Solution to the Traveling Salesman Problem Using Heuristic Function Maximization

Chien Y. Huang*
Grumman Aerospace Corporation, Bethpage, New York

and

Stephen H. Lane†
Princeton University, Princeton, New Jersey

An approach for solving the traveling salesman problem as formulated in the AIAA Artificial Intelligence Design Challenge is presented. The approach is based on heuristic measures involving intercity fares, city values, and value-to-fare ratios. Other heuristic factors take into account the available budget, the expected trip cost, and the likelihood of low-cost hub cities. Monte Carlo simulations show that this heuristic method produces optimal solutions over 90% of the time for problems dominated by fare, and 55% (84% in a revised program) of the time for problems not dominated by fare or value.

Introduction

THE solution to the traveling salesman problem is a classic exercise in operations research and linear programming. The objective in the most basic problem is to minimize the cost (or distance) of travel between cities. Additional constraints such as fixed travel budgets and fare uncertainty are present in more complicated formulations. The problem is theoretically well-posed and suitable for application of general optimization techniques.

Solutions to the traveling salesman problem have been extensively investigated in the past.[1-5] The survey paper of Bellmore and Nemhauser[1] divides the various approaches into two main categories: exact techniques and approximate techniques. Each of these categories can in turn be decomposed further into three parts: a starting scheme, a route generation scheme, and a termination rule. When the solution found is always optimal, the method is exact. Integer programming and dynamic programming are examples of exact techniques. If at the termination of the algorithm there is no guarantee that the solution is optimal, the technique is approximate.

Exact methods suffer from increasingly larger memory requirements (to store intermediate results) and longer running times as the size of the problem is expanded. This is due to the "curse of dimensionality" associated with combinatorial expansions. In addition, large numbers of loop constraints and intermediate variables are required to eliminate unconnected subtours in the integer programming and dynamic programming solutions. Approximate methods attempt to produce reasonable (i.e. close to optimal) results using much lower computational and memory requirements. This paper presents an approximate solution to the traveling salesman problem based on a heuristic approach.

The basic goal of a heuristic approach is the formulation of a cost function that steers the search for a solution toward the final desired state. An early application of heuristics to the traveling salesman problem appeared in Ref. 6 Advances in the area of artificial intelligence in recent years have led to a more thorough understanding of heuristic search techniques.[7-9] One method is known as hill climbing, or the A^* algorithm.[10] It carries out the search from a city (or node) by proceeding along increasing directions of a heuristic function. The complexity and optimality of the A^* algorithm have been examined, and it has been shown that, due to its formulation, optimal results are produced when a consistency assumption is satisfied.[11] Modifications to the A^* algorithm for bidirectional methods, which simultaneously carry out searches from the initial state forward and from the goal state backward, have also been proposed.[12]

The next section identifies and analyzes some of the generic scenarios of the traveling salesman problem. This is followed by an outline of the proposed heuristic approach, including a description of the heuristic measures used and details of their implementation. The performance of the heuristic algorithm is then evaluated using Monte Carlo simulation techniques. The results are summarized and extensions of this approach to larger problems are discussed in the conclusions.

Problem Analysis

Generic Scenarios

The precise statement of the AIAA Traveling Salesman Design Challenge can be found in Ref. 13. Although the problem is a modified version of the basic traveling salesman problem, the key to its solution is still the logic that determines the next city to visit from any given city. An exhaustive approach would explore all the possible city routes to find the optimal solution. It is, however, quite likely that there are indications implicit in the data on how to proceed from one city to the next. These clues manifest themselves as generic scenarios. By identifying these scenarios, strategies can be devised and algorithms developed to exploit the embedded information, thus allowing the problem to be solved using smaller computational resources. Some of the basic generic scenarios present in the traveling salesman problem include:

1) Problems dominated by city value. This is the case where the distribution of city values determines the travel plan.

2) Problems dominated by fare. In this case the fare structure favors traveling along some specific routes regardless of the city values.

3) Problems not dominated by city value or fare. This is the most general case. Tradeoffs must be made between intercity fares and city values when searching for the optimal solution.

