Multiple Algorithm Solution to the Artificial Intelligence Design Challenge

Karl W. Doty*
The Aerospace Corporation, El Segundo, California

This paper responds to the Artificial Intelligence Design Challenge at the 1987 AIAA Guidance, Navigation, and Control Conference. It describes a computer program that solves a variation of the classical traveling salesman problem that is representative of problems encountered in complex guidance and control systems. The problem is to find a tour connecting some or all of 11 cities that maximizes the total value of the cities visited while meeting a travel budget constraint. The solution procedure used was to construct several candidate tours using a variety of approaches, then adjust the highest-value tour found by solving the traveling salesman problem to find the optimal city ordering. An efficient bounding procedure was used to reduce the number of computations required to evaluate the stochastic budget constraint. The program consistently produced in a small amount of time the best answers known on 25 data sets designed to test the program's robustness.

Introduction

THIS paper responds to the Artificial Intelligence Design Challenge at the AIAA 1987 Guidance, Navigation, and Control Conference. The challenge is to create a computer program that runs on a personal computer and solves a modification of the well-known traveling salesman problem (TSP).

Details of the problem are presented in Ref. 1. The solution approach taken was to try several different methods of constructing tours. This was to provide robustness in that one construction method may work well on one set of problem data but a different method may work better on other problem data. After all construction methods were tried, the best tour found was perturbed in order to try to increase the value or decrease the cost. The final step was to find the optimal ordering for the selected set of cities by solving the traveling salesman problem for this set.

Several complications made the challenge problem more difficult than the standard traveling salesman problem. The major difference is that the set of cities to be visited, as well as their order, must be determined. The second complication is that transit costs are stochastic, making the cost of the tour a random variable. Evaluating a particular quantile of this distribution was expensive, so bounding procedures were developed to reduce the computation time. The local constraint of visiting Boston after Los Angeles required many lines of computer code but did not significantly impact the algorithms.

Tour Construction Methods

The tour construction methods are used to construct tours that attempt to maximize the total city value while satisfying the constraints. The best of these tours is then improved by the tour improvement methods described in the following section. Several different but related methods are used to construct tours, and the best tour found, regardless of the method used, is selected.

All of the construction methods are special cases of the "insertion method," which is a standard approach for the TSP.[2,3] An initial tour is formed that starts and ends at the home city. Another city is chosen and inserted so that the tour is now home city-second city-home city. A third city is chosen and is inserted in the best position (either before or after second city), with "best" meaning minimum cost. This process continues until either all cities are in the tour or no city not yet in the tour can be inserted without violating the budget constraint.

The methods differ in the order that the cities are chosen for insertion. The first method, which is surprisingly good, randomly orders the cities. The second method inserts the cities in order of their value. The third method uses a value-to-cost ratio criterion and also allows several partial tours to be saved at each stage.

Random Insertion

The random insertion method is simple yet powerful. The cities are ordered randomly, and each is then inserted in the least costly position until the budget constraint is used up. This is a standard heuristic for the traveling salesman problem. The cities are then reshuffled, and the insertion procedure is tried again. This is repeated until either the desired number of trials has taken place or the desired amount of time has elapsed. The best result is saved to compare with the other methods.

As an example, suppose with the sample data the random order is

MSY BOS MSP DEN ATL SEA PHX DTT DFW CHI LAX

The tour starts with the home city DTT and then inserting the first city becomes DTT-MSY-DTT. Putting BOS after MSY is $60 (in coach fare) cheaper than putting it before, so the tour becomes DTT-MSY-BOS-DTT. Continuing the insertion procedure with the next three cities, the tour becomes DTT-MSP-DEN-MSY-ATL-BOS-DTT. The best spot for the next city, SEA, is after MSP. However, the resulting tour violates the budget constraint with probability 0.085, which exceeds the constraint in the sample data of 0.05. Thus SEA is rejected, and the next city (PHX) is inserted (in this case after DEN). This satisfies the budget constraint. DTT is skipped, since it is already in the tour as the home city. DFW and CHI can both be successfully inserted, giving the tour

DTT-CHI-MSP-DEN-PHX-DFW-MSY-ATL-BOS-DTT

This violates the budget constraint with probability 0.015, which is acceptable. The best place for the last city, LAX, is after DEN, but this gives a tour that violates the budget constraint with probability 0.63. The final tour has a value of 74 and an expected cost of $2888 ($2088 from the transit costs, $800 in daily expenses).

