Rapid Solution of Constrained Traveling Salesman Problems

Clifford D. DeJong*
Kaman Sciences Corporation, Colorado Springs, Colorado

This paper addresses the AIAA Guidance, Navigation, and Control Artificial Intelligence Design Challenge. The approach taken to solve this problem begins with a Monte Carlo code that quickly generated tours satisfying all the constraints in order to find excellent tours to measure the performance of algorithms. The solution begins with an initial tour that starts and ends with the home city and satisfies the local constraint of visiting Boston after Los Angeles. Variants of this small initial tour are created by inserting remaining cities into the tour at the most expensive link. Each variant tour is expanded by successively adding the city that maximizes the tour value while minimizing the cost and remaining within budget. The optimal tour is the best tour over all variants of the initial tour. This method attains a value of 88 for the sample problem, with a 0.005 probability of exceeding the budget, in one minute on an AT&T 6300 with 8087 coprocessor.

Introduction and Summary

THE AIAA Guidance, Navigation, and Control Artificial Intelligence Design Challenge is described in Ref. 1. This paper describes an approach that resulted in a very fast code that achieves near-optimal results on several data sets. The coding was done in TURBO PASCAL V3.0 on an AT&T 6300 PC with 8087 coprocessor.

A Monte Carlo code that quickly generated tours satisfying all the constraints was developed in order to find excellent solutions to test algorithms. Exhaustive solutions of two special cases were also evaluated. A heuristic method was developed that began by setting up a small initial tour that satisfied the local constraint and was within budget. The initial tour for the baseline problem is pictured in Fig. 1. The tour is then expanded by examining all possibilities of inserting a remaining city into each leg of the tour. The best possibility, i.e., the best combination of city and leg to maximize the ratio of the new tour value divided by the new tour cost, is added to the tour if it can be done within budget. The process continues until no more cities can be added. This approach gave very good results for the sample problem and other contrived problems. The optimal tour for the baseline case with value 88 is shown in Fig. 2.

Two refinements were added to the basic approach. Guided by a better Monte Carlo tour, it was observed that the small initial tour could be varied by inserting one of the remaining cities at the most expensive leg and then expanding the tour as described, often leading to a better solution. The original approach was then modified to select the optimal tour as the best over all variants. This modification improved performance somewhat in many cases, in exchange for an increase in computer time. The other refinement was to add logic to find intermediate stopovers in the air fare matrix that reduced the cost of travel. This is crucial in some cases that can be envisioned and involves negligible additional computation time.

This exercise has shown that a Monte Carlo code is a way to develop problem understanding and to give benchmark results for algorithm assessment. Exhaustion was possible to find true optimality over a limited set. Special constraints can be handled by careful selection of the initial tour. Incremental expansion of the tour by marginal gains is fast and produced good tours. Improvement of this method by varying the initial tour was relatively costly. For a 10% improvement, execution time increased by an order of magnitude. Most of the execution time was in cost evaluation; a simpler cost formulation reduces run time by a factor of 10.

Monte Carlo and Exhaustive Tours

One difficulty with algorithm development in a problem of this type is that the optimal solution is not available to use as a measure of how well a particular algorithm performs. Enumerating all possible tours is expensive; the 10 cities (11 less the home city) can produce 3,628,800 distinct tours of length 10 if each city is visited only once, with 1,814,400 tours of length nine, etc. This was evaluated over a very limited set of inputs. The optimal tour value for the baseline case was 88. For the baseline with a home city of Denver, the optimal tour value was 82.

To evaluate tours over a wider variety of problem conditions, a code was written to generate random tours that would

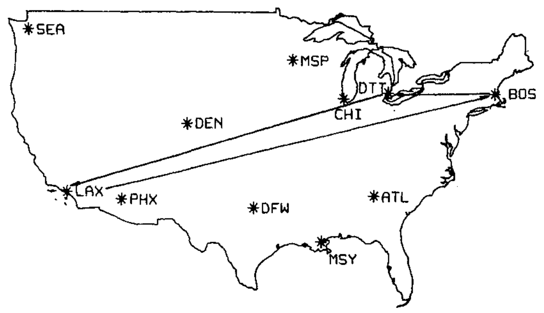

Fig. 1 Initial tour.