4) Problems dominated by a hub city. A hub is defined as a city whose to and from fares are substantially lower than all the

Presented as Paper 87-2338 at the AIAA Guidance, Navigation, and Control Conference, Monterey, CA, Aug. 17–19, 1987; received July 2, 1987; revision received Dec. 11, 1987. Copyright © 1988 by Grumman Aerospace Corp. Published by the American Institute of Aeronautics and Astronautics, with permission.

*Senior Engineer, Flight Controls. Member AIAA.

†Graduate Student. Student Member AIAA.

other cities. In this case, considerable savings can be realized if certain expensive flight segments are rerouted through the hub city.

5) Problems with local maxima. This is the case where the fare data contains locally optimal routes that may mislead simple algorithms that seek globally optimal routes.

Constraint-Related Scenarios

The AIAA problem states that the expected trip cost C_e must be within the allotted budget B with probability P greater than some lower bound P_l. C_e is a function of the first-class fare multiplier M_{fc} and the layover cost C_l, while P is a function of the probability of a first-class upgrade P_f. Large values of P_l, P_f, M_{fc}, and C_l imply that the trip should be terminated with a large reserve in the budget in order to meet the problem constraints, whereas small values allow the final trip cost to be closer to the budgeted amount. These constraints play a role in determining whether the spending pattern should be liberal or conservative.

The amount of the remaining budget at any stage of the search significantly affects the route the salesman takes as well. When the budget is small or nearly exhausted, it may be prudent to visit the cities with higher values first. If a less "valuable" city is visited first, the budget may not allow the more "valuable" city to be visited later. This situation shifts the search strategy toward a problem dominated by city values.

The requirement that Boston be visited after Los Angeles (in order to claim Los Angeles' value) implies that visiting Boston before Los Angeles should be discouraged. It is possible, however, that Los Angeles may be a low-value city (relative to the average value). In this case, going through Los Angeles before Boston is not as desirable. The exact strategy is dependent on the intercity fare structure and values of the cities.

The Heuristic Approach

The approach taken in this paper to solve the traveling salesman problem is to construct a route by successively including cities into the tour based on their "attractiveness" as measured by a heuristic cost function. This is done by ranking each city in terms of the heuristic function's value, and conducting limited searches in the directions of the cities that scored well. The approach is basically a hill-climbing algorithm, as the search moves toward the goal state by maximizing the heuristic cost function.

The power of heuristics lies in its ability to quantify "intangible" clues that can lead searches in the direction of the optimal solution. These clues are largely generated based on the physics of the problem and the intuition of the designer. From the previous analysis of the generic scenarios, city values, intercity fares, and value-to-fare ratios are chosen as the primary heuristic measures. The first heuristic measure is sensitive to cities with high values, and it is useful for problems represented by scenario A. The second heuristic measure directs searches along the paths of lowest fare, and it is intended to respond to situations similar to scenario B. The last heuristic parameter reflects the tradeoff between value and fare. A city with a high value and low fare is desirable. Value-to-fare ratios provide the necessary compromise for the general class of problems outlined in scenario C. Before these heuristic measures can be manipulated in the form of a heuristic cost function, they must be *normalized* (i.e., divided by) with respect to their average values. This ensures that the terms of the heuristic function are compared based on their relative importance, not their absolute values. Furthermore, this normalization allows more complex heuristic measures to be constructed from these primary elements by arithmetic manipulations.

The presence of a hub city as described in scenario D poses a potential problem, as it may continuously attract the search in its direction and prevent the heuristic algorithm from first considering the other cities. Hub cities are defined as any city with an average fare, to and from all the other cities, that is 0.8 of a standard deviation below the average fare in the problem. If a city is identified as a hub, it is temporarily removed from the list of candidate cities used in the search for the optimal route. After a route segment is chosen, an attempt is made to reroute the tour through the hub. If this yields a lower cumulative trip cost, the hub city is spliced into the route.

