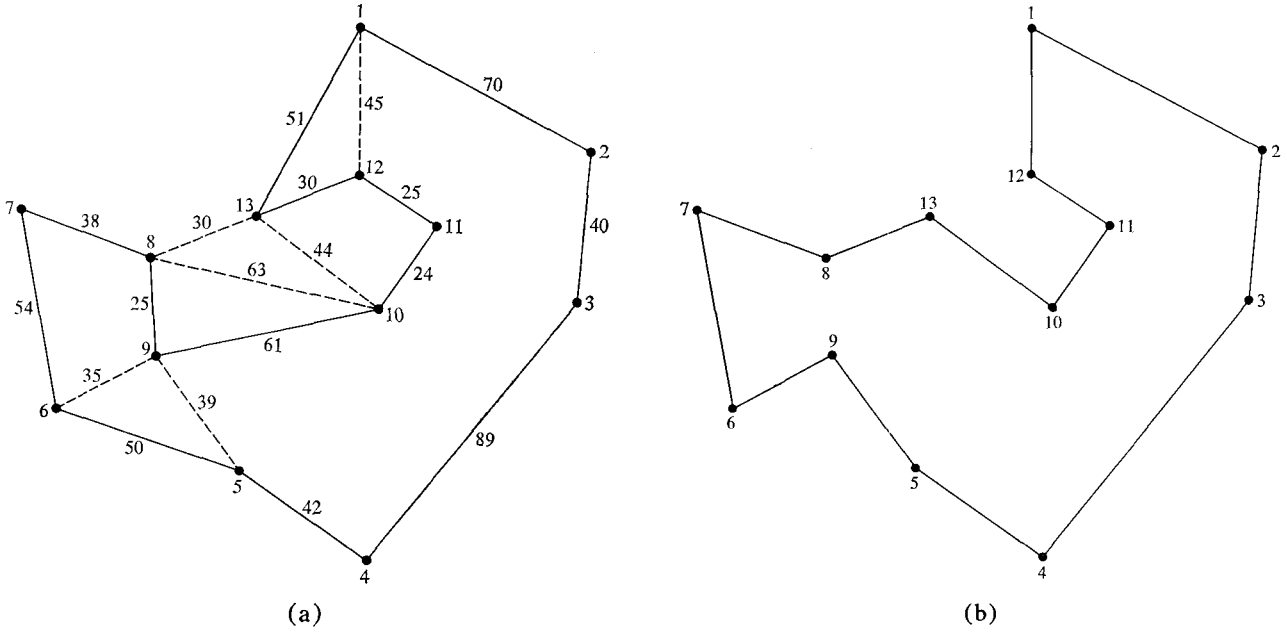

Figure 3 Thirteen-city problem: (a) nonoptimal tour with data for five-link-exchange correction and (b) corrected tour.

in the link that now gives the minimum tour length increment. In most cases one reinserts the node in the link that was created when the node was taken out of the tour, thus producing no change. For example, taking Node 1 out of the solution (Fig. 1c) by deleting links (1, 2) and (1, 5) and creating link (2, 5), one obtains the following increments for the respective links: 23 (5, 2), 41 (2, 3), 48 (3, 4), 61 (4, 7) and 36 (7, 5). The minimum increment is 23, which results in our breaking link (5, 2) to produce links (1, 5) and (1, 2). The tour in this case remains the same. When Node 5 is considered, however, an improved tour does result. Links (5, 1) and (5, 7) are first deleted and link (7, 1) is created. The increments for the respective links are 33 (1, 2), 28 (2, 3), 6 (3, 4), 29 (4, 7) and 20 (7, 1). Thus Node 5 is to be inserted between Nodes 3 and 4 by producing the links (5, 3) and (5, 4) and deleting the link (3, 4). The resulting tour, which is 14 units shorter, is shown in Fig. 1d.

In effect this procedure exchanges three links in the tour for three other links. Although a similarity exists in this respect to the "3-opt" algorithm,[4] much smaller sets of three links each are considered here at each stage (because two of the three links must be adjacent). In addition, far fewer changes occur in the tour at each stage. As a result, the computing time needed to generate the corrections remains relatively short. The timing estimate cannot be precise because an additional pass is made for each tour improvement. If no corrections are ever made, although a correction pass is made at each stage, timing can be estimated and is proportional to the number of increment calculations. At each stage there are k nodes that must be evaluated in $k - 1$ links. Summing over all stages one gets $(n^3 - n - 6)/3$. A shortcut can be taken by considering at each stage only the new links from the previous stage, but this procedure was not incorporated in the results described.