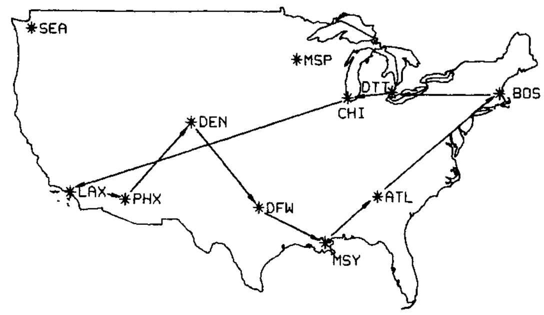

Fig. 2 Optimal tour.

Received June 25, 1987; presented as Paper 87-2332 at the AIAA Guidance, Navigation, and Control Conference, Monterey, CA, Aug. 17–19, 1987; revision received Jan, 11, 1988. Copyright © American Institute of Aeronautics and Astronautics, Inc., 1987. All rights reserved.

*Program Manager, Strategic Navy Division.

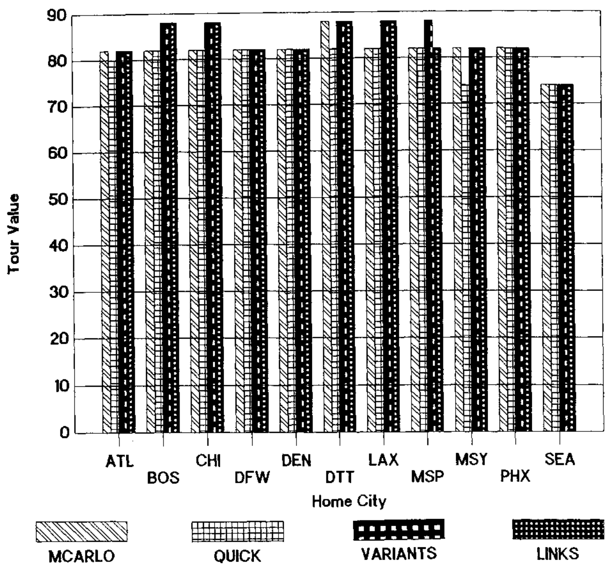

Fig. 3 Comparison of results, sample problem.

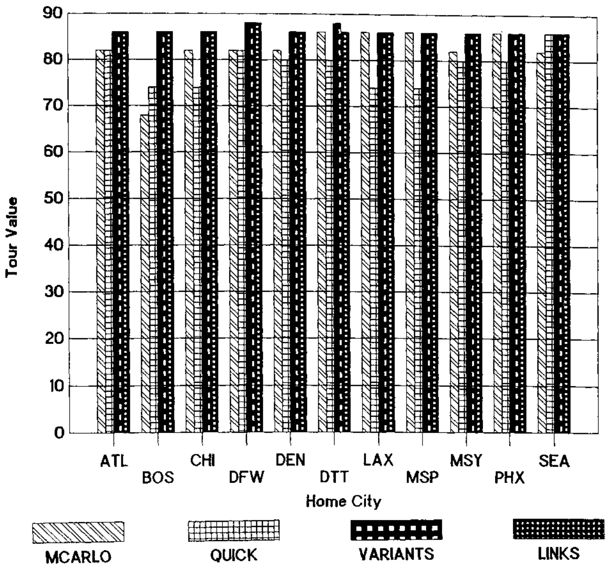

Fig. 4 Comparison of results, random fares.

satisfy the problem constraints and execute very quickly. The purpose was to allow the computer to generate several thousands of tours. It was assumed that the best of these Monte Carlo tours would at least be an excellent tour, although not necessarily an optimal tour. The Monte Carlo code MCARLO found the optimal solution of 88 for the baseline case (when run over a weekend) and a second-best solution of 82 after only a few minutes. For the baseline tour with a home city of Denver, an hour of Monte Carlo execution gave a solution of 82, which is also the optimal tour value.