Since searches always tend to move toward cities with the lowest fare and highest value, a "telescoping" heuristic measure is employed to avoid local maxima as described by scenario E. This heuristic attempts to look through the "hills" to determine if they are worth climbing. Hills that obscure "steep" valleys on the other side are to be avoided as they usually lead to nonoptimal routes. The depth of the telescoping should be chosen as a function of the computational resources at hand and the desired accuracy of the solution. Taken to an extreme, if the depth of the telescoping provides "tunnel vision" through all the hills of the heuristic cost function, then the heuristic approach can be expected to produce the same results as an exhaustive method.

In order to meet the probability constraints, a heuristic measure called a reduction constant (RC) is also devised. It is based on P_l, P_f, M_{fc}, and C_l, and it is used to approximate the additional sum that should be set aside to ensure that the expected cost is less than the budget constraint. The calculation of RC is described in the Appendix.

Algorithm Description

A flow chart outlining the heuristic algorithm is shown in Fig. 1. In the preprocessing phase, average city values and average fares to and from each city are computed along with their corresponding standard deviations. The intercity fares and the city values are then normalized by their respective averages, and normalized value-to-fare ratios are computed. Hub cities are also identified during this phase. Since Los Angeles is constrained to be visited before Boston in order to pick up its valuation, it is made more "attractive" by increasing its (normalized) value by an amount proportional to the product of its value and the normalized standard deviation of the city values (thus, a high-value Los Angeles would be made more "desirable" and vice versa). Similarly, Boston is made less attractive by lowering its value proportional to the increase in Los Angeles' value. The last step of the preprocessing phase is the computation of the reduction constant. This constant will be subtracted from the total available budget along with the intercity fare and layover cost as each new city is added to the tour.

In the iterative phase of the heuristic algorithm in Fig. 1, the heuristic function

$$h_j = -W_f \cdot F_{n_j} + W_v \cdot V_{n_j} + W_{vf} \cdot V_{fn_j} + W_t \cdot T_j \tag{1}$$

is computed at each stage of the search for every j city. F_{n_j}, V_{n_j}, and V_{fn_j} are the normalized fare, value, and value-to-fare ratios of each city, while W_f, W_v, and W_{vf} are their corresponding weightings. T_j is the telescoping heuristic measure, and W_t is its weighting. The first three weightings in Eq. (1) and set to 1.0, as there is little reason to favor one heuristic measure over the other. (This was found to be untrue. See the Postscript.) The telescoping heuristic is based on the maximum value-to-fare ratio of the i cities that are accessible from the candidate city j, i.e.,

$$T_j = \max_{i \neq j} (V_{fn_i}) \tag{2}$$

This provides a correction to the search direction by examining the local optimality conditions at a depth of two cities further along on the tour. In situations where the optimal route requires going "downhill" for three or more cities before the hill climbing begins, this heuristic measure may not point toward the global maximum. The depth of two was chosen for its computational simplicity. W_t is used to trade off the present optimality conditions vs their future expectations. It has been set equal to 0.1, as there are 10 cities to consider at any given time.