Another correction process developed by the author handles some complex situations that arise in large problems. The process begins in the same manner as the three-link exchange described previously. A node is isolated from the solution by deleting the two links adjoining the node and creating the link L that completes the tour. This produces a tour of reduced length and the length reduction will be called I_1. The insertion increments for the isolated node are then calculated for all links except L. The link that produced the best of these increments (the next-best for the node if we assume that L produced the best increment) is identified and increment I_2 is the temporary addition to the tour length. Each remaining node in the tour that is not connected to any of the links involving I_1 and I_2 is then isolated in turn in the same manner as the first node and we obtain a tour-length reduction called I_3. Finally, the tour-length increment I_4 is calculated when the last isolated node is inserted in link L. The resulting tour is shorter if $I_2 + I_4 - I_1 - I_3 < 0$. This type of correction involves substituting five links in the existing tour for five other links. Figure 3a illustrates the correction procedure.

Consider Node 9 in this figure. If Node 9 is taken out of the tour, the reduction I_1 in tour length is $d_{8,9} + d_{9,10} -$

$d_{8,10} = 23$. The link created (L) is (8, 10). The next-best place to insert Node 9 is in link (5, 6) with an increase I_2 in tour length of $d_{6,9} + d_{5,9} - d_{5,6} = 24$. As each node is considered in turn, Node 13 is eventually isolated and $I_3 = d_{13,12} + d_{13,1} - d_{12,1} = 36$. When Node 13 is inserted in link (8, 10), the increment I_4 is $d_{13,8} + d_{13,10} - d_{8,10} = 11$. The tour has been changed by creating links (6, 9), (5, 9), (13, 8), (13, 10) and (1, 12) and deleting links (8, 9), (9, 10), (6, 5), (13, 12) and (13, 1); it is now 24 units shorter and is shown in Fig. 3b.

The five-link exchange is obviously expensive in terms of computing time. For a tour of k nodes, timing is proportional to $k(2k - 6)$. If used at each stage, the expression must be summed from $k = 5$ to $k = n$. Corrections made with this technique are rare relative to those derived from the three-link exchange. The author used the five-link exchange only at the final stage.

The algorithm

These ideas indicate that a variety of algorithms can be developed from the basic one of Karg and Thompson and a variety of decision rules can be used in conjunction with the correction techniques. The most successful combination of decision rules and tour-improvement techniques was developed after considerable experimentation, most of which is described in Ref. 6, and resulted in the following algorithm:

1. Choose two nodes i_1 and i_2 and form a tour with links (i_1, i_2) and (i_2, i_1). Set the number of links j equal to 2.

2. Define I_k^* as the minimum tour-length increment that results from the insertion of node k in each existing link (p, q). Define nodes p^* and q^* as the nodes connected by the link (p^*, q^*) that produced I_k^*. In addition, define J_k^* as the increment for node k most nearly equal to I_k^*. Thus for node k, I_k^* is the best increment, J_k^* the next-best increment and (p^*, q^*) the link to break if inserting node k. Calculate I_k^* and J_k^* for each node that is not currently included in the tour.

3. If $j \leq 4$, choose the node m for which I_k^* is a maximum. If $j > 4$, choose the node m for which $|J_k^* - I_k^*|$ is a maximum. In either case, p^* and q^* are identified.

4. Insert the node m into the tour between p^* and q^* by adding links (p^*, m) and (m, q^*) and deleting link (p^*, q^*). Increase the number of links in the tour by one.

5. If $j \leq 10$, or if j is even, or if $j = n$, use the three-link-exchange correction procedure on the present tour. If a tour improvement results, repeat this step after making the correction. If no improvement results, continue with Step 6.

6. If $j < n$, return to Step 2. If $j = n$, use the five-link-exchange correction procedure. If a tour improvement results, apply the correction and repeat Step 5. If no improvement results, a solution exists.

Note that for a symmetric n-node problem only $n(n-1)/2$ starting states are possible and thus there are only $n(n-1)/2$ possible solutions (if we make a deterministic decision in cases in which $I_k^* = J_k^*$). The basic Karg-Thompson algorithm produces solutions for each of the $n!$ permutations of the nodes.