Received June 29, 1987; presented as Paper 87-2329 at the AIAA 1987 Guidance, Navigation, and Control Conference, Monterey, CA, Aug. 17-19, 1987; revision received Jan. 4, 1988.

*Engineering Specialist.

One complication that arises is that it is costly (in terms of time) to determine whether a proposed tour satisfies the budget constraint. This is because the costs are stochastic. The bounding procedures that will be described can usually determine whether the tour satisfies or violates the constraint. If the bounds cannot determine whether the constraint is satisfied, the entire probability distribution could be calculated and the constraint evaluated, but this is costly. Instead, the tour is saved for later evaluation. If another tour is later found that does satisfy the budget constraint and has either a higher value or equal value and lower cost, then the saved tour is inferior and can be eliminated from consideration.

In 24 of the 25 test cases constructed by the author (see the following for a description of the test cases), the random insertion method was able to find a tour with a value equal to the best known value within 10 s (which is the time used in the submitted program). In this amount of time, between 50 and 80 tours were examined for each test case. The number depended primarily on the number of cities included before the budget was violated.

Value-Ordered Insertion

The value-ordered insertion method orders the cities by value. As in the random insertion method, they are inserted into the tour at the least costly spot, one at a time, until no city can be inserted without violating the budget constraint. Ties between cities with equal value are broken by selecting the one that produces the least costly tour. The method is very fast, requiring about 1 s. In 16 out of 25 test cases, a tour was found with value equal to the best known.

In the sample data, the highest value cities, in order, are LAX, CHI, BOS, and DFW. Inserting these one at a time produces the tour DTT-CHI-LAX-DFW-BOS-DTT. The remaining cities all have equal value. PHX produces the tour with the smallest cost, so it is inserted at its best place, after LAX. Inserting ATL next produces the lowest cost tour, and it is put after DFW. MSY is next, and it is placed between DFW and ATL. DEN is then placed after PHX, resulting in the tour DTT-CHI-LAX-PHX-DEN-DFW-MSY-ATL-BOS-DTT. This tour has a value of 88 and violates the budget constraint with probability 0.005, which is within the constraint. Next comes MSP, and placing it at its best position, after CHI, produces a tour that violates the budget constraint with probability 0.42. This is unacceptable, so MSP is rejected. Inserting SEA is even worse, so the tour DTT-CHI-LAX-PHX-DEN-DFW-MSY-ATL-BOS-DTT is the final tour found by this method. Incidentally, this is the best tour by any method for the sample data.

Beam Search/Value-Cost Ratio Insertion

This is another insertion method, with the criteria for city selection being the maximum ratio of value to expected cost. At each stage (stages corresponding to a number of cities), W partial tours are retained, where W is called the beamwidth.

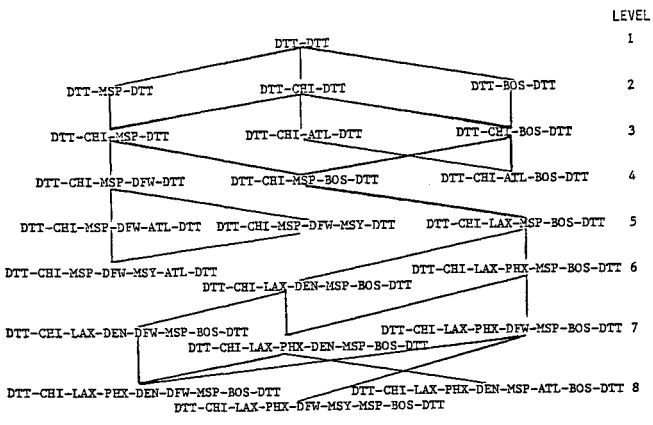

Fig. 1 "Beam search" search tree.

This is called a "beam search" because the search tree has a constant width, and thus resembles a beam. This is one of a number of search methods often used for artificial intelligence problems.[4] The submitted program uses a value of 15 for W.

The partial tours start, as in the previous methods, from the home city to itself. Each of the $N-1$ cities remaining is then considered to enter the tour, and the total value and total expected cost are computed. The W partial tours that give the highest value of the value-cost ratio are saved; (if W is greater than $N-1$ all $N-1$ that can be constructed are saved). Each of these partial tours is considered in turn, and each city not already in the tour is inserted in its best spot. This produces $W(N-2)$ tours (assuming $W < N-1$), but only the W with the highest value-cost ratio are saved. Duplicates with all cities in the same positions are eliminated, but if two tours have the same cities in different positions, both are retained. This process continues until either all N cities are in the tour or no additional cities can be inserted without violating the budget constraint. At this point, the tour with the highest value is returned as the solution found by the beam search method.