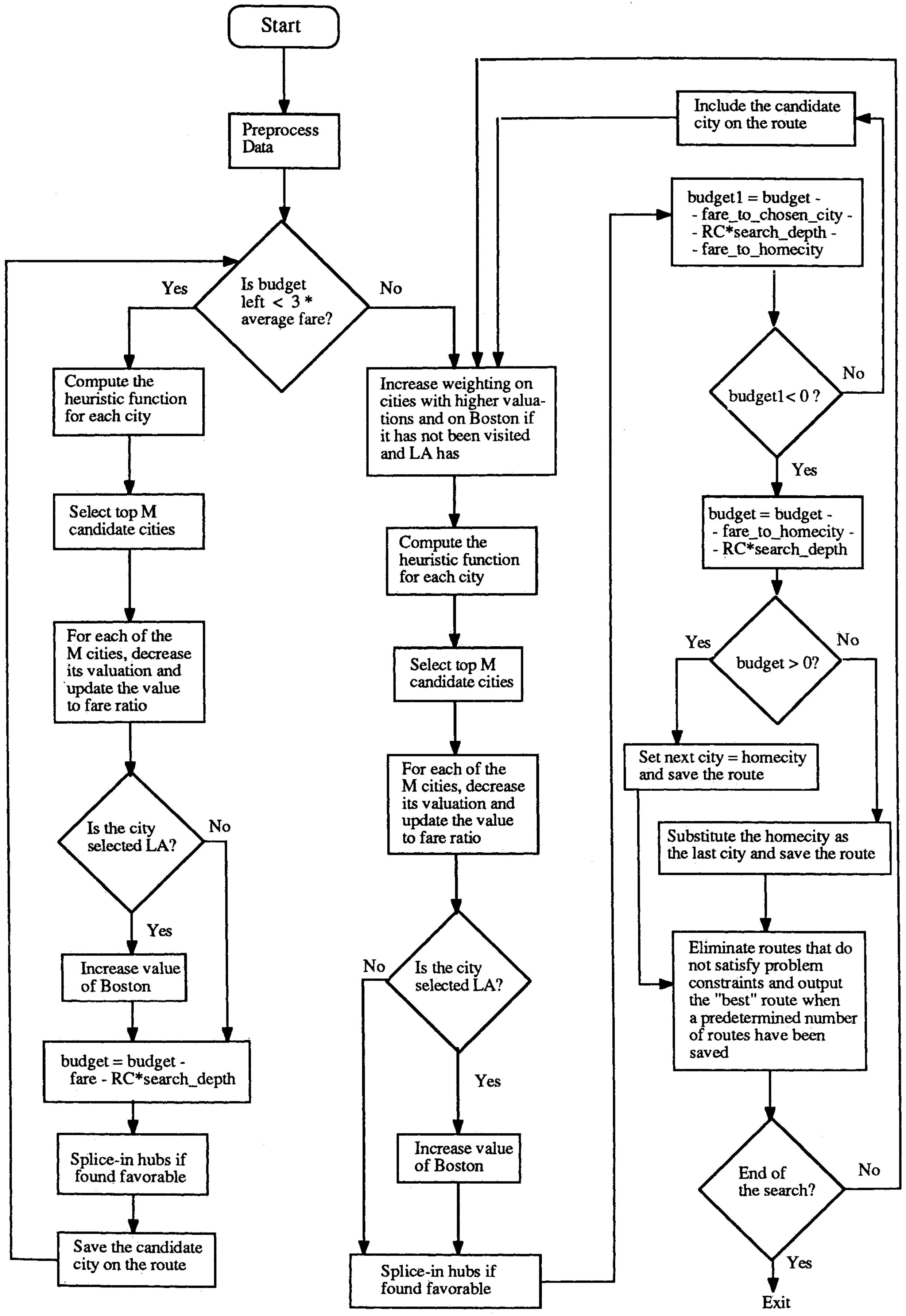
Start
Preprocess Data
Is budget left < 3 * average fare?
Yes
No
Compute the heuristic function for each city
Select top M candidate cities
For each of the M cities, decrease its valuation and update the value to fare ratio
Is the city selected LA?
No
Yes
Increase value of Boston
budget = budget - fare - RC*search_depth
Splice-in hubs if found favorable
Save the candidate city on the route
Increase weighting on cities with higher valuations and on Boston if it has not been visited and LA has
Compute the heuristic function for each city
Select top M candidate cities
For each of the M cities, decrease its valuation and update the value to fare ratio
Is the city selected LA?
No
Yes
Increase value of Boston
Splice-in hubs if found favorable
Include the candidate city on the route
budget1 = budget - - fare_to_chosen_city - - RC*search_depth - - fare_to_homecity
budget1 < 0 ?
No
Yes
budget = budget - - fare_to_homecity - - RC*search_depth
budget > 0?
Yes
No
Set next city = homecity and save the route
Substitute the homecity as the last city and save the route
Eliminate routes that do not satisfy problem constraints and output the "best" route when a predetermined number of routes have been saved
End of the search?
No
Yes
Exit

Fig. 1 Program flowchart.

