Title: The Time-Constrained Traveling Salesman Problem: A Tour of Existing Solution Techniques

Course:
Year: 1991
Author(s): C. Shlaes

Report No: P91024

Abstract: This paper explores various proposed methods for solving a time-constrained traveling salesman problem (TCTSP). A small (seven city) problem was formulated for use as an illustration for some of the solution techniques discussed below.

THE TIME-CONSTRAINED TRAVELING SALESMAN PROBLEM: A TOUR OF EXISTING SOLUTION TECHNIQUES

CAROLE SHLAES

EMGT 505

DECEMBER 6, 1991

TABLE OF CONTENTS

INTRODUCTION

The classical traveling salesman problem (TSP) has been widely studied for many years. It determines the shortest route for a trip, starting in one city, visiting some number of other cities and returning to the starting city, where all in the intercity distances are known. However, the presence of additional constraints, such as time windows in which one or more of the cities must be visited, can complicate the solution process for the general TSP. This paper explores various proposed methods for solving a time-constrained traveling salesman problem (TCTSP). A small (seven-city) problem was formulated for use as an illustration for some of the solution techniques discussed below.

BACKGROUND

Researchers have studied the classical TSP for many years, based both on its applications to diverse situations and its relationship to many other combinatorial optimization problems.[1] The general form of the TSP occurs in various situations other than that of an actual salesman. For example, the same problem arises in deciding the task sequences when the various task-to-task changeover times differ. Other situations in which the TSP can be directly applied include vehicle routing[2], job sequencing and computer wiring. The problem formulations for these types of problems contain (amongst others): (1) a set of constraints to impose connectivity of routes, and (2) a set of constraints to impose traversibility of the solution.[3]

However, in many real world applications, additional constraints, such as time windows, transportation costs, due dates, vehicle capacities, etc., must be added to the general TSP to portray the situation accurately. The addition of these constraints tends to destroy whatever structure the original unconstrained problem had.[4] Thus, solutions to the TSP with additional constraints are often elusive.

The present paper looks at some of the solution techniques for the general TSP and then at the ways in which some of these techniques have been extended to help solve the specific situation of a time-constrained TSP (TCTSP).

PROBLEM DEFINITION

The specific problem under consideration here is the Time-Constrained TSP (TCTSP), which is a special case of the general TSP. The TCTSP assumes that there are n cities of interest. A salesperson based at city 1 must visit each of the remaining n-1 cities exactly once before returning to city 1. Additionally, however, the TCTSP requires that the visit to each city be made within specific time windows. Several sets of distinct time windows (or no time windows) can be specified for each city. The TCTSP tries to find the sequence of cities, or tour, that visits each city within an open time window and minimizes the total length of the tour.[5]

OREGON EXAMPLE

An example problem for investigating some of the methods of solving a TSP was formulated based on seven travel destinations in Oregon. The destinations include: (1) Astoria; (2) Bend; (3)

Crater Lake; (4) Portland (the starting and ending city); (5) The Dalles; (6) Wallowa Lake; and (7) Yachats. Figure 1 shows the general location of these cities. Table 1 shows the distances in miles between the cities that is used for all calculations. The problem is first solved as a general TSP. Then some of the TCTSP-solving techniques were reviewed by adding time windows for the visits to some of the cities.

FIGURE 1: OREGON TRAVELING SALESMAN PROBLEM

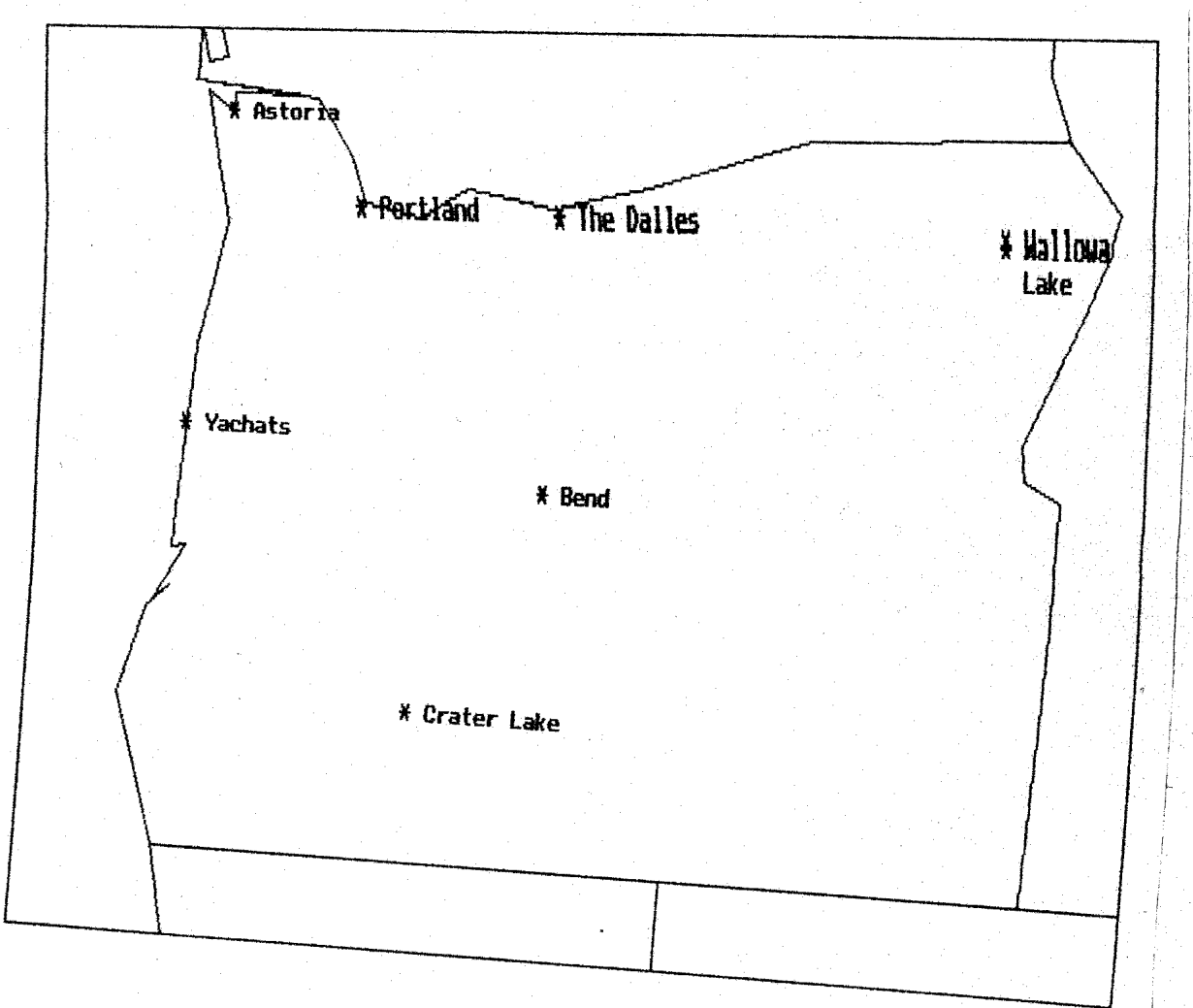


TABLE 1: DISTANCES FOR OREGON TSP

	Astoria	Bend	Crater Lake	Portland	The Dalles	Wallowa Lake	Yachats
Astoria	--	255	335	95	175	421	130
Bend	255	--	107	160	131	348	207
Crater Lake	335	107	--	250	239	455	264
Portland	95	160	250	--	83	329	138
The Dalles	175	131	239	83	--	246	220
Wallowa Lake	421	348	455	329	246	--	487
Yachats	130	207	264	138	220	487	--

CHARACTERISTICS OF THE GENERAL TSP

Size of Problem

One of the major problems that make a general TSP difficult to solve is the large number of possible sequences that must be examined to arrive at an optimal solution. For a symmetric ($c_{ij} \neq c_{ji}$) n-city tour, there are $1/2[(n-1)!]$ possible tours.[6] Thus, for the seven-city symmetric Oregon tour, there are 360 possible sequences to evaluate. For an asymmetric 30-city problem, there would be over 4.4×10^{30} possible sequences.

There is twice the total number of possible tours in an asymmetric ($c_{ij} = c_{ji}$) n-city TSP, or $(n-1)!$[7] The case of asymmetric costs, which is typical in the scheduling of chemical processes, has proven considerably more problematic for heuristics and has received less attention.[8]

Note that for each path, there corresponds a reverse path x^R. For a symmetric TSP, $f(x) = f(x^R)$ for all x in the subspace. Therefore, the minimum path will not be unique (can traverse the tour in the reverse order).[9]

NP-Complete

As shown above, the number of possible sequences for a TSP grows very rapidly as the number of cities increases. In addition, as the problem size increases, the number of calculations required to solve it quickly grows out of convenient computation range.