As an example, let $W = 3$ with the sample data. Figure 1 shows the search tree obtained, with the tree pruned to show only the retained partial tours at each level. The final tour found has a value of 82 and expected cost of \$2615.20. Notice that by level 5, MSP is in all the partial tours, although it is not in the optimal tour previously mentioned. Since cities are not removed, it is impossible for this method to find the optimal tour. With a beamwidth of 10, though, the optimal tour was found.

In the 25 test cases, a beamwidth of 1 was sufficient to find 11 tours with a value equal to the best known. With a beamwidth of three, 17 were found, with 10, 21 were found, and a beamwidth of 15 was sufficient to find such a tour in all 25 cases. The time required is linearly related to both the beamwidth and the number of cities in the final tour. For a beamwidth of 15, the time in seconds was about $1.2*NT$.

Tour Improvement Methods

The three tour construction methods described, used in combination, are likely to find the optimal solution. In some cases, though, it is possible to increase the value or decrease the cost of the best tour found. This section discusses three methods of tour improvement. The first, city substitutions, can increase the value of the tour by substituting a city not in the tour for one or more that is. The second, city position switching, exchanges one or more cities in one part of the tour for one or more in another part to try and reduce the expected cost. Both of these methods are fast. The third, the traveling salesman solution, is comparatively slow. However, it guarantees finding the minimum cost solution for the given set of cities. It is applied last to only the highest value tour found. If a reduction in cost is found, the city substitution method is again tried to attempt to increase the tour value.

City Substitutions

The city substitution method tries to exchange cities not in the tour for cities that are in the tour in order to increase the value. Each city not in the tour is considered in order of decreasing value. Candidates for exchange that are in the tour are any city or chain of two or more cities with a smaller value than the city under consideration for insertion. These cities in the tour are taken out, and the city outside the tour is inserted into its best position, which may or may not be the same as the position where the other cities were removed. If the resulting tour satisfies the budget constraint, it is taken as the current "best tour," since it is known to have a higher value than the previous best. The method also allows an exchange in which cities have equal value, if this reduces the cost. Furthermore, if a city can be added without deleting any other cities, this is also done.

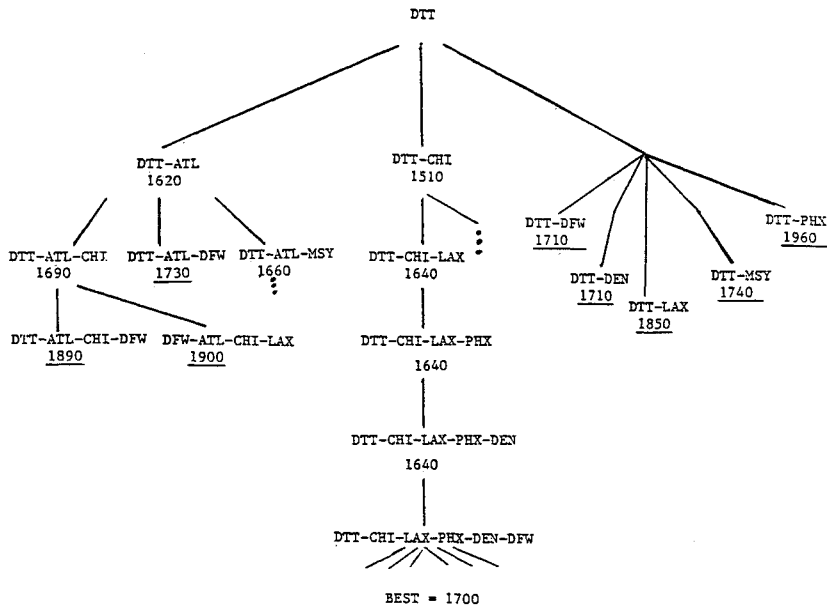

Fig. 2 Partial traveling salesman search tree (numbers indicate cost lower bounds).

As an example, the beam search method found the following tour for the sample data with a beam width of 3:

DTT-CHI-LAX-PHX-DEN-DFW-MSP-BOS-DTT

The DFW-MSP-BOS coach cost is \$600. Exchanging ATL for MSP gives a cost of \$520, a savings of \$80. This then allows MSY to be added between DFW and ATL, producing the optimal tour.