Application

Computer results have been obtained and compiled for a variety of problems. Each problem is well known and is generally labeled by the number of cities or nodes it includes. These are (1) the 25-city problem of Held and Karp,[2] (2) the 33-city problem of Karg and Thompson,[5] (3) the 42-city problem of Dantzig, Fulkerson and Johnson,[7] (4) the 48-city problem of Held and Karp[2] and (5) the 57-city problem of Karg and Thompson.[5] The distance matrices for the 33-, 42- and 57-city problems can be found in Ref. 5 and for the 25- and 48-city problems in Ref. 2. Each of these problems involves finding a tour through a set of cities in the 48 contiguous states. Although there are similarities in the general shape of the optimal tours, the author conjectures that most of the complexities of the metric† traveling-salesman problem are demonstrated by these problems. Despite the metric nature of the problem, the distances used are functions of highway route lengths, not the point-to-point lengths of Euclidian distances. The known or conjectured optimal tours of these problems are shown in Figs. 4 through 8, respectively.

- *Numerical results*

Table 2 contains a summary of the results of the five problems studied and Table 3 shows the distribution of solutions with respect to their deviation from the optimal tour. The data were obtained for all problems by using node numbers corresponding to the row (column) subscripts in the referenced distance matrices. For each problem, solutions were obtained for $n(n - 1)/2$ unique starting states (i_1, i_2) corresponding to (1, 2), (1, 3), $\cdots$, (1, n), (2, 3), (2, 4), $\cdots$, (2, n), $\cdots$, ($n - 2$, $n - 1$), ($n - 2$, n) and ($n - 1$, n).

The distribution of the solutions (Table 3) is evidence of the nature of the locally optimal solutions. A relatively small number of unique solutions can be obtained. The problem is discrete in nature and, furthermore, consecutive integral solutions may not exist. Thus gaps appear in the frequency distribution and flat intervals in the cumulative distribution.

† A metric problem is one for which the nodes exist in a metric space. A space is metric if, for each pair of points (x, y) in the space, the distance $d(x, y)$ between the points is such that (1) $d(x, y) = 0$ if and only if $x = y$, (2) $d(x,y) = d(y, x)$ and (3) $d(x, y) + d(y, z) \geq d(x, z)$.

From the data in Table 3 the computing time for an n-node problem can be approximated by $(9n^3 + 140n^2 + 400n)$ μsec per solution; the time will be slightly greater for smaller problems because the three-link-exchange procedure is executed disproportionately often.

• *Comparison with other algorithms*

Some comparisons can be made of the results presented here with the first-phase-algorithm results of Lin[4] and of Karg and Thompson.[5] Differences in the computers and programming languages used make exact comparison difficult, however.

Karg and Thompson report finding one optimal solution out of 100 tours for the 33-city problem and 10 out of 225 for the 42-city problem. In 100 tries, the best solution for the 57-city problem was 331 distance (cost) units (2.8%) from the conjectured optimal value. The author, in earlier work, found no optimal solution in 600 tries on the 48-city problem, but a best solution of 11,470 units. The extensions described here obviously shift the distribution of solutions toward the optimum.

Lin, without the second-phase reduction, reports a probability of about 0.05 of finding an optimal tour for the 48-city problem and 0.02 for the 57-city problem.

Figure 4 Optimal tour for the 25-city problem.

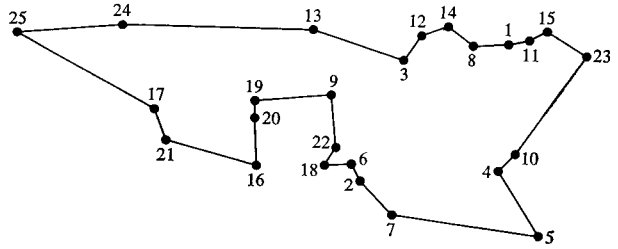

Figure 5 Probable optimal tour for the 33-city problem.

1. Chicago, Ill.
2. Indianapolis, Ind.
3. Marion, Ohio
4. Erie, Pa.
5. Carlisle, Pa.
6. Wana, W. Va.
7. Wilkesboro, N. C.
8. Chattanooga, Tenn.
9. Barnwell, S. C.
10. Bainbridge, Ga.
11. Baton Rouge, La.
12. Little Rock, Ark.
13. Kansas City, Mo.
14. La Crosse, Wis.
15. Blunt, S. Dak.
16. Lincoln, Nebr.
17. Wichita, Kans.
18. Amarillo, Texas
19. Truth or Consequences, N. Mex.
20. Manuelito, N. Mex.
21. Colorado Springs, Colo.
22. Butte, Mont.
23. Lewiston, Idaho
24. Boise, Idaho
25. Twin Falls, Idaho
26. Salt Lake City, Utah
27. Mexican Hat, Utah
28. Marble Canyon, Ariz.
29. Reno, Nev.
30. Lone Pine, Calif.
31. Gustine, Calif.
32. Redding, Calif.
33. Portland, Oreg.