Like almost all problems of vehicle routing and scheduling, the TSP is classified as NP-complete.[10] Computational complexity theory has provided strong evidence that any optimizing algorithm for its solution is likely to perform very poorly on some occasions; more formally, its worst case running time is likely to grow exponentially with problem size.[11]

According to the theory of nondeterministic polynomial time completeness (NP-completeness), if an efficient algorithm can be found for the TSP, then efficient algorithms could be constructed for all problems in class NP-complete. An algorithm is deemed efficient if its execution time is bounded by a polynomial function written in terms of some reasonable measure of problem size.[12] So far, the general TSP has resisted numerous attempts to solve it efficiently.

SOLUTION TECHNIQUES FOR THE GENERAL TSP

Exact Solution IP Formulation

The TSP has been formulated as integer linear programming models to get an exact solution to the problem. A useful formulation of the TSP follows[13]:

$$\text{Min} \quad \Sigma\Sigma c_{ij}x_{ij} \tag{1}$$

$$\text{s.t.} \quad \Sigma_i \, x_{ij} = 1 \qquad j = 1 \text{ to } n \tag{2}$$

$$\Sigma_j \, x_{ij} = 1 \qquad i = 1 \text{ to } n \tag{3}$$

$$u_i - u_j + nxij \leq n - 1 \qquad \text{for } 2 \leq i \neq j \leq n \tag{4}$$

$$x_{ij} = 0 \text{ or } 1 \qquad \text{for all } i, j \tag{5}$$

u_i and u_j are arbitrary real numbers

c_{ii} is defined to be a large positive number to make sure that all the x_{ii} will be zero. Constraint (4) is formulated to prevent the occurrence of subtours.

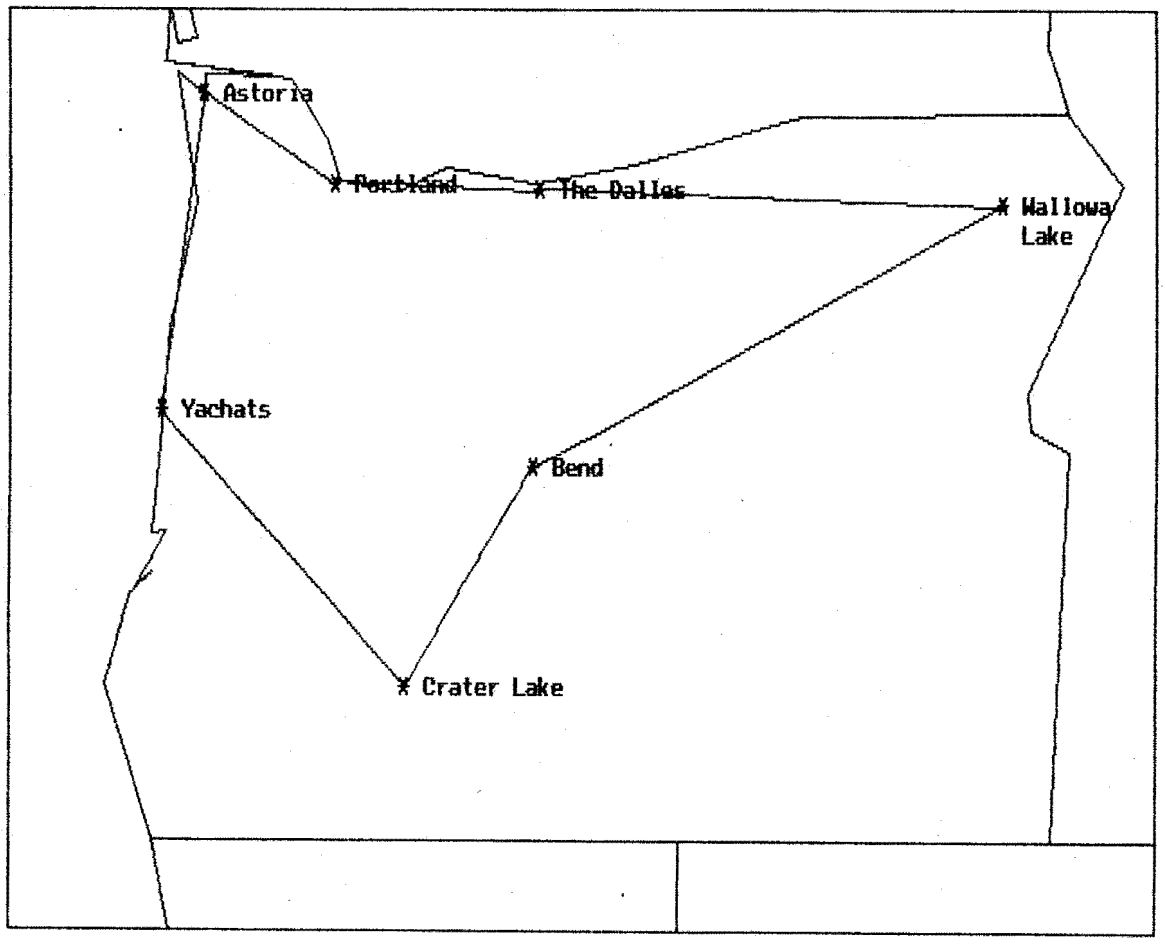
The dimensions of the formulation do not appear to be unreasonably large, at least for small problems. There are n constraints for the sum of the columns to equal 1, n constraints for the sum of the rows to equal one and n(n-1) for the subtour prohibition. Therefore, the complete formulation contains $n^2 + n$ constraints and n^2 variables.[14]

For the Oregon example problem, there are 56 constraints and 49 variables. The program listing for use with LINDO is included in Appendix A, along with the results of the model. The resulting tour, which is optimal at 1273 miles, is shown in Figure 2.

When solving the formulation with LINDO, over 1150 iterations and several minutes of computer time were required to

solve the problem. The number of computations required to solve the problem would increase rapidly for a program with more cities. In addition, the LP code would grow accordingly. For example, a 30-city problem would require 930 constraints and 900 variables. To set up and solve a problem of this size, some sort of matrix generator would be necessary to the input program for use by a LP code like LINDO. In addition, the maximum size of the code's constraint and variable set must be taken into account. For example, the student version of LINDO can only solve an eight-city problem.

FIGURE 2: OPTIMAL SOLUTION TO OREGON TSP



Explicit Enumeration

Since there are (n-1)! possible ways to order the cities in an asymmetric TSP, it can require prohibitive amounts of computer time to enumerate all possible solutions for problems involving even a fairly small number of cities. (There are over 3.6 million possibilities for a problem with only 10 cities.)

One method for solving large TSP problems and similar combinatorial problems is to sample from the large number of possible solutions, using a computer to evaluate some large number of the possible solutions at random. This is done by selecting successive random sequences. The average of all the costs of the sample sequences, the standard deviation of the costs and the lowest-cost sequence are computed. Then the probability that another sample will yield a lower cost solution, and an estimate of how much lower it is likely to be can be evaluated. When the expected gain is less than the cost of additional sampling, the best solution thus far selected may be adopted.[15]

Branch and Bound

Due to the large number of possible combinations of cities, branch-and-bound methods are often used to implicitly enumerate all possible solutions to a combinatorial optimization problem.