City Position Switches

The position switches are designed to quickly find small changes in the tour that reduce the expected cost. There are two types of switches allowed. The first takes two cities in the tour and exchanges their positions. The second, called Or-Opt in the literature,[5] takes 1, 2, or 3 consecutive cities in the tour and inserts them between two other cities in the tour. A switch is kept if it reduces the expected cost. If a switch is found, the process is restarted. This continues until no switches reduce cost.

This method requires less than a second for execution. It was placed in a loop along with the tour substitution method previously described. If one or the other was successful, the other was then tried in order to improve the tour still further. These two methods produced a good candidate for the final perturbation, the traveling salesman solution.

Traveling Salesman Solution

Given a set of cities, the minimum cost ordering can be found by solving the classical traveling salesman problem using a branch and bound algorithm, such as those described in Ref. 2. While known algorithms for solving the TSP require execution time that is exponential in the number of cities, this is acceptable for sufficiently small problems. In the challenge problem, there are at most 11 cities, and the problem can be solved within 1 min in all cases tried. Unlike the city position switch method, solving the TSP guarantees finding the optimal ordering.

A good initial tour is required in order to quickly solve the TSP. The tour construction methods are designed to find this tour, and the other improvement methods can make it even better. The benefit of a good initial tour is that a tight upper bound on the cost is known, which allows many possible alternative orderings to be excluded.

Figure 2 illustrates part of the traveling salesman search tree for the sample data problem, starting with the tour obtained by the value-ordered insertion method. The coach cost for this tour is \$1700, and the objective of the algorithm is to determine if any reordering of the cities produces a smaller coach cost (the total expected cost for a given set of cities is a linear function of the coach cost). The top node of the search tree consists only of the home city, DTT. The first partial tour constructed is DTT-ATL. Using the bounding procedure described, a lower bound of \$1620 is obtained for the coach cost. Since this is below the best known cost of \$1700, a better order of the cities may exist starting with this partial tour, so it is explored further. BOS cannot be added due to the city precedence constraint, so CHI is added next. The bound of \$1690 is still below the minimum known cost, so more cities are added. Two possible cities are DFW and LAX, both of which produce bounds greater than \$1700, and thus they are eliminated. All other extensions of this partial tour also produce bounds greater than \$1700, so the partial tour of DTT-ATL-CHI does not produce a complete tour with a smaller cost. The branching procedure then backs up the search tree and explores the next three-city partial tour, DTT-ATL-DFW. The branching and bounding process continues until it is determined that DTT-ATL will not produce a cheaper tour. BOS is rejected as a second city due to the city precedence constraint, so the next two-city partial tour considered is DTT-CHI. This partial tour can be extended all the way to six cities, at which point there are only three cities left (ATL, BOS, and MSY), so all six possible orderings are considered. The best one has a cost of \$1700, and the tour produced is the same as the one with which the algorithm started. The search through the tree continues, but no better tour is found.

Special Considerations

Stochastic Travel Costs

The travel costs in this problem are not constant but are rather random variables. Between each pair of cities, the salesman has a probability *PFC* that he will have to travel first class, which will increase the coach cost by a fraction *FCP*. For a tour, there is a probability distribution of possible costs. The travel cost can take any value between traveling coach on all links and traveling first class on all links. For a tour with *NT* links, there are 2^{NT} values of the travel cost, depending on which ones the first-class premium must be paid.

One effect of the stochastic travel costs is that the expected cost of a tour is used to compare two tours with equal values. The bigger effect of the stochastic travel costs relates to the budget constraint, which takes the form that the budget must be satisfied with a probability *GPC*. In order to determine whether the constraint is satisfied, the 2^{NT} subsets of links that