At every stage of the search, all cities except for pre-identified hub cities are considered as candidates, even if they have been visited. This handles situations where a city is only accessible (in terms of the cost) from one other city, which may appear, hence, two or more times in the optimal route. This differs from many other approaches that eliminate cities from consideration once they have been visited.

Computation of h allows the attractiveness of each city to be ranked. A search to the first M candidate cities with the highest scores is conducted. M is a parameter that is determined by the budget and average fare, and it varies with the depth of the search. In the early stages of the search, M is set large, as a wrong city choice here will prevent the optimal route from being found. M is decreased during the intermediate and final stages of the search to reflect the fact that there are fewer attractive candidates to be considered. The heuristic approach reduces to an exhaustive search if M is set equal to the total number of cities minus one.

Before moving from the present city to a candidate city, some housekeeping must take place. First, the value of the chosen city is set to zero, so that it becomes less attractive during subsequent searches. If the chosen city is Los Angeles, the value of Boston is increased by two city-value standard deviations. The budget is then reduced by the intercity fare, layover cost, and the reduction constant. If a hub is present, an attempt is made to reroute the tour through it. If this proves to be cost effective, the route is then modified to include the hub city.

To ensure that the last stop is always the home city, the search enters into a final "homing phase" when the available budget falls below three times the average intercity fare. The homing strategy is to temporarily reroute the tour from the candidate city to the home city, and check if the tour is still under budget. If it is, the candidate city is made part of the route and the search continues. Otherwise, the search is terminated, and the home city is added as the final city on the tour. Once in the homing phase, remaining nonzero city values are augmented by one city value standard deviation. This shifts the problem to one dominated by value (scenario B). Boston's value is increased even more if the route at that point includes Los Angeles but not Boston. This follows from the previous discussions. An exceptional situation can occur when all the cities have been visited, but the homing phase has never been entered. In this case, a provision in the program automatically designates the next stop as the home city.

After a route is constructed, it is temporarily saved (e.g., in an array). When a prearranged number of routes has been generated, the program enters a postprocessing loop, whose function is to examine the solutions for the highest-valued itinerary that meets the probability constraint. If one is found, it is reported and stored away. This route is checked against additional solutions generated by subsequent postprocessing stages, and it is replaced (and notified) if a better (in terms of value or cost) route is found. The optimal route is the "best" route found by this iterative process at the end of the search.

Monte Carlo Simulation

Optimality Measures

Since there is no guarantee that the heuristic approach will generate an optimal solution, it is desirable to know how close the heuristic solution is to the optimal. This requires a measure of optimality to be devised. The primary factor to consider in such a metric is the total amount of city values accumulated. Actual travel cost should also be taken into account, but only if it is greater than the optimal cost (travel cost can always be kept lower by visiting fewer cities). A preliminary measure of optimality (R) that embodies these factors can be defined as

$$R = \frac{(V/V^*)}{(C/C^*)} \qquad \text{if } (C - C^*) \geq 0 \tag{3a}$$

$$R = V/V^* \qquad \text{if } (C - C^*) < 0 \tag{3b}$$

where V and C are the accumulated city values and cost of the heuristic solution, and V^* and C^* are the corresponding quantities of the optimal route.

If the heuristic approach uses a smaller number of searches than an exhaustive method and still produces a solution that is optimal, it is both computationally and technically superior. Nonoptimal heuristic solutions that are very close to optimal [as defined by Eq. (3)] may still be acceptable, especially when their lower computational effort is considered. Modifying Eq. (3) to reflect the number of searches made allows a new measure of optimality (R_s) to be defined as

$$R_s = R + (1 - R)(1 - N/N^*) \qquad \text{if } R > T_s \tag{4a}$$

$$R_s = R \qquad \text{if } R \leq T_s \tag{4b}$$

where N is the number of heuristic searches made, N^* the number of exhaustive searches made, and T_s is a threshold defined by