If constraint (4) is removed from the above formulation, the remaining relaxation is a standard assignment problem, which can then be solved using branch and bound techniques. If the optimal solution to the assignment problem is feasible for the TSP (i.e., if the assignment problem contains no subtours), the optimal solution to the assignment problem is also optimal.[16] If the

assignment problem contains subtours, the minimum objective value in the assignment problem is a lower bound for the minimum cost of a tour.[17]

If there are any subtours in an assignment solution, the solution is not a tour assignment. The problem can be solved using branch and bound, adding two subproblems, with the constraints $x_{ij} = 0$ or $x_{ji} = 1$ for a variable that generated a subtour. Thus, one subproblem contains the specific element of the matrix constrained to be part of the solution and the other subproblem prohibits the same element.[18]

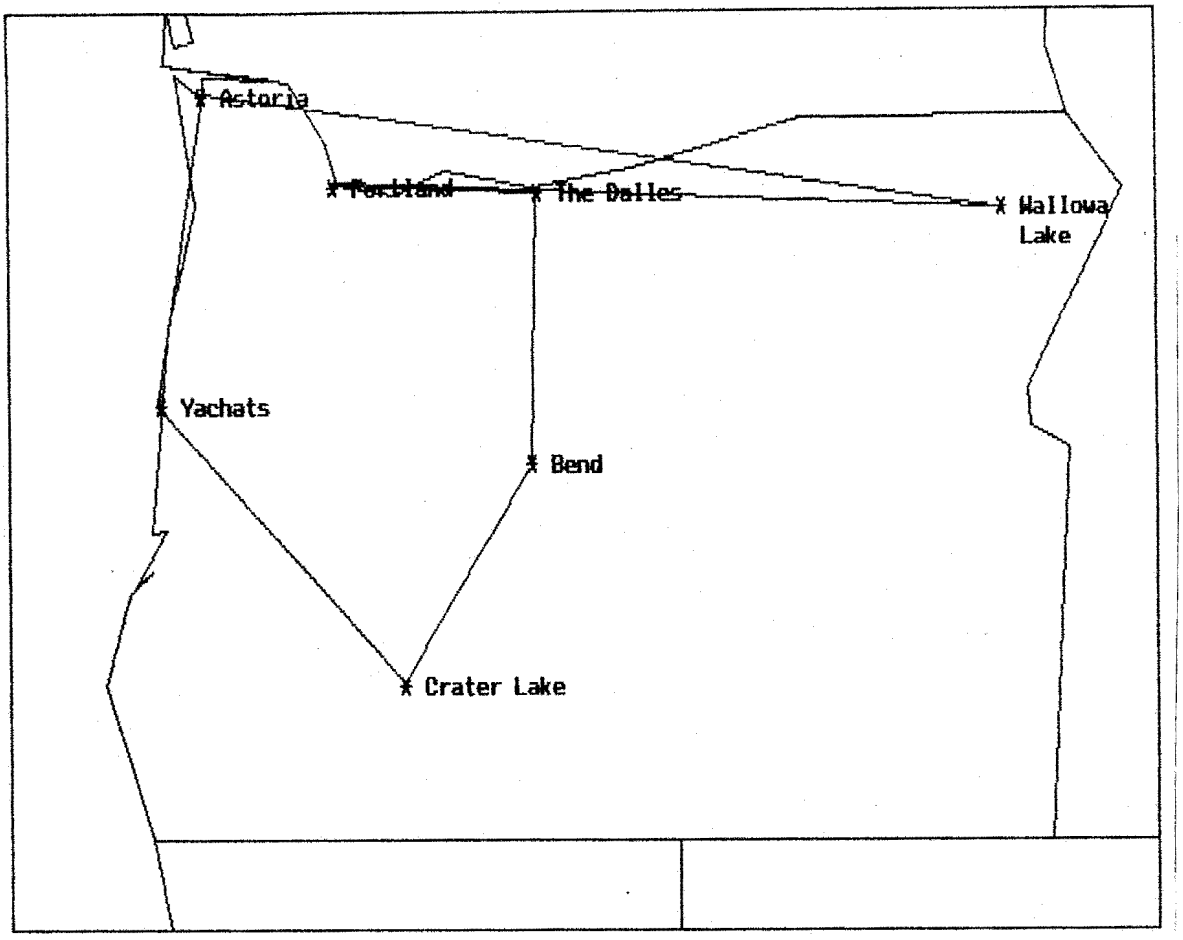
Heuristics

Due to the large number of sequences that are possible in even a relatively small TSP, many practitioners questioned the need for an exact solution and have instead utilized algorithms that give an approximation of the optimal solution. This has resulted in a variety of heuristics for the solution of the general TSP. There are many heuristics for the general TSP, and the primary ones that have been extended to help solve the TCTSP will be discussed here.

Nearest Neighbor - One of the simplest heuristics for solving the general TSP is the nearest neighbor heuristic. This heuristic begins at a city and travels to the nearest city not yet visited. The tour is completed by returning to the initial city.

Figure 3 shows the tour generated by using the nearest neighbor heuristic on the Oregon problem. The total mileage for

FIGURE 3: NEAREST NEIGHBOR SOLUTION TO OREGON TSP



this route was 1465 miles, which is significantly larger than the 1273-mile tour produced by the exact solution. While not optimal, this number would provide an upper bound on other solutions, such as the branch and bound. In many practical situations, the result of the nearest neighbor algorithm is used as a starting point for some of the other heuristics discussed below.

Although this heuristic has some drawbacks, its importance lies in its ability to generate relatively good solutions rapidly

in problems where the cost of implementing an optimum-seeking method is prohibitive.[19]

The drawbacks to the nearest neighbor heuristic include: (1) it is sensitive to which city is designated as the origin[20]; (2) unless the graph is complete, the procedure may fail to find a tour even if one exists; and (3) even on complete graphs it can perform very badly by being forced to choose arcs of very large weight in the last steps.[21] This last drawback is illustrated in the Oregon tour, where the trip to Wallowa Lake is done at the very end of the tour and adds many miles to the tour.

Nearest Insertion - The nearest insertion heuristic is used to build up a closed tour of the cities.[22] The method begins with a randomly selected pair of cities, constituting a tour of length 2. Then a third city is inserted to minimize the resulting 3-city tour, then a fourth city is inserted, and so on until a complete tour has been constructed. This heuristic is also sensitive to which pair of cities is designated as a seed and the order in which jobs are considered for insertion.[23] Heuristic rules for these facets of the algorithm have not been thoroughly explored, but one way to proceed is to repeat the algorithm several times, each time beginning with a seed pair that is randomly selected.[24] Although the above heuristic works well on small problems, the number of possible insertions grows quickly and the process become unwieldy for large problems.[25]

k-Interchange - Local search heuristics are also useful for the general TSP. Given a tour, the k-interchange heuristic

replaces k arcs in the tour by k arcs that are not in the tour, if such a change yields a tour of lower cost. This method is typically used by starting from a feasible tour produced by another method (e.g., nearest neighbor, nearest insertion) to see if the interchanges can be made to improve the tour.[26]

Dynamic Programming

The TSP can be solved by using dynamic programming. The traveler must visit each city exactly once. When there is only one city to visit, the problem is trivial; simply go from the current location to the initial city. Dynamic programming works backwards in this manner until all the cities have been visited.

Let any stage be indexed by the number of cities that have already been visited. At any stage, to determine which city should next be visited, two things must be known: the current location and the cities that have already been visited. The state at any stage consists to the last city visited and the set of cities that have already been visited. We define f(i,S) to be the minimum distance that must be travelled to complete a tour if the t-1 cities in the set S have been visited and city i were the last city visited.[27]

For large TSP's, the state space becomes very large and the branch-and-bound approach (above and others) is much more efficient that the dynamic programming approach. For example, for a 30-city problem, if the solution is at stage 16 (15 cities have already been visited), there are over 1 billion possible states. This is a limitation on the practical application of dynamic programming for moderate-size TSP's. In many problems,

Given n cities, the model defines a single, nonnegative variable, t_i, to be the time that the traveller visits city i. Since the traveller must return to city 1 at the end of the tour, the formulation includes an additional variable t_{n+1}, to determine the time at which the tour is completed. The model assumes that a complete, symmetric, nonnegative distance matrix, $||d_{ij}||$, is known and that time is a scalar transformation of distance so that time and distance may be used interchangeably. Additionally, it assumes that the triangle inequality holds for the distance measure.

$$\min \quad t_{n+1} - t_1 \tag{6}$$

s.t.

$$t_i - t_1 \geq d_{i1} \qquad i = 2, 3, \ldots, n \tag{7}$$

$$|t_i - t_j| \geq d_{ij} \qquad i = 3, 4, \ldots, n;\ 2 \leq j < i \tag{8}$$

$$t_{n+1} - t_i \geq d_{i1} \qquad i = 2, 3, \ldots, n \tag{9}$$

$$t_i \geq 0 \qquad i = 1, 2, \ldots, n+1 \tag{10}$$

$$l_i < t_i < u_i \qquad i = 2, 3, \ldots, n \tag{11}$$

where

t_i = the time that the traveller visits city i.

$|X|$ = the absolute value of X

d_{ij} = the shortest time required to travel from city i to city j

l_i = the lower bound on the time window for the traveller to visit city i. By assumption, all $l_i \geq 0$.

u_i = the upper bound on the time window for the traveller to visit city i, $u_i \geq l_i$ for all i.

This formulation assumes a single upper and lower bound for each time window, but it can be extended to include a set of distinct time windows for each city, one of which must be satisfied.

The proposed model appears to be quite concise. It contains (i) n-1 variables; (ii) 2n-2 linear constraints; (iii) ((n-1)*(n-2))/2 absolute value constraints; and (iv) n-1 time window constraints. Thus, for the seven-city Oregon model, there would be 6 variables, 10 linear constraints, 15 absolute value constraints and 6 time window constraints.