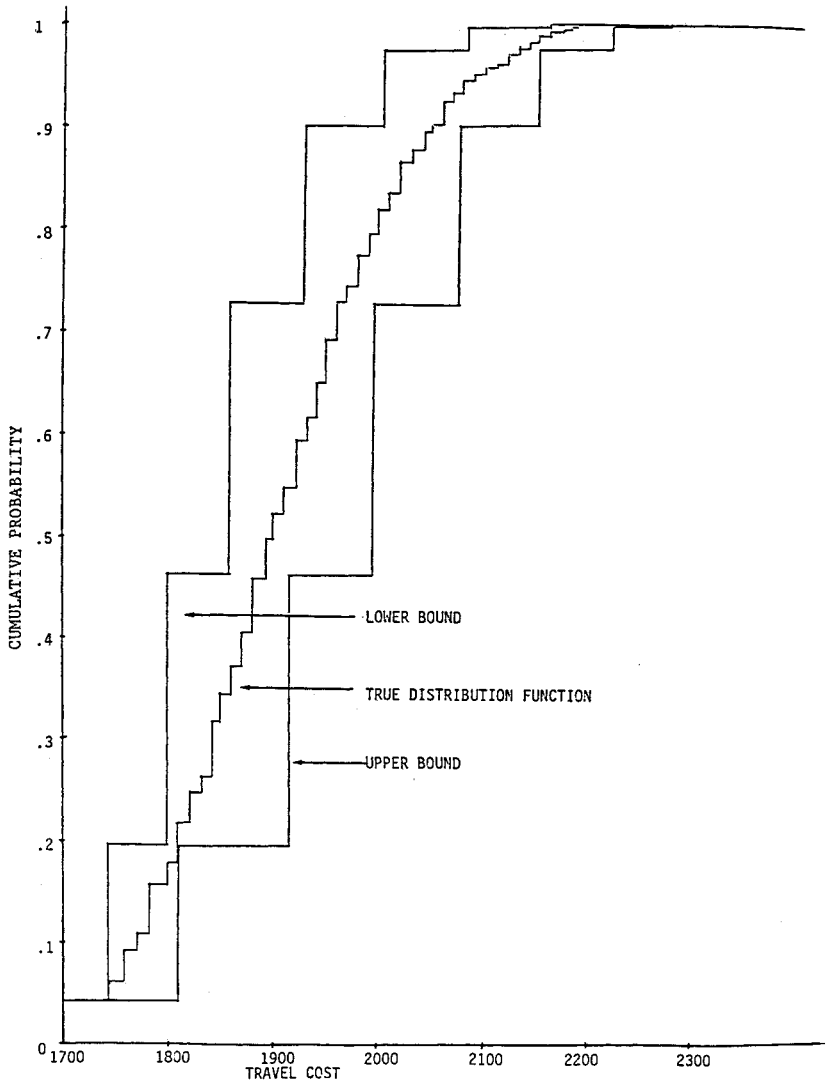

Fig. 3 Cost distribution function and bounds.

might have the first-class premium applied to them must be enumerated. The probability that K particular links require the first-class premium is $PFC^K*(1-PFC)^{NT-K}$. Denoting these K links as the set S, the cost of this tour is

$$\sum_{[T(i),T(i+1)]\in S} c_{T(i)T(i+1)}*(1+FCP)$$

$$+\sum_{[T(i),T(i+1)]\notin S} c_{T(i)T(i+1)}$$

Summing the probabilities of all sets of links that have a cost less than or equal to the budget constraint gives the probability that the constraint is satisfied.

The process of computing the probability that the budget constraint is satisfied is time consuming. It would be desirable to be able to determine whether the constraint is satisfied without actually computing the probability that it is satisfied. A bounding procedure can be used to make this determination.

Partition the set of 2^{NT} tour costs into $NT+1$ subsets, based on how many links require the first-class premium. Consider the subset with K first-class links. The maximum cost in the subset has the K most expensive (based on coach cost) links with the premium, and the other $NT-K$ links at coach. Suppose that this cost satisfies the budget constraint. Since it is the largest cost in that subset, all members of that subset satisfy the budget constraint. Furthermore, all members of all subsets that have fewer than K first-class links satisfy the budget constraint. If the sum of the probabilities of these subsets is greater than the global probability constraint, then the proposed tour satisfies the budget constraint.

The probability of a subset with K first-class links is

$$P(NT,K)=\binom{NT}{K}PFC^K(1-PFC)^{NT-K}$$

Suppose that

$$\sum_{J=0}^{K} P(NT,J) > GPC$$

but

$$\sum_{J=0}^{K-1} P(NT,J) < GPC$$

Then this bounding procedure says that if adding the first-class premium to the K most expensive links produces a tour cost that satisfies the budget constraint, then the tour satisfies the global probability constraint. If the tour cost does not satisfy the budget constraint, however, this does not say that the tour violates the global probability constraint but that it is uncertain.