$$T_s = (V^* - V_{\text{avg}})/V^* \tag{5}$$

where V_{avg} is the average city value. T_s determines when a heuristic solution is too far from the optimal to be considered, no matter how small the number of searches made. According to Eqs. (4) and (5), the worst case acceptable is a heuristic solution that misses a city of average value. The term $(1 - R)$ in Eq. (4) represents the "price" paid for suboptimality, while $(1 - N/N^*)$ gauges the amount of computation and memory savings. Their product represents a tradeoff between optimality and computational resources. When N is small, R_s is very close to 1. When $N = N^*$, $R_s = R$. Reasonable heuristic solutions should yield high values for R_s (for example, in excess of 0.95).

Simulation Results

The algorithm was programmed in Pascal and run on an IBM PC-AT using as the input the original problem stated in Ref. 13. The optimal solution was found in roughly 10 s, although the search continued on for another four minutes. This performance can be improved as described in the Postscript.

To further evaluate the algorithm, Monte Carlo simulations were conducted for a seven-city problem to evaluate the effectiveness of the heuristic approach. Optimality measures R and R_s were computed based on optimal solutions obtained from an exhaustive approach using

$$N^* = \sum_{k=0}^{(L - L_o)} \frac{(L - 1)!}{k!}$$

searches, where L is the total number of cities, and L_o is the number of cities on the optimal tour. Combinatorial expansions of cities were used with the assumption that no city other than the home city would be visited more than once. Optimal routes were defined as tours of cities that satisfy all the problem constraints and have the largest accumulated value. In many situations there were more than one optimal solution. In this case, the solution with the lowest cost was used in the optimality measures of Eqs. (3) and (4).

Problems dominated by fare were created by setting all the city values to be the same and by generating random intercity

Table 1 Results for problems dominated by fares

Avg. R	Std. R	Avg. R_s	Std. R_s	Avg. N	Std. N
0.96	0.046	0.98	0.038	400	70

Table 2 Results for the problem not dominated by fare or value

Avg. R	Std. R	Avg. R_s	Std. R_s	Avg. N	Std. N
0.93	0.074	0.96	0.076	300	80

fare data with a Gaussian distribution having a mean of 350 and a standard deviation of 75. After evaluating 100 random data sets, results indicate that the heuristic solution was optimal 92% of the time. The average R, R_s, and N along with their respective standard deviations can be found in Table 1.

Problems not dominated by value or fare were also investigated. These problems were constructed by generating random city value data with a uniform distribution on the interval [1,20], and random fare data having the same Gaussian distribution as above. Results after 100 simulated runs indicate that for this class of problems, the heuristic solution was optimal 55% of the time. The associated average R, R_s, and N along with their respective standard deviations can be found in Table 2.

Conclusions

Optimal solutions for random data sets were found 92% of the time for problems dominated by fare, and 55% of the time for problems dominated by value-to-fare ratio. In cases where the heuristic solution found was not optimal, the high values of the optimality measures R and R_s indicate that results were close to optimal. The magnitudes of the heuristic weightings determine the "blending" of the generic scenarios and have a direct bearing on the search direction and depth. Given the results, it appears that the weightings have been turned to favor problems dominated by fare.

Larger problems imply higher computational requirements. To reduce the amount of processing, the number of nodes expanded at any stage of the search can be made to vary according to the data and time constraint. A more powerful telescoping heuristic also must be devised, since finding of a solution based on local conditions, such as fare and value, may not be possible. Generation of this heuristic requires additional insights into the problem. Lastly, to improve the optimality of the solutions, the heuristic weightings should be fine-tuned. It is believed that a set of weightings can be found, as demonstrated in the Postscript, to produce good results over a wide range of problems.

Appendix: Calculation of the Reduction Constant

The details of the algorithm that generate the RC are described herein. The strategy is to first assume that the optimal route is an N-city itinerary, and then determine how much of the budget should be set aside in order to ensure meeting the probability constraint.