The solution of the model, however, is complicated by the presence of the absolute value constraints. These constraints introduce both a nondifferentiability and a nonconvexity. Solutions may be obtained, however, using a branch and bound procedure.[34]

Discrete cases of the proposed model are determined by the two possible cases of the $|t_i - t_j| \geq d_{ij}$ absolute value constraints. These constraints act as the disjunctive constraints for the branch and bound solution. Given a choice of one of the cases for each disjunctive constraint in the current enumeration, the resulting TCTSP model is a linear program. The solution to each subproblem, therefore, may be obtained by solving the associated linear program. The branch and bound solution procedure is initiated by relaxing the absolute value constraints (3) and the time window constraints (6) of the proposed TCTSP model.[35]

The presence of distinct time windows in a vehicle scheduling problem allows some simple reductions to be performed that may reduce the complexity of the solution process. Two

types of reduction procedures are used to preprocess the vehicle scheduling test set. First, whenever $l_i + d_{i,j} > u_i$ for any node pair (i,j), node j precedes node i. Second, each node without a time window is examined for possible fit between each pair of nodes for which the time windows were enforced. It the fit is found to be infeasible, arcs may be eliminated from the reversible arc list.[36]

According to the testing results, the proposed algorithm was shown to be effective on several small- to moderate-size vehicle scheduling where a large percentage of the demand points possessed time windows.[37] It appears that the smaller the percentage of points with time windows, the poorer the results. This tends to indicate that if the system is very constrained by the time windows, it is more easily solved. This is most likely due to the limited number of scheduling decisions that are left to determine.

For the Oregon model, the time window for Yachats was added as described above. The LINDO code was prepared and is included as Appendix B. However, the implementation of the branch and bound portion of the problem has not yet been successful. More work will be required to investigate the performance of this IP formulation on the Oregon model.

Explicit Enumeration

Since the TCTSP has the same number of possible solution sequences as the general TSP, it may be possible to enumerate and test all possible sets of tours. This is true particularly if certain classes of solution can be eliminated at the beginning as not feasible.

The presence of distinct time windows in a vehicle scheduling problem allows some simple reductions to be performed that may reduce the complexity of the solution process.[38] For example, the time constraints may set the order of visiting some cities. Thus, any sequences which violate this ordering can be removed from the possible solution set before the problem is solved. However, if only a few cities have time windows, the possible solution set may still be prohibitively large to explicitly enumerate.

Set partitioning approaches are most useful when the number of candidate tours can be limited due to the restrictive constraints imposed on the tours (tight time windows, limited route duration or capacity limitations).[39]

Reduction of Possible Schedules Through Dominance

Enumeration techniques used to solve the TCTSP may be improved by restricting the search to a subset of all possible sequences, if this can be done without eliminating feasible sequences.

A method for restricting the set of possible schedules that need to be examined was proposed by Erschler, Fontan, Merce and Roubellat.[40] The method is based on the concept that some sequences may dominate others when scheduling n independent jobs on a single machine with ready times and due dates. The concept permits a restricted set of schedules, the "most feasible" ones, to be established using only the ordering of ready times and due dates.

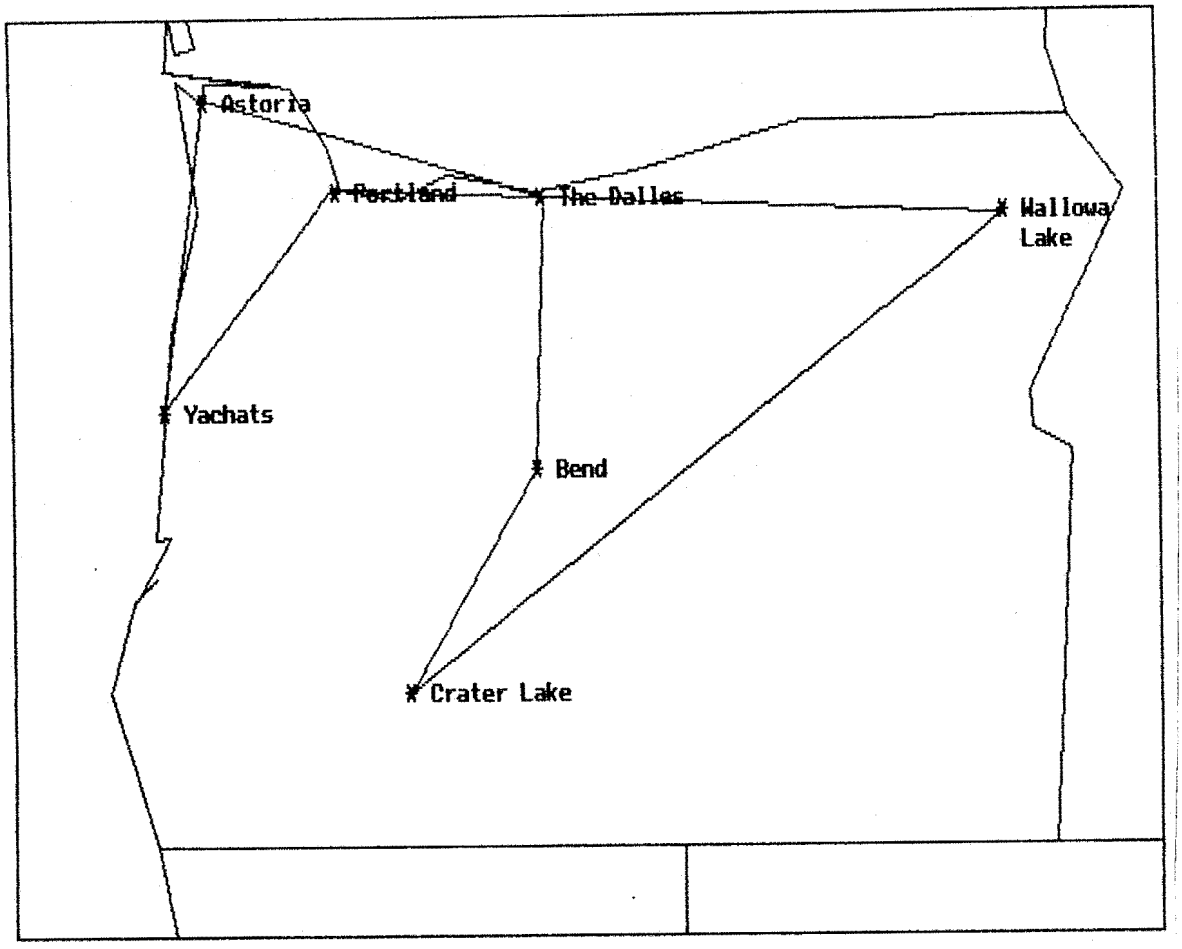
A Time-Oriented Nearest Neighbor Heuristic - This heuristic is an extension of the heuristic for the general TSP. At every iteration, the heuristic searches for the city "closest" to the last city added to the tour. The "closeness" measure tries to account for both geographical and temporal closeness of cities. It is a measure of the distance between two cities and the time difference between their respective delivery times, given the current point in the schedule.[44]

The search is performed among all the cities that can feasibly (with respect to time windows and travel time) be added to the end of the emerging tour. With this heuristic, a new tour is started anytime the search fails, unless there are no more cities to visit. The addition of new tours limits the suitability of this heuristic for the single-vehicle TCTSP. Further, the failure of a search with this method does not necessarily mean that a feasible solution does not exist.

This heuristic was used to solve the Oregon TSP with the added time window constraint for Yachats. The resulting tour of 1465 miles is shown in FIG. 4. Once again, the use of the nearest neighbor criteria for adding cities to the end of the tour resulted in the longest arcs being left to the very end. Therefore, the visit to Wallowa Lake once again made the tour route longer than an optimal solution.

Insertion Heuristics - This heuristic appears to hold promise for the one-vehicle TCTSP. After accounting for the cities with fixed sequence position due to the time window constraints, the unvisited cities can be assigned positions in the sequence.