For each NT, the critical value of K obtained can be computed in advance, since it does not depend on the tour found. An array $KCRIT(NT)$ is constructed with these values. They are then referred to throughout the program to determine if a proposed tour satisfies the budget constraint.

The same type of bounding procedure can be used to produce a lower bound on the tour cost. Suppose that all of the tour costs in the subsets with *KCRIT* or more first-class links violate the budget constraint. This would be true if adding the premium to the *KCRIT least* expensive links violates the constraint. Then the tour violates the global probability constraint, since the sum of the probabilities of those subsets exceeds $1 - GPC$.

To evaluate a proposed tour, both bounds are used. First the upper bound is computed. If this satisfies the budget constraint, then the tour is accepted. If it does not, then the lower bound is computed. If the lower bound is greater than the budget constraint, the tour is rejected. If the upper bound is greater than the budget constraint and the lower bound is lower, then either the exact probability of satisfying the budget constraint is computed by enumerating all 2^{NT} tour costs, or the tour is saved for future study.

Figure 3 plots the upper and lower bounds for the probability distribution of the travel cost of the optimal tour for the sample data, along with the true distribution. Of particular interest are the bounds at 0.95, since this is the global probability constraint. The lower bound is \$2004, and the upper bound is \$2152; (the true value is \$2080). Adding the \$800 travel cost to the upper bound gives \$2952. Since this is below the budget constraint, the tour is known to be acceptable without actually computing the probability that the budget constraint is satisfied.

Multiple City Visits

The challenge problem allows each city to be visited more than once. Even though subsequent visits do not accrue the value, they may be desirable in order to reduce the cost. For the standard TSP, it is easy to allow multiple city visits. For each pair of cities, the minimum cost path is found using a shortest-path algorithm such as the Floyd-Warshall algorithm.[6] The cost of this path replaces the single link cost in the cost matrix, and the TSP is solved without allowing repeated visits. Each link in the resulting optimal tour is then replaced with the shortest path, yielding a final tour that may include multiple visits.

Because of complications in the challenge problem, particularly the stochastic costs and the desire for short execution time, this approach was not taken. The similar approach that was taken was to allow a single city to be inserted between any pair of cities. The best city, if any, is found by computing for each pair of cities i and j the city k that minimizes $c_{ik} + c_{kj} - c_{ik} + EXP$. The EXP term comes from the fact that the expenses are incurred even though the value is not accrued. If the preceding expression is less than 0, the best k is saved in a matrix of insertion cities. Given a proposed tour of non-reported cities, this matrix is referred to insert additional cities if desirable prior to the computation of tour costs.

City Precedence Constraints

There is a local constraint in the problem in that the value for Los Angeles is not accrued unless Boston follows it in the tour. Thus, the only reason for having Los Angeles without Boston or having Los Angeles after Boston would be if Los Angeles reduced the cost of the tour. This can be found by using the modified cost matrix previously discussed. As a result, the tour construction and perturbation methods should require that Los Angeles not appear unless Boston follows it.

This requirement is easy to handle in the perturbation methods. Each perturbation is checked to make sure it does not violate the constraint. For example, consider the perturbation that exchanges two cities in the tour. Assuming both Los Angeles and Boston are in the tour, Los Angeles cannot be switched with Boston or any city following it, and Boston cannot be switched with Los Angeles or any city preceding it. Any other switch is acceptable. In the traveling salesman solution, Boston is not allowed to enter the partial tour until Los Angeles is in it, again assuming both cities are in the complete tour.

Table 1 Number of cases (out of 25) maximum value found

	Random insertion	Value-based insertion	Beam search/ ratio insertion	Final program
Before perturb	24	16	25	25
After perturb	25	18	25	25

Table 2 Number of cases (out of 25) maximum value and minimum cost found

	Random insertion	Value-based insertion	Beam search/ ratio insertion	Final program
Before perturb	17	7	18	20
After perturb	25	18	25	25

There are two ways to handle the constraint in the construction procedures. One is not to allow Los Angeles to be in the tour until Boston is in it and then force Los Angeles before Boston. The second is to allow Los Angeles in, then force Boston after it if Boston is to be inserted. If Boston is not in the final tour, Los Angeles is removed from the tour, and the insertion procedure continues until the budget constraint is reached. The first approach is used in the beam search method, while the second is used in the random insertion and the value-based insertion. The advantage of using both approaches rather than just one is that any bias toward having or not having these cities in the tour is removed.