Step 1: Estimate the number of cities N that may be visited based on the budget B, the average fare F_{avg}, and the layover cost C_l, i.e.,

$$N = \text{truncate}[B/(F_{\text{avg}} + C_l)] + 1 \tag{A1}$$

Step 2: Estimate the probability P_m of having m first-class segments in an N-city route

$$P_m = \binom{N}{m}(P_f)^m(1 - P_f)^{N-m} \tag{A2}$$

where

$$\binom{N}{m} = \frac{N!}{m!(N-m)!} \tag{A3}$$

is the standard formula of combinations; m is initially set to N.

Step 3: Calculate the probability P of the route having $N, N-1, \ldots m$ first-class segments, which are assumed to result in trip costs higher than the budget:

$$P = P - P_m \tag{A4}$$

P is equal to 1.0 at the beginning.

Step 4: If $P > P_l$ and $m \neq 0$, decrement m and go back to step 2. Or if $P > P_l$ and $m = 0$, set RC to zero or a small value and exit. Else compute the cost C of traveling first class in m segments of the N-city route:

$$C = [m \cdot M_{fc} + (N - m)] \cdot F_{\text{avg}} + N \cdot C_l \tag{A5}$$

The variable m at this point represents the maximum number of first-class segments that can be allowed in order for an N-city itinerary to meet the probability constraint, while C represents the cost associated with this scenario. It is observed that if only routes with $C < B$ are admitted as possible solutions, then $P > P_l$ and the probability constraint would be satisfied. This leads to the next step.

Step 5: If $C \leq B$, then set RC to a small value. Else calculate the reduction constant as

$$RC = \frac{2(C - B)}{N(N + 1)} \tag{A6}$$

Step 6: Exit.

$(C - B)$ is the budget amount or buffer that must be reserved at end of the N-city route to ensure that the probability of the expected cost being under budget meets the probability constraint. The factor $2/N(N + 1)$ in Eq. (A6) is based on the strategy that the reduction constraint is to be subtracted in an amount proportional to the depth of search, i.e, at each stage of the search

$$B = B - \text{intercity-fare} - RC \cdot \text{depth-of-search} \tag{A7}$$

The reason is to prevent setting aside too much from the budget for routes shorter than N, which would be the case if the reduction constant is subtracted in fixed amount. Equation (A6) is derived by requiring that the budget buffer be equal to the sum of the reduction constants, i.e.,

$$\begin{aligned}(C - B) &= \sum_{k=1}^{N} k \cdot RC \\ &= RC \cdot \sum_{k=1}^{N} k \\ &= RC \frac{N(N+1)}{2}\end{aligned} \tag{A8}$$

Solving Eq. (A8) for RC leads to the expression shown in Eq. (A6).

Postscript

This addendum describes some "bugs" that were discovered in the original program (not in the algorithm) after the test data described in Ref. 13 were made available. It also presents updated results using the corrected version of the program.

The two cases (the Hard-P and Soft-Maze) where no solution was found by the original program were first dealt with. The problem was due to the oversight of not subtracting the reduction constant in the last segment of the route. In those two cases, the last leg in many of the routes constructed tended to be very expensive, which led to final itineraries not meeting the probability constraint. This problem was corrected by ensuring that RC is subtracted from the budget for the last segment. If the budget falls below zero, the itinerary is shortened by eliminating the last city.

Subtraction of the reduction constant in each stage of the search was changed from constant to that of linearly increasing function. This follows from the discussion in the Appendix.

The choice of the heuristic measures, as indicated by the simulation data, was poor. After some investigation, it was found that the primary drivers of the searches are the normalized fare and value alone. Since the value-to-fare ratio was given the same weighting as those two heuristics, it guided the search along erroneous paths in many cases. By increasing the fare and value weightings W_f, W_v from 1.0 to 2.0, the effect of V_{fn} was lessened, and better results were obtained.

Since there was no hub in the test data sets, the check for hubs was subsequently turned off. This action had very small impact on the execution times.

The last change made to the original program was on the frequency of reporting the results. In the original program when 150 routes had been produced, the postprocessing would start to check for the optimal route. It was found that in most instances the optimal route was produced within the first 30 searches; hence, the program was revised to examine the results (and notify if necessary) each time 30 new routes are generated. This number can be set lower, but it is felt that it will lead to the reporting of two many intermediate results.