The MCARLO code starts with a "tour" from the home city that returns to the home city. A city that is not on the tour is randomly selected and added to the tour, just before the return to the home city, and the budget constraint is checked. This is repeated until the budget constraint is exceeded. The value of the last valid solution and its expected cost are used to update the best solution found so far, as appropriate. This method will evaluate about 12 tours per second. MCARLO was exercised for one hour or more with each of the 11 cities as the home city. This was done for the baseline inputs and for a matrix of random air fares with the same mean and standard deviation. This provided a set of 22 problems with excellent solutions to measure algorithms against.

A difficult and time-consuming part of the tour evaluation is checking the budget constraint. The normal approximation to the binomial distribution of the air fares was tried and found to be inaccurate. The solution adopted is to evaluate quickly the maximum cost of a tour by adding up all first-class legs. If this maximum cost is below the budget, then the tour is acceptable. If not, then the entire probability and cost tree must be generated, where each leg of the tour causes the trees to be expanded by a factor of two, corresponding to the coach and first-class fares.

A comparison of the MCARLO results with the initial algorithm (QUICK), an intermediate code (VARIANTS), and the final solution (LINKS) is shown in Figs. 3 and 4 for the sample problem and random fares, respectively. This comparison will be discussed.

Quick Algorithm

To become familiar with the traveling salesman problem, two readily available survey papers were reviewed.[2,3] Both papers described solutions that started with an initial tour and gradually improved the tour by inserting a well-chosen city into the tour. This suggested the following algorithm:

1) Construct an initial tour consisting of Home-LAX-BOS-Home.

2) Consider the set of tours formed by placing each city not already on the tour into each possible interior leg of the tour.

3) From these tours, select the best tour, i.e., the city and leg that produces the tour with the maximum ratio of value to cost.

4) Insert the selected city into the tour, if this can be done within budget. Repeat steps 2–4. If the city cannot be added within budget, delete the selected city and leg from consideration and repeat steps 3 and 4 with the next best selection.

5) Continue the algorithm until no more cities can be added within budget.

The initial tour, depicted in Fig. 1, is a key concept that provides a convenient means of satisfying constraints of beginning and ending the tour at the home city and visiting LAX before BOS. Logic was added to check for LAX or BOS being the home city, and the cost of the initial tour was checked. If the cost is excessive, the initial tour is reset to Home-Home and LAX is excluded from consideration.

To select the best expansion for the tour, the marginal gain was maximized. That is, the new tour that produced the most value per dollar was selected. Since total costs are larger than total values, this was implemented as minimizing the ratio of the sum of the fares to the sum of the values, which allowed much faster integer arithmetic to be used. The budget constraint was handled as described above for MCARLO.

QUICK executes in only 6 s and offers very good performance. For the baseline problem, QUICK finds a tour with a value of 82, compared to the optimal value of 88. As seen in Figs. 3 and 4, QUICK performs as well as MCARLO in many cases.

Variants Refinement

As noted, QUICK produced a value of 82 for the baseline case, whereas 88 was optimal. The two tours were very similar, and it was observed that the QUICK algorithm would converge to the 88 tour if the initial tour were slightly varied; i.e., QUICK was converging to a local optimum. Therefore, a systematic way to vary the initial tours for QUICK to evaluate was needed.

The method selected, denoted VARIANTS, is to set up the initial tour as before and find the longest, i.e., the most expensive, leg. Each city not on the initial tour is inserted at this leg, one at a time. The QUICK algorithm is then applied to each of these variants of the initial tour, and the solution is the best tour generated.

Execution time for VARIANTS increased to 57 s, and performance significantly improved, as seen in Figs. 3 and 4. Solutions found are equal to or better than Monte Carlo solutions. An 88 is found for the baseline case.

Table 1 Sample problem cheaper fares

From city	To city	Via
BOS	PHX	LAX
BOS	SEA	LAX
DTT	LAX	CHI
LAX	BOS	MSP
LAX	CHI	MSP

Table 2 Run time comparison

	QUICK		LINKS	
n	Tour Value	Time, s	Tour Value	Time, s
11	88	1.5	94	5.8
22	120	5.6	124	52.3
33	128	14.0	142	205.8
44	150	27.2	150	494.9
55	132	44.7	136	878.9

Refinement Links

Both QUICK and VARIANTS expand the tour by considering only cities that are not already on the tour. It is easy to envision a situation where the fare from, say, DTT to LAX might be cheaper through CHI. This situation would not be found by either QUICK or VARIANTS if CHI were already on the tour.