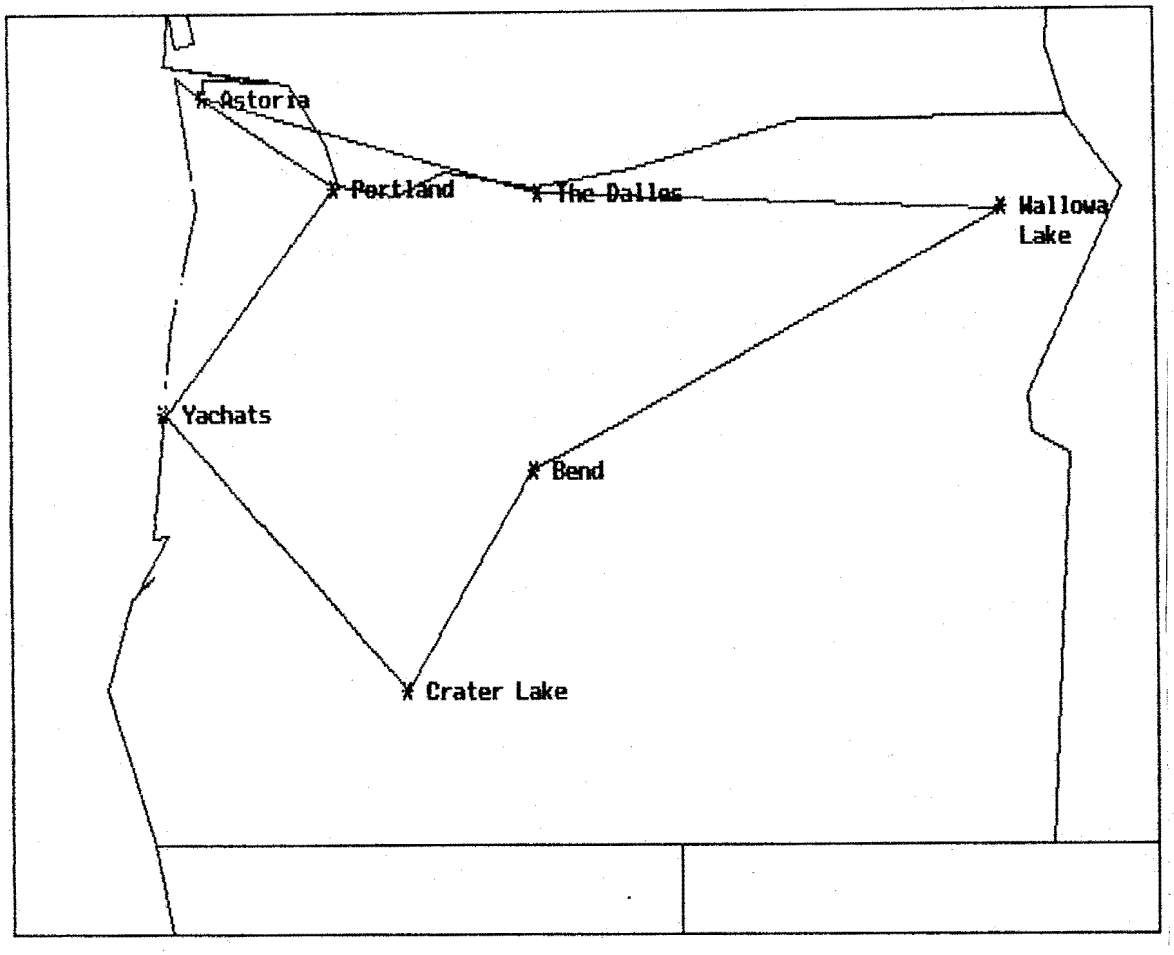
FIGURE 4: TIME-CONSTRAINED NEAREST NEIGHBOR HEURISTIC



For each unvisited city, its best feasible insertion place (with respect to time windows and distance) in the emerging tour is calculated. Then the best city to insert is selected. For the situation with time constraints only, the selection of the city in the tour will be based on the insertion costs to minimize a measure of total tour distance and time.[45]

The Oregon TSP was solved using this heuristic and the resulting tour was 1373 miles. It is shown in FIG. 5. This illustrates the importance of not being too greedy in the initial stages of tour development.

FIGURE 5: INSERTION HEURISTIC FOR OREGON TCTSP



Interchange Heuristics - These improvement heuristics attempt, at every iteration, to interchange k arcs in the current TCTSP solution with k arcs presently not used. Starting from an initial solution, such as one given by one of the heuristics described above, time-feasible k-interchanges are considered within the tour. A feasible k-interchange is performed if and only if it provides an improved solution, i.e., it produced at most the same number of vehicles, and its leads to a reduction in the routing and scheduling cost (e.g., distance).[46]

Goal Programming

With the inclusion of additional constraints in a TSP, it may be impossible to find a feasible solution to the problem due to conflicting time and distance constraints. If a feasible solution to a TCTSP cannot be found, the problem may be a candidate for solution by goal programming.

The basic approach of goal programming is to establish a specific numeric goal for each of the objectives, formulate an objective function for each objective and then seek a solution that minimizes the (weighted) sum of deviations of these objective functions from their respective goals.[47] Depending on whether some deviation from the constraints is permissible, the goal program could be formulated to either: (1) minimize the deviation of the time constraints from the established windows, or (2) relax the requirement that all n cities must be visited.

TCTSP Space State Relaxation Method/Dynamic Programming

Another technique for an exact solution to the TCTSP has been described by Christofides et al.[48] The technique is based on the observations that (i) every "routing" problem is essentially a shortest path problem on some underlying graph with additional constraints, and (ii) dynamic programming can be used to solve shortest path problems subject to constraints on an expanded "state-space graph."

Consider the TSP defined on the graph $G = (X,A)$ with the following additional restrictions. With each vertex x_i in X we associate r_i "time windows," the kth one being defined by the pair of times (e_i^k, u_i^k), where $e_i^k < u_i^k$, $k = 1, \ldots r_i$. We

also assume (without loss of generality) that the "time windows" are disjoint and ordered for each vertex.

Let δ_i be the "processing time" of vertex x_i. A TSP tour starting from a specified vertex x_i visiting every other vertex only once and returning to vertex x_i is required. The time of departure of the traveller from vertex x_i is 0 and the time of visiting vertex x_i is t_i. A feasible tour is one which satisfies $e_i^k \leq t_i \leq u_i^k$ for some $k = 1, \ldots, r_i$ for every $x_i \in X$.

If the traveller goes from vertex x_j to vertex x_i directly, the visiting time t_i is given by:

either: $t_i = t_i + \delta_i + c_{ji}$, if $e_i^k \leq t_i + \delta_i + c_{ji} \leq u_i^k$

or: $t_i = e_i^k$, if $u_i^{k-1} < t_i + \delta_i + c_{ji} \leq e_i^k$

or: $t_i = \infty$ if $t_i + \delta_i + c_{ji} > u_i^{r_i}$.

We wish to find that tour that minimizes the time, T say, of returning to vertex x_1; i.e., we wish to minimize $T \equiv t_{1n} + \delta_{1n} + c_{11}$.

Let $f(S,x_i)$ be the least duration of a feasible path starting at x_i visiting every vertex in the set S and finishing at vertex x_i (excluding time δ_i). For a given S and x_i let:

$$h = \min_{(S-x_i,x_j)\in\hat{\Omega}^{-1}(S,x_i)} [f(S - x_i,x_j) + \delta_i + c_{ji}],$$

where $\hat{\Omega}^{-1}(S,x_i) = \{S-x_i,y \mid y\in(S-x_i) \cap R(x_i)\}$.

We now have:

$f(S,x_i) = h$ if $e_i^k \leq h \leq u_i^k$

e_i^k if $u_i^{k-1} < h \leq e_i^k$

∞ if $h > u_i^{r_i}$.

CONCLUSION

Although both the TSP and the TCTSP are easy to state and understand, they both unleash a wide variety of problems as the number of cities to be visited grows. The solution of the TCTSP appears to differ depending on the number of cities with time constraints. At one end of the spectrum, when only a few of the cities have time windows, the problem is basically a routing problem. The tour can be broken into segments and the cities without time windows inserted into the order in a fairly concise manner. At the other end of the spectrum, when a large number of the cities have time constraints, the problem is primarily a scheduling problem. The time-constrained cities are ordered, and the relatively few remaining cities are fit into the schedule as well as possible. There may be very few choices for the position on the non-constrained cities, so the problem may become almost trivial.

The stickiest problems appear to arise when there are about equal number of cities with time constraints and cities without. With this situation, most of the solution techniques seem to get overwhelmed by the number of possible combinations which need to be tested. Therefore, it appears that more research is required on both the routing problem and the scheduling problem so that their solution techniques can converge into a good solution to the TCTSP.

REFERENCES

1. Lawler, E.L., Lenstra, J.K., Rinnooy Kan, A.H.G. and Shmoys, D.B., The Traveling Salesman Problem, New York: John Wiley & Sons, 1985, p. 1.