With these modifications, the new execution times (on an IBM PC-AT) for obtaining the *optimal* (except for the Easy-P case) solution are shown in Table 3. Note that they are considerably lower than the times reported in Ref. 13. As further tests, the Monte Carlo simulation was repeated. The revised program found the optimal solution in 92% of the times for problems dominated by fare and 84% of times for problems not dominated by fare and value. The average and standard deviations of R, R_s, and N for these two cases are shown in Tables 4 and 5. As indicated by the results, the performance is improved.

Table 3 New execution times (in seconds) for the test problems

Original	Easy-p	Ran-fare	Hard-P	Soft-Maze	Clusters	Iso-1	Iso-2
4.0	24.0	5.5	5.5	14.5	2.5	3.0	3.0

Table 4 New results for problems dominated by fare

Avg. R	Std. R	Avg. R_s	Std. R_s	Avg. N	Std. N
0.99	0.035	0.99	0.034	360	60

Table 5 New results for problems not dominated by value or fare

Avg. R	Std. R	Avg. R_s	Std. R_s	Avg. N	Std. N
0.99	0.019	1.0	0.0073	270	73

Finally, we realize that these changes were made with a great deal of hindsight. Because of the nature of the approach, however, the weakness can only be uncovered by checking the algorithm against a large variety of data. Furthermore, the changes do not pertain to the algorithm; they were mostly programming bugs, except for the case of the heuristic weighting changes. With the revisions, it is expected that the program will produce very reasonable results for many other problems.

References

[1]Bellmore, M. and Nemhauser, G. L., "The Traveling Salesman Problem: A Survey," *Operations Research*, Vol. 16, No. 3, May–June 1968, pp. 538–557.

[2]Lin, S., "Computer Solutions of the Traveling Salesman Problem," *The Bell System Journal*, Vol. 44, Dec. 1965, pp. 2245–2269.

[3]Held, M. and Karp, R. M., "The Traveling Salesman Problem and Minimum Spanning Trees," *Operations Research*, Vol. 18, Nov.–Dec. 1970, pp. 1138–1162.

[4]Gomory, R. E., "The Traveling Salesman Problem," *Proceedings of the IBM Scientific Computing Symposium on Combinatorial Problems*, IBM, White Plains, N.Y., 1964.

[5]Kolman, B. and Beck, R. E., *Elementary Linear Programming with Applications*, Academic Press, New York, 1980.

[6]Karg, L. L. and Thompson, G. L., "A Heuristic Approach to Solving Traveling Salesman Problems," *Management Science*, Vol. 10, Jan. 1964, pp. 225–248.

[7]Nilsson, N. J., *Problem-Solving Methods in Artificial Intelligence*, McGraw-Hill, New York, 1971.

[8]Nilsson, N. J., *Artificial Intelligence*, Tioga Publishing Co., Palo Alto, CA, 1980.

[9]Winston, P., *Artificial Intelligence*, Addison Wesley, Reading, MA, 1984.

[10]Hart, P. E., Nilsson, N. J., and Raphael, B., "A Formal Basis for the Heuristic Determination of Minimum Cost Paths," *IEEE Transactions on System, Science, and Cybernetics*, SSC-4(2), Institute of Electrical and Electronic Engineers, New York, 1968, pp. 100–107.

[11]Martelli, A., "On the Complexity of Admissible Search Algorithms," *Artificial Intelligence*, Vol. 8, No. 1, 1972, pp. 1–13.

[12]Pohl, I., "Bidirectional Search," *Machine Intelligence*, Vol. 6, edited by B. Meltzer and D. Michie, Edinburgh University Press, Edinburgh, Scotland, 1971, pp. 127–140.

[13]Deutsch, O. L., "The A.I. Design Challenge—Background, Analysis, and Relative Performance of Algorithms," AIAA Paper 87-2339, 1987; also in this issue.