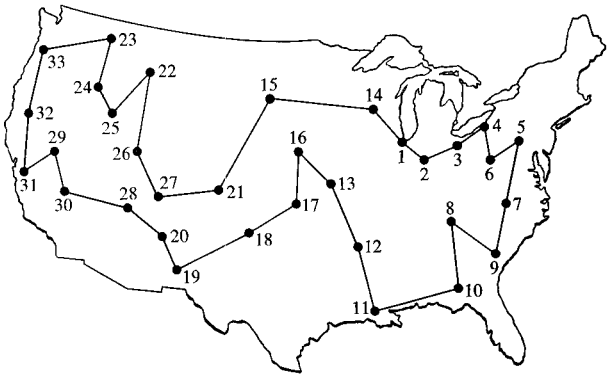

Figure 6 Optimal tour for the 42-city problem.

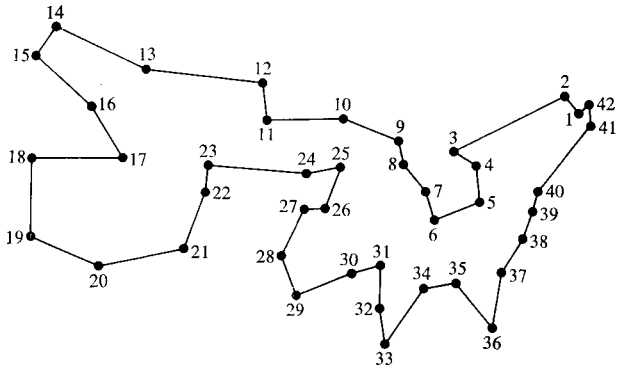

Figure 7 Optimal tour for the 48-city problem.

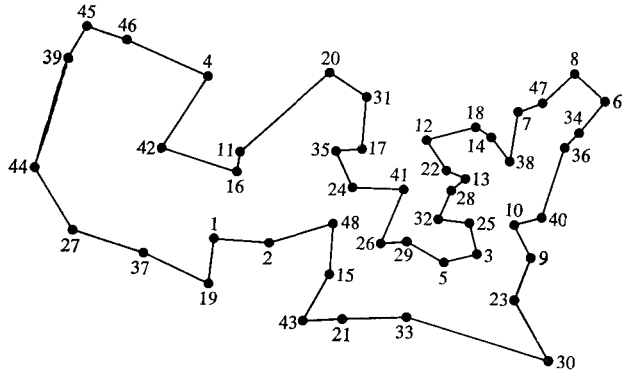

Figure 8 Probable optimal tour for the 57-city problem.

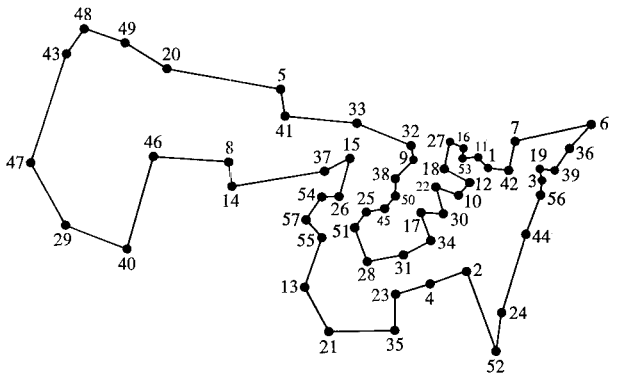

Table 2 Summary of computational results.

Problem (number of cities)	Solution (arbitrary distance units)				Number of solutions	Average computing time (sec)	Number of unique solutions	Number of optimal solutions; percent
	Best	Worst	Average	Known optimal				
25	1,711	1,763	1,731	1,711	300	0.28	4	53; 17.7
33	10,861	11,122	10,958	10,861	528	0.55	5	31; 5.9
42	699	728	702	699	861	1.06	12	553; 64.3
48	11,461	12,143	11,582	11,461	1,128	1.32	28	46; 4.1
57	12,955	13,908	13,150	12,955	1,596	2.14	104	9; 0.056

Table 3 Cumulative distribution of solutions.