2. The original and classical vehicle routing problem can be traced back in time many hundreds of years. It is a problem that arises every winter. The problem: How does Santa Claus do it? Santa has a single vehicle with finite capacity that leaves from a single depot; millions of stochastic demands having tight time windows must be satisfied within a 24-hour period. The charm and challenge of vehicle routing problems are reflected in our wondering about Santa's problem, and in our desire to help Santa out and get into his good graces. He knows whether we (our solutions) are good or bad. Vehicle Routing: Methods and Studies, (Golden, B.V. and Assad, A.A., eds). Amsterdam: Elsevier Science Publishers B.V. (North-Holland), 1988, page v.

3. Christofides, N., Mingozzi, A. and Toth, P., "State-Space Relaxation Procedures for the Computation of Bounds to Routing Problems," Networks, Vol. 11, No. 2, 1981, pp. 146.

4. Christofides, N., Mingozzi, A. and Toth, P., "State-Space Relaxation Procedures for the Computation of Bounds to Routing Problems," Networks, Vol. 11, No. 2, 1981, p. 146.

5. Baker, E.K., "An Exact Algorithm for the Time-Constrained Traveling Salesman Problem," Operations Research, Vol. 31, No. 5, 1983, p. 938.

6. Arnoff, E.A. and Sengupta, S.S., "Chapter 4: Mathematical Programming," Progress in Operations Research, Vol. 1, (Ackoff, R.A., ed.), New York: John Wiley, 1961, p. 151.

7. Murta, K., Linear and Combinatorial Programming, New York: John Wiley & Sons, 1976, p. 410.

8. Miller, D.L. and Pekny, J.F., "Exact Solution of Large Asymmetric Traveling Salesman Problems," Science, Vol. 251, Feb. 1991, p. 755.

9. Foulds, L.R., Combinatorial Optimization for Undergraduates, New York: Springer-Verlag, 1984, p. 5.

10. Lawler, E.L., Lenstra, J.K., Rinnooy Kan, A.H.G. and Shmoys, D.B., The Traveling Salesman Problem, New York: John Wiley & Sons, 1985, p. 58.

11. Haimovich, M., Kan, A.H.G. Rinnooy, and Stogie, L., "Analysis of Heuristics of Vehicle Routing Problems," in Vehicle Routing: Methods and Studies, (Golden, B.L. and Assad, A.A., eds.), Amsterdam: Elsevier Science Publishers B.V. (North-Holland) 1988, p. 47.

12. Miller, D.L. and Pekny, J.F., "Exact Solution of Large Asymmetric Traveling Salesman Problems," Science, Vol. 251, Feb. 1991, p. 754.

13. Murta, K., Linear and Combinatorial Programming, New York: John Wiley & Sons, 1976, pp. 410-411.

14. Baker, K.R., Introduction to Sequencing and Scheduling, New York: John Wiley, 1974, p. 104.

15. Richmond, S.B., Operations Research for Management Decisions, New York: The Ronald Press Co., 1968, pp. 464.

16. Winston, W.L., Operations Research: Applications and Algorithms, Boston: Duxbury Press, 1987, pp. 414.

17. Murta, K., Linear and Combinatorial Programming, New York: John Wiley & Sons, 1976, pp. 450-451.

18. Baker, K.R., Introduction to Sequencing and Scheduling, New York: John Wiley, 1974, pp. 97-98.

19. Baker, K.R., Introduction to Sequencing and Scheduling, New York: John Wiley, 1974, p. 102.

20. Baker, K.R., Introduction to Sequencing and Scheduling, New York: John Wiley, 1974, p. 102.

21. Nemhauser, G.L. and Wolsey, L.A., Integer and Combinatorial Optimization, New York: John Wiley and Sons, 1988, p. 475.

22. Foulds, L.R., Combinatorial Optimization for Undergraduates, New York: Springer-Verlag, 1984, p. 172.

23. Baker, K.R., Introduction to Sequencing and Scheduling, New York: John Wiley, 1974, p. 103.

24. Baker, K.R., Introduction to Sequencing and Scheduling, New York: John Wiley, 1974, pp. 102-103.

25. Thierauf, R.J. and Klekamp, R.C., Decision Making Through Operations Research, 2d ed., New York: John Wiley & Sons, 1975, pp. 526.

26. Nemhauser, G.L. and Wolsey, L.A., Integer and Combinatorial Optimization, New York: John Wiley and Sons, 1988, p. 475.

27. Winston, W.L., Operations Research: Applications and Algorithms, Boston: Duxbury Press, 1987, pp. 808-810.

28. Winston, W.L., Operations Research: Applications and Algorithms, Boston: Duxbury Press, 1987, p. 810.

29. Bodin, L.D. and Golden, B.L., "Classification in Vehicle Routing and Scheduling," Networks, Vol. 11, No. 2, 1981, pp. 103-106.

30. Desrochers, M., Lenstra, J.K., Savelsvergh, M.W.P., Soumis, F., "Vehicle Routing with Time Windows: Optimization and Approximation, in Vehicle Routing: Methods and Studies, (Golden, B.L. and Assad, A.A., eds.), Amsterdam: Elsevier Science Publishers B.V. (North-Holland), 1988, pp. 65-66.

31. Assad, A.A., "Modeling and Implementation Issues in Vehicle Routing," Vehicle Routing: Methods and Studies, (Golden, B.L. and Assad, A.A., eds.), Amsterdam: Elsevier Science Publishers B.V. (North-Holland) 1988, p. 23.

32. Baker, E.K., "An Exact Algorithm for the Time-Constrained Traveling Salesman Problem," Operations Research, Vol. 31, No. 5, 1983, pp. 938-945.

33. Baker, E.K., "An Exact Algorithm for the Time-Constrained Traveling Salesman Problem," Operations Research, Vol. 31, No. 5, 1983, p. 938.

34. Baker, E.K., "An Exact Algorithm for the Time-Constrained Traveling Salesman Problem," Operations Research, Vol. 31, No. 5, 1983, pp. 940.

35. Baker, E.K., "An Exact Algorithm for the Time-Constrained Traveling Salesman Problem," Operations Research, Vol. 31, No. 5, 1983, pp. 941.

36. Baker, E.K., "An Exact Algorithm for the Time-Constrained Traveling Salesman Problem," Operations Research, Vol. 31, No. 5, 1983, pp. 943.

37. Baker, E.K., "An Exact Algorithm for the Time-Constrained Traveling Salesman Problem," Operations Research, Vol. 31, No. 5, 1983, p. 944.

38. Baker, E.K., "An Exact Algorithm for the Time-Constrained Traveling Salesman Problem," Operations Research, Vol. 31, No. 5, 1983, p. 943.

39. Assad, A.A., "Modeling and Implementation Issues in Vehicle Routing," Vehicle Routing: Methods and Studies, (Golden, B.L. and

Assad, A.A., eds.), Amsterdam: Elsevier Science Publishers B.V. (North-Holland) 1988, p. 24.

40. Erschler, J., Fontan, G., Merce, C. and Roubellat, F., "A New Dominance Concept in Scheduling n Jobs on a Single Machine with Ready Times and Due Dates," Operations Research, Vol. 31, No. 1, 1983, pp. 114-127.

41. Solomon, M.M., "On the Worst-Case Performance of Some Heuristics for the Vehicle Routing and Scheduling Problem with Time Window Constraints," Networks, Vol. 16, 1986, p. 161.

42. Solomon, M.M., "On the Worst-Case Performance of Some Heuristics for the Vehicle Routing and Scheduling Problem with Time Window Constraints," Networks, Vol. 16, 1986, pp. 161-74.

43. Kolen, A.W.J., Rinnooy Kan, A.H.G. and Trienekens, H.W.J.M., "Vehicle Routing with Time Windows," Operations Research, Vol. 35, No. 2, 1987, pp. 272.

44. Solomon, M.M., "On the Worst-Case Performance of Some Heuristics for the Vehicle Routing and Scheduling Problem with Time Window Constraints," Networks, Vol. 16, 1986, pp. 164-165.

45. Solomon, M.M., "On the Worst-Case Performance of Some Heuristics for the Vehicle Routing and Scheduling Problem with Time Window Constraints," Networks, Vol. 16, 1986, pp. 164-165.

46. Solomon, M.M., "On the Worst-Case Performance of Some Heuristics for the Vehicle Routing and Scheduling Problem with Time Window Constraints," Networks, Vol. 16, 1986, pp. 165-166.

47. Hillier, F.S., Lieberman, G.J., Introduction to Operations Research, 5th ed., New York: McGraw-Hill, 1990, p. 271.

48. Christofides, N., Mingozzi, A. and Toth, P., "State-Space Relaxation Procedures for the Computation of Bounds to Routing Problems," Networks, Vol. 11, No. 2, 1981, pp. 145-164.

Bibliography

---, Vehicle Routing: Methods and Studies, (Golden, B.L. and Assad, A.A., eds.), Amsterdam: Elsevier Science Publishers B.V. (North-Holland), 1988.