The fare matrix[4] of the sample problem was examined to identify links that should be inserted between cities to reduce travel costs. For each pair of cities i and j, the method looks for a stopover city k, so that $c_{ik} + c_{kj} < c_{ij}$, where c_{ij} is the cost of travel from city i to j, including air fare and miscellaneous costs. Five links to reduce costs in the baseline problem were found and are listed in Table 1. Ten cheaper fare links were found for the random fare matrix.

VARIANTS was refined to generate the cheaper fare links and use them whenever a city is inserted into a tour. The LINKS code requires 60 s for execution, slightly more than VARIANTS. Performance in the baseline case is the same, at 88. For both the sample problem fares and the random fares, tour values are slightly down in some cases from those found by VARIANTS. However, costs for LINKS were often improved.

Final Code

LINKS was selected as the final code for the contest entry because its performance is very nearly as good as VARIANTS. It offers the potential for excellent solutions in situations where a hub city is used.

Run Time

The LINKS code can solve the 11-city problem in only 60 s of time on the AT&T PC. To evaluate the order of the method, very similar problems were solved for up to 55 cities. The baseline data were used in the expanded problems as much as possible. Cost for the 11-city problem was evaluated by the generation of the full cost and probability trees, which required two trees of size 2^n, where n is the number of links in the tour. For n larger than 11, tree size quickly becomes unmanageable. In that case, the cost constraint can be handled by adapting the normal approximation to the binomial distribution or by expressing the cost constraint as a limit on the expected value of the cost. In these runs, the expected cost was constrained to \$3,000. The fare matrix was duplicated as often as necessary, with the zero elements replaced by the average of neighboring fares in the matrix. In addition, all fare values, miscellaneous costs, and the budget of \$3,000 were divided by $n/11$, so that integer arithmetic could still be used for speed.

Evaluations were made for both QUICK and LINKS versions of the codes. Results are given in Table 2. Note that the run time for LINKS for the 11-city case has been reduced from 60 s to 5.8 s. The only change for this case is cost evaluation. The figure also shows that QUICK results are within 10% or less of LINKS. QUICK run times increase according to the square of n, while LINKS run times vary by the third power. Memory requirements are modest in both cases. For LINKS, 55 cities require about 56K of memory, with an increase of 1K per 11 cities added.

Conclusions

In the course of developing this set of solutions to the Artificial Intelligence Design Challenge, some interesting conclusions have been observed. The initial approach of developing a Monte Carlo code was an illuminating way to develop good understanding and, with sufficient time and a reasonably efficient code, the Monte Carlo approach will give benchmark results for the evaluation of algorithms over a wide set of problems. Exhaustion of all possible tours was possible for a limited set, for fine tuning and to settle the issue of true optimality for those problems. The constraints of starting and ending at the home city and visiting Boston after Los Angeles were easily handled by careful selection of the initial tour. Incremental expansion of this initial tour by selecting the next city and position on the tour by simple marginal gains was fast and produced very good tours. For larger problems, this simple method was found to require run times that increased according to the square of n. Because this method converges to a local optimum, it was improved by varying the initial tour. The improved method is of the order n^3 and obtains results that were only about 10% or less better than the simple method, at the cost of a substantial increase in execution time. Finally, for this particular problem, most of the execution time was spent in evaluating the cost constraint. An alternative formulation of this constraint to bound the expected cost, rather than to achieve a specified probability of exceeding the budget, reduces execution time by an order of magnitude.

References

[1]Deutsch, O. L., "Artificial Intelligence Design Challenge at the 1987 Guidance, Navigation, and Control Conference," *Journal of Guidance, Control, and Dynamics*, Vol. 2, Sept./Oct. 1986, p. 513.

[2]Karp, R. M., "Combinatorics, Complexity, and Randomness," *Communications of the ACM*, Vol. 29, Feb. 1986.

[3]Lewis, H. R. and Papadimitriou, C. H., "The Efficiency of Algorithms," *Scientific American*, Jan. 1978.

[4]Pollack, M. and Wiebenson, W., "Solutions of the Shortest-Route Problem—a Review," *Operations Research*, Vol. 8, March–April 1960.