Problem (number of cities)	Percent of solutions within $\pm p$ percent of the optimal solution							
	p = 1	2	3	4	5	6	7	8
25	72.3	72.8	78.7	78.7	100			
33	69.3	98.0	100					
42	68.3	94.7	98.8	99.7	100			
48	69.9	86.3	97.8	99.8	99.9	100		
57	52.4	71.6	83.6	90.4	94.5	99.8	99.9	100

The average results of ten tours each reported for the 42-, 48- and 57-city problems are, respectively, 713, 11,667 and 13,176 units. Computing times on an IBM 7094 Model 2 were 2.08, 2.80 and 5.05 sec per solution, respectively. The average computing time was approximately $30n^3$ μsec.

The author's algorithm yields a slightly better average solution, but the difference is not necessarily significant. Lin's algorithm yields slightly higher probabilities of optimal solutions, which may be significant for large problems. Running times are roughly comparable when differences between the IBM System 360/65 and 7094/2 are considered. The minimum storage requirements for the two algorithms are also comparable.

Remarks

Certain nodes in large problems, if included in the starting state, seem to produce many more optimal solutions than other nodes. For example, Node 10 of the 57-city problem was in the starting state of eight of nine optimal solutions. Node 13 of the 48-city problem was involved in 31 of the 46 optimal solutions. The reason for this consequence is not known.

The author learned in prior work that the five-link-correction procedure produces only small average improvements, even though the improvements may produce an optimal solution. Approximately 0.3 and 0.7 sec per solution for the 48- and 57-city problems, respectively, were used for this correction. Overall computing time has been reduced in a later version of the program (by implementing the three-link-exchange shortcut) to approximately $(4n^3 + 150\,n^2 + 300n)$ μsec for an n-node problem, with observed average times per solution of 0.163, 0.768 and 1.25 sec for the 25-, 48- and 57-city problems, respectively.

Additionally, the author considers that the algorithm described and the Karg-Thompson algorithm are best suited to problems with geometric properties. Both, however, can be applied to nonmetric problems and modified to solve nonsymmetric problems.

Conclusions

The algorithm described appears to be competitive with existing heuristic techniques. It would be naive to assume that the best combination of decision rules and correction techniques has been found; further experimentation may prove fruitful.

The method does yield a high percentage of near-optimal solutions, with over 50% of the solutions for each problem within 1% of the optimum. Thus one needs to obtain only a small number of solutions to be assured with a reasonable degree of confidence that the best solution is close to optimum.

The size of the problem does not determine the degree of difficulty of solving it. This conclusion is borne out by the large number of optimal solutions of the 42-city problem. Large problems with nodes concentrated in one or more areas, however, prove difficult. For dense problems, increased size results in more unique solutions and, in turn, a longer tail on the frequency distribution of solutions away from optimality.

Acknowledgment

Some of the material in this paper, but not the precise algorithm described, was included in the author's M.S. thesis[6] at Syracuse University. Special thanks are due R. Sargent of Syracuse for his assistance in preparing the thesis and for his suggestions on formulating this paper.

References

1. M. Bellmore and G. L. Nemhauser, "The Traveling-Salesman Problem: A Survey," *Operations Res.* **16,** 538 (1968).
2. M. Held and R. Karp, "A Dynamic Programming Approach to Sequencing Problems," *SIAM J. Appl. Math.* (formerly *J. Soc. Indust. Appl. Math.*) **10,** 196 (1962).
3. J. Little, K. Murty, D. Sweeney and C. Karel, "An Algorithm for the Traveling-Salesman Problem," *Operations Res.* **11,** 972 (1963).
4. S. Lin, "Computer Solution of the Traveling-Salesman Problem," *Bell System Tech. J.* **44,** 2245 (1965).
5. R. L. Karg and G. L. Thompson, "A Heuristic Approach to Solving the Traveling-Salesman Problem," *Management Sci.* **10,** 225 (1965).
6. T. C. Raymond, "New Heuristic Algorithms for the Traveling-Salesman Problem," M.S. thesis, Syracuse University 1968.
7. G. B. Dantzig, D. R. Fulkerson and S. M. Johnson, "Solution of a Large-scale Traveling-Salesman Problem," *Operations Res.* **2,** 393 (1954).

Received January 6, 1969