Arnoff, E.A. and Sengupta, S.S., "Chapter 4: Mathematical Programming," Progress in Operations Research, Vol. 1, (Ackoff, R.A., ed.), New York: John Wiley, 1961.

Assad, A.A., "Modeling and Implementation Issues in Vehicle Routing," Vehicle Routing: Methods and Studies, (Golden, B.L. and Assad, A.A., eds.), Amsterdam: Elsevier Science Publishers B.V. (North-Holland) 1988, pp. 7-46.

Baker, K.R., Introduction to Sequencing and Scheduling, New York: John Wiley, 1974.

Baker, E.K., "An Exact Algorithm for the Time-Constrained Traveling Salesman Problem," Operations Research, Vol. 31, No. 5, 1983, pp. 938-945.

Christofides, N., Mingozzi, A. and Toth, P., "State-Space Relaxation Procedures for the Computation of Bounds to Routing Problems," Networks, vol. 11, no. 2, 1981, pp. 145-164.

Desrochers, M., Lenstra, J.K., Savelsvergh, M.W.P., Soumis, F., "Vehicle Routing with Time Windows: Optimization and Approximation, Vehicle Routing: Methods and Studies, (Golden, B.L. and Assad, A.A., eds.) Amsterdam: Elsevier Science Publishers B.V. (North-Holland), 1988, pp. 65-84.

Emmons, H., Flowers, A.D., Khot, C.M. and Mathur,K., Storm: Personal Version 2.0, Oakland: Holden-Day, Inc., pp. 115-25.

Erschler, J., Fontan, G., Merce, C. and Roubellat, F., "A New Dominance Concept in Scheduling n Jobs on a Single Machine with Ready Times and Due Dates," Operations Research, Vol. 31, No. 1, 1983, pp. 114-127.

Foulds, L.R., Combinatorial Optimization for Undergraduates, New York: Springer-Verlag, 1984.

Haimovich, M., Rinnooy Kan, A.H.G., and Stogie, L., "Analysis of Heuristics of Vehicle Routing Problems," Vehicle Routing: Methods and Studies, (Golden, B.L. and Assad, A.A., eds.) Amsterdam: Elsevier Science Publishers B.V. (North-Holland) 1988, pp. 47-61.

Hillier, F.S. and Lieberman, G.J., Introduction to Operations Research, 5th ed., New York: McGraw-Hill, 1990.

Kolen, A.W.J., Rinnooy Kan, A.H.G. and Trienekens, H.W.J.M., "Vehicle Routing with Time Windows," Operations Research, Vol. 35, No. 2, 1987, pp. 266-273.

Lawler, E.L., Lenstra, J.K., Rinnooy Kan, A.H.G. and Shmoys, D.B., The Traveling Salesman Problem, New York: John Wiley & Sons, 1985.

Lenstra, J.K. and Rinnooy Kan, A.H.G, "Complexity of Vehicle Routing and Scheduling Problems," Networks, vol. 11, no. 2, 1981, pp. 221-228.

Miller, D.L. and Pekny, J.F., "Exact Solution of Large Asymmetric Traveling Salesman Problems," Science, vol. 251, Feb. 1991, pp. 754-61.

Murta, K., Linear and Combinatorial Programming, New York: John Wiley & Sons, 1976.

Nemhauser, G.L. and Wolsey, L.A., Integer and Combinatorial Optimization, New York: John Wiley and Sons, 1988.

Richmond, S.B., Operations Research of Management Decisions, New York: The Ronald Press Co., 1968, pp. 461-65.

Solomon, M.M., "On the Worst-Case Performance of Some Heuristics for the Vehicle Routing and Scheduling Problem with Time Window Constraints," Networks, vol. 16, 1986, pp. 161-74.

Thierauf, R.J. and Klekamp, R.C., Decision Making Through Operations Research, 2d ed., New York: John Wiley & Sons, 1975, pp. 525-27.

Winston, W.L., Operations Research: Applications and Algorithms, Boston: Duxbury Press, 1987.

```
MIN      100000 X11 + 255 X12 + 335 X13 + 95 X14 + 175 X15 + 421 X16
     + 130 X17 + 255 X21 + 100000 X22 + 107 X23 + 160 X24 + 131 X25
     + 348 X26 + 207 X27 + 335 X31 + 107 X32 + 100000 X33 + 250 X34
     + 230 X35 + 455 X36 + 264 X37 + 95 X41 + 160 X42 + 250 X43
     + 100000 X44 + 83 X45 + 329 X46 + 138 X47 + 175 X51 + 131 X52
     + 239 X53 + 83 X54 + 100000 X55 + 246 X56 + 220 X57 + 421 X61
     + 348 X62 + 455 X63 + 329 X64 + 246 X65 + 100000 X66 + 220 X67
     + 130 X71 + 207 X72 + 264 X73 + 138 X74 + 220 X75 + 487 X76
     + 100000 X77
SUBJECT TO
       2)   X11 + X12 + X13 + X14 + X15 + X16 + X17 =     1
       3)   X21 + X22 + X23 + X24 + X25 + X26 + X27 =     1
       4)   X31 + X32 + X33 + X34 + X35 + X36 + X37 =     1
       5)   X41 + X42 + X43 + X44 + X45 + X46 + X47 =     1
       6)   X51 + X52 + X53 + X54 + X55 + X56 + X67 =     1
       7)   X61 + X62 + X63 + X64 + X65 + X66 + X67 =     1
       8)   X71 + X72 + X73 + X74 + X75 + X76 + X77 =     1
       9)   X11 + X21 + X31 + X41 + X51 + X61 + X71 =     1
      10)   X12 + X22 + X32 + X42 + X52 + X62 + X72 =     1
      11)   X13 + X23 + X33 + X43 + X53 + X63 + X73 =     1
      12)   X14 + X24 + X34 + X44 + X54 + X64 + X74 =     1
      13)   X15 + X25 + X35 + X45 + X55 + X65 + X75 =     1
      14)   X16 + X26 + X36 + X46 + X56 + X66 + X76 =     1
      15)   X17 + X27 + X37 + X47 + X57 + X67 + X77 =     1
      16)   7 X23 + U2 - U3 <=    6
      17)   7 X24 + U2 - U4 <=    6
      18)   7 X25 + U2 - U5 <=    6
      19)   7 X26 + U2 - U6 <=    6
      20)   7 X27 + U2 - U7 <=    6
      21)   7 X32 - U2 + U3 <=    6
      22)   7 X34 + U3 - U4 <=    6
      23)   7 X35 + U3 - U5 <=    6
      24)   7 X36 + U3 - U6 <=    6
      25)   7 X37 + U3 - U7 <=    6
      26)   7 X42 - U2 + U4 <=    6
      27)   7 X43 - U3 + U4 <=    6
      28)   7 X45 + U4 - U5 <=    6
      29)   7 X46 + U4 - U6 <=    6
      30)   7 X47 + U4 - U7 <=    6
      31)   7 X52 - U2 + U5 <=    6
      32)   7 X53 - U3 + U5 <=    6
      33)   7 X54 - U4 + U5 <=    6
      34)   7 X56 + U5 - U6 <=    6
      35)   7 X57 + U5 - U7 <=    6
      36)   7 X62 - U2 + U6 <=    6
      37)   7 X63 - U3 + U6 <=    6
      38)   7 X64 - U4 + U6 <=    6
      39)   7 X65 - U5 + U6 <=    6
      40)   7 X67 + U6 - U7 <=    6
      41)   7 X72 - U2 + U7 <=    6
      42)   7 X73 - U3 + U7 <=    6
      43)   7 X74 - U4 + U7 <=    6
      44)   7 X75 - U5 + U7 <=    6
      45)   7 X76 - U6 + U7 <=    6
END
INTE    49

LP OPTIMUM FOUND AT STEP      31
OBJECTIVE VALUE =    1125.42900
FIX ALL VARS.(    17)  WITH RC >    100.000
SET       X73 TO <=     0 AT     1, BND=   -1127.      TWIN= -1253.          68
SET       X62 TO >=     1 AT     2, BND=   -1273.      TWIN= -1130.         105

NEW INTEGER SOLUTION OF     1273.00000      AT BRANCH       2 PIVOT      105
```