Computer Program Description and Testing

The program was written in FORTRAN 77 and used the Lahey compiler. It contained about 1300 lines of source code and was divided into 20 subroutines. The executable file was 77,000 bytes long. The program was tested on 25 test cases, constructed to be similar to the sample data and yet cover a wide range of possibilities. The data that were varied were city values, transit costs between cities, budget constraint, first-class probability, first-class premium, and global probability constraint. A random number generator was used to select values for each of these parameters. For the city values, numbers between 5 and 20 were selected. The budget value was between \$1500 and \$4000. The first-class premium was between 0 and 50%, while the first-class probability was between 0 and 0.6. For the global probability constraint, one of the four values was selected: 0.5, 0.8, 0.95, and 0.99.

The transit costs are the most important item in selecting a good algorithm. Some heuristic algorithms are good on highly structured costs, such as ones that are symmetric and satisfy the triangle inequality but perform poorly on more random costs. To cover a range of possibilities, two methods were used, each for 12 test cases. For the first 12 test cases, costs were purely random, between \$50 and \$750. For the second 12, costs were selected that approximately satisfied the symmetry and triangle inequality conditions, just like the sample data. For the test case 25, the sample data was used.

The following tables show how well each of the construction algorithms and the final program performed when the 25 test cases were run. The random insertion method was limited to 10 s, and the beam search method used a beamwidth of 15. Table 1 shows how often the best known value was found, both before and after the tour perturbation methods were run. Table 2 shows how often both the best known value and corresponding lowest known cost were found. The final program found the best known answer (both highest known value and lowest known cost) in 80% of the test cases using just the tour construction methods and in all of the test cases with the perturbation methods. The last line of Table 2 shows that the beam search with the perturbation methods found the best answer in all test cases. It was decided to retain the other construction methods as well to increase robustness. If the time limit of

20 min had been much smaller, the beam search alone would have been used.

Execution times for the complete program ranged from 16–65 s, with an average of 29 s. Of this time, about a third was spent on the random insertion, a third was spent on the beam search, and the remaining third was mostly spent on the traveling salesman algorithm.

Conclusion

The program is able to solve problems in the form of the challenge problem quickly and produce an optimal or near-optimal solution. The most important parts of the program that contributed to solving the problem are 1) use of several good tour construction methods to increase program robustness, 2) traveling salesman solution to optimally reorder a set of cities, 3) fast bounds to compute in most cases, whether or not the stochastic budget constraint is met, and 4) accommodation of multiple city visits.

The program was tuned to the specifics of the challenge problem. Since there was a maximum of 11 cities, it was feasible to run the traveling salesman algorithm on one set of cities. With fewer cities (or more allowed execution time), the algorithm could have been run on many sets of cities, while with more cities, heuristics would have been required instead of the optimal TSP algorithm. Two aspects of the solution procedure, though, are applicable to a wide range of problems. The first is that using several heuristic techniques can produce a more robust solution procedure than just using one. The second is that heuristic methods (tour construction) can be combined with optimization methods (traveling salesman algorithm) to produce answers that are better than those arrived at through heuristics alone but are much faster than complete optimization. The program shows that this combined heuristic and optimization approach is able to come up with good solutions to stochastic combinatorial optimization problems such as the one posed.

Acknowledgment

This work was sponsored by the Aerospace Sponsored Research Program.

References

[1]Deutsh, O. L., "The A.I. Design Challenge—Background, Analysis, and Relative Performance of Algorithms," *Proceedings of the AIAA Guidance, Navigation and Control Conference*, Vol. 1, AIAA, New York, 1987, pp. 465–476.

[2]Lawler, E. L., Lenstra, J. K., Rinnooy Kan, A. H. G., and Shmoys, D. B., (eds.), *The Traveling Salesman Problem*, Wiley, Chichester, Great Britain, 1985.

[3]Golden, B. L., Bodin, L., Doyle, T., and Stewart, W., "Approximate Traveling Salesman Algorithms," *Operations Research*, Vol. 28, 1980, pp. 694–711.

[4]Winston, P. H., *Artificial Intelligence*, Addison-Wesley, Reading, MA, 1984, Chap. 4.

[5]Or, I., "Traveling Salesman-Type Combinatorial Problems and their Relation to the Logistics of Regional Blood Banking," Ph.D. Thesis, Northwestern Univ., Evanston, IL, 1976.

[6]Floyd, R. W., "Algorithm 97, Shortest Path," *Communications of the ACM*, Vol. 5, 1962, p. 345.