APPENDIX A

OBJECTIVE FUNCTION VALUE

1) 1273.00000

VARIABLE	VALUE	REDUCED COST
X11	.000000	99904.000000
X12	.000000	294.000000
X13	.000000	84.000000
X14	1.000000	.000000
X15	.000000	91.000000
X16	.000000	163.000000
X17	.000000	.000000
X21	.000000	112.000000
X22	.000000	99992.000000
X23	1.000000	-191.000000
X24	.000000	18.000000
X25	.000000	.000000
X26	.000000	43.000000
X27	.000000	30.000000
X31	.000000	93.000000
X32	.000000	.000000
X33	.000000	99603.000000
X34	.000000	9.000000
X35	.000000	.000000
X36	.000000	51.000000
X37	1.000000	-12.000000
X41	.000000	.000000
X42	.000000	200.000000
X43	.000000	.000000
X44	.000000	99906.000000
X45	1.000000	.000000
X46	.000000	72.000000
X47	.000000	9.000000
X51	.000000	91.000000
X52	.000000	182.000000
X53	.000000	.000000
X54	.000000	.000000
X55	.000000	99928.000000
X56	1.000000	.000000
X57	.000000	.000000
X61	.000000	163.000000
X62	1.000000	225.000000
X63	.000000	42.000000
X64	.000000	72.000000
X65	.000000	.000000
X66	.000000	99580.000000
X67	.000000	30.000000
X71	1.000000	-9.000000
X72	.000000	203.000000
X73	.000000	-30.000000
X74	.000000	.000000
X75	.000000	93.000000
X76	.000000	186.000000
X77	.000000	99827.000000
U2	4.000000	.000000
U3	5.000000	.000000
U4	.000000	.000000
U5	1.000000	.000000
U6	2.000000	.000000
U7	6.000000	.000000

ROW	SLACK OR SURPLUS	DUAL PRICES
2)	.000000	90.000000
3)	.000000	43.000000

```
      5)          .000000          91.000000
      6)          .000000         102.000000
      7)          .000000         -72.000000
      8)          .000000          47.000000
      9)          .000000        -186.000000
     10)          .000000         -51.000000
     11)          .000000        -341.000000
     12)          .000000        -185.000000
     13)          .000000        -174.000000
     14)          .000000        -348.000000
     15)          .000000        -220.000000
     16)          .000000            .000000
     17)         2.000000            .000000
     18)         3.000000            .000000
     19)         4.000000            .000000
     20)         8.000000            .000000
     21)         5.000000            .000000
     22)         1.000000            .000000
     23)         2.000000            .000000
     24)         3.000000            .000000
     25)          .000000            .000000
     26)        10.000000            .000000
     27)        11.000000            .000000
     28)          .000000            .000000
     29)         8.000000            .000000
     30)        12.000000            .000000
     31)         9.000000            .000000
     32)        10.000000            .000000
     33)         5.000000            .000000
     34)          .000000            .000000
     35)        11.000000            .000000
     36)         1.000000            .000000
     37)         9.000000            .000000
     38)         4.000000            .000000
     39)         5.000000            .000000
     40)        10.000000            .000000
     41)         4.000000            .000000
     42)         5.000000            .000000
     43)          .000000            .000000
     44)         1.000000            .000000
     45)         2.000000            .000000

NO. ITERATIONS=      105
BRANCHES=     2 DETERM.=  1.000E     0
BOUND ON OPTIMUM:  1129.714
FLIP       X62 TO <=          0 AT     2 WITH BND=    -1129.7140
SET       X47 TO >=     1 AT     3, BND=   -1134.      TWIN= -1130.          140
SET       X63 TO <=     0 AT     4, BND=   -1136.      TWIN= -1345.          165
SET       X36 TO <=     0 AT     5, BND=   -1136.      TWIN= -.1000E+31      191
DELETE       X26 AT LEVEL      6
DELETE       X36 AT LEVEL      5
DELETE       X63 AT LEVEL      4
FLIP       X47 TO <=          0 AT     3 WITH BND=    -1129.7140
SET       X74 TO >=     1 AT     4, BND=   -1134.      TWIN= -1150.          254
SET       X63 TO <=     0 AT     5, BND=   -1136.      TWIN= -1345.          284
SET       X36 TO <=     0 AT     6, BND=   -1136.      TWIN= -.1000E+31      315
DELETE       X26 AT LEVEL      7
DELETE       X36 AT LEVEL      6
DELETE       X63 AT LEVEL      5
FLIP       X74 TO <=          0 AT     4 WITH BND=    -1150.1430
SET       X46 TO <=     0 AT     5, BND=   -1165.      TWIN= -1184.          409
SET       X64 TO >=     1 AT     6, BND=   -1214.      TWIN= -1192.          450
SET       X43 TO <=     0 AT     7, BND=   -1218.      TWIN= -1377.          476
SET       X42 TO <=     0 AT     8, BND=   -1223.      TWIN= -1390.          487
SET       X35 TO <=     0 AT     9, BND=   -1229.      TWIN= -1344.          507
```

X17	.000000	130.000000
X21	.000000	255.000000
X22	.000000	100000.000000
X23	1.000000	107.000000
X24	.000000	160.000000
X25	.000000	131.000000
X26	.000000	348.000000
X27	.000000	207.000000
X31	.000000	335.000000
X32	.000000	107.000000
X33	.000000	100000.000000
X34	.000000	250.000000
X35	.000000	230.000000
X36	.000000	455.000000
X37	1.000000	264.000000
X41	.000000	95.000000
X42	.000000	160.000000
X43	.000000	250.000000
X44	.000000	100000.000000
X45	1.000000	83.000000
X46	.000000	329.000000
X47	.000000	138.000000
X51	.000000	175.000000
X52	.000000	131.000000
X53	.000000	239.000000
X54	.000000	83.000000
X55	.000000	100000.000000
X56	1.000000	246.000000
X57	.000000	220.000000
X61	.000000	421.000000
X62	1.000000	348.000000
X63	.000000	455.000000
X64	.000000	329.000000
X65	.000000	246.000000
X66	.000000	100000.000000
X67	.000000	220.000000
X71	1.000000	130.000000
X72	.000000	207.000000
X73	.000000	264.000000
X74	.000000	138.000000
X75	.000000	220.000000
X76	.000000	487.000000
X77	.000000	100000.000000
U2	3.000000	.000000
U3	4.000000	.000000
U4	.000000	.000000
U5	1.000000	.000000
U6	2.000000	.000000
U7	5.000000	.000000

ROW	SLACK OR SURPLUS	DUAL PRICES
2)	.000000	.000000
3)	.000000	.000000
4)	.000000	.000000
5)	.000000	.000000
6)	.000000	.000000
7)	.000000	.000000
8)	.000000	.000000
9)	.000000	.000000
10)	.000000	.000000
11)	.000000	.000000
12)	.000000	.000000
13)	.000000	.000000
14)	.000000	.000000

16)	.000000	.000000
17)	3.000000	.000000
18)	4.000000	.000000
19)	5.000000	.000000
20)	8.000000	.000000
21)	5.000000	.000000
22)	2.000000	.000000
23)	3.000000	.000000
24)	4.000000	.000000
25)	.000000	.000000
26)	9.000000	.000000
27)	10.000000	.000000
28)	.000000	.000000
29)	8.000000	.000000
30)	11.000000	.000000
31)	8.000000	.000000
32)	9.000000	.000000
33)	5.000000	.000000
34)	.000000	.000000
35)	10.000000	.000000
36)	.000000	.000000
37)	8.000000	.000000
38)	4.000000	.000000
39)	5.000000	.000000
40)	9.000000	.000000
41)	4.000000	.000000
42)	5.000000	.000000
43)	1.000000	.000000
44)	2.000000	.000000
45)	3.000000	.000000

NO. ITERATIONS= 1153
BRANCHES= 32 DETERM.= -1.000E 0

```
                        !OR TCTSP FORMULATION FOR LINDO
MIN T8 - T1             ! OBJECTIVE FUNCTION
ST
T2 - T1 > 95
T3 - T1 > 160
T4 - T1 > 250
T5 - T1 > 83
T6 - T1 > 329
T7 - T1 > 138
T3 - T2 > 255           ! THESE CONSTRAINTS MUST BE ABSOLUTE VALUES
T4 - T2 > 250
T4 - T3 > 250
T5 - T2 >  83
T5 - T3 >  83
T5 - T4 >  83
T6 - T2 > 329
T6 - T3 > 329
T6 - T4 > 329
T6 - T5 > 329
T7 - T2 > 138
T7 - T3 > 138
T7 - T4 > 138
T7 - T5 > 138
T7 - T6 > 138
T8 - T2 > 95
T8 - T3 > 160
T8 - T4 > 250
T8 - T5 > 83
T8 - T6 > 329
T8 - T7 > 138
T7 > 140                ! TIME WINDOW FOR YACHATS (LOWER LIMIT)
T7 < 160                ! TIME WINDOW FOR YACHATS (UPPER LIMIT)
```

APPENDIX B