Expert Systems Approach for Generalized Traveling Salesman Problem

Yuval Lirov*
Washington University, St. Louis, Missouri

The expert systems approach for solving multiobjective control problems posed in the environment of high complexity, uncertainty, and global and local constraints is exemplified via the solution of a generalized traveling salesman problem.

Introduction

CURRENT control problems are related to hybrid, time-variant, and multiobjective control systems. The distinguishing characteristics of these problems are that they are usually NP-hard (nonlinear programming-hard) and that the descriptions of the system and objective functions are specified as being time-variant, not linear, and not differentiable. On the other hand, the solution is usually required in real time. Therefore, on these problems, the classical schemes, such as the gradient method, may not be applied.

Our ambition is to obtain a methodology to design such control systems that could, in real time, identify the environment and the plant, create appropriate control goals, and design appropriate controllers and controls. Such systems will have to operate with a variety of variables of different origin at the same time, thus prohibiting utilization of classical algorithms, which are based on the assumption of continuity of change. In the absence of mathematically sound ground, symbolic processing and heuristic programming will be required at every phase.

We have proposed a methodology to handle such problems.[1] The essence of our methodology is to introduce a varying formal model to serve as the basis for a description of the system and its goals. It allows for formulation of appropriate decision and learning mechanisms to be applied at the system identification, objective definition, and control phases. The research into the construction, interaction, and complexity of these mechanisms employs a variety of methods that belong to systems theory, computational complexity, and artificial intelligence.

Following our approach, the classical open/closed loop control paradigm is replaced by the following control procedure, which we call a semantic controller (Fig. 1). It consists of a set of plants being identified, a set of corresponding controllers being designed, and a single automatic designer taking care of the three tasks of identification, goal selection, and controller design.

Each of the three tasks is achieved via an appropriate correlator that has a generic expert system[2] structure (Fig. 2). These expert systems collaborate to achieve the common control goal of the plant survival in the changing environment, yet each of them performs its own search using separate data nd knowledge bases, and an inference engine.

In this framework, the problem posed for the AIAA Artificial Intelligence Design Challenge[3] can be viewed as the input to the goal selector obtained from the identifier. In the rest of this paper, we discuss its efficient implementation.

Received June 19, 1987; presented as Paper 87-2328 at the AIAA Guidance, Navigation, and Control Conference, Monterey, CA, Aug. 17-19, 1987; revision received Nov. 2, 1987.

*Research Assistant; currently, MTS, AT&T Bell Laboratories, Holmdel, NJ.

System Overview

At the most general level, the system employs the divide-and-conquer and generate-and-test approaches: The tours are incrementally generated and tested until the best solution is obtained. The data base is used to store the initial data, the best solution obtained so far, and the learned data. The knowledge base consists of the rules for the next city inclusion in the current tour, the tour evaluation, and its transformation. The inference engine searches the space of feasible tours.

The search space consists of subsets of the set of all possible tours. Every subset has two measures associated with it: achievable value and cost. The achievable value of the subset is defined as the upper bound that is less or equal to the highest possible value that could be collected by any tour traversing the members of the subset. The cost of the subset is defined as the minimal total expenditure required to visit every city in the subset.

The search space is structured in a tree of nodes corresponding to the aforementioned subsets. The data corresponding to every node in the search tree consist of the shortest tour, the ordered list of next candidate cities, and the two aforementioned measures. The search problem is solved by developing the search tree until the optimal solution is found.

The search process is specified at three different levels: 1) next node selection, 2) next city selection, and 3) cost (minimal tour) computation.

The next node selection is performed using a heuristic node evaluation function, in which the input values for the heuristic evaluation function are updated during the learning process at the lower level, that is, the cost (minimal tour) computation. In addition, the usual branch-and-bound and dynamic programming techniques are utilized to facilitate an efficeint search of the solution tree. This constitutes a classic A^* search procedure.

The cost (minimal tour) computation is itself an NP-complete problem. Our program computes an overestimate of the cost using the simulated annealing approach.[4] The main advantage of this approach is that the annealing process terminates fast when the initial tour to be "annealed" is already close to the minimal one. In this sense, the information learned at the previous stages of the compuation is not lost but fully exploited at the later stages.

Heuristic Search

The next node selection is performed according to the following rule: next_node = arg max {h(node)}, where

$$h(n) \triangleq v(n)^2 + \alpha(n) \cdot v(n)$$

$$\alpha(n) \triangleq [\text{budget} - \text{cost}\ (n)]/\text{budget}$$

$v(n)$ – upper bound of the tour value (achievable value)

Lemma: If $v(n) \geq v(m)$, then $h(n) \geq h(m)$ for all n and m.

Proof: Note that $0 \leq \alpha(n) < 1$ for all n. The following three cases can occur:

1) $v(n) = v(m)$, $\alpha(n) \leq \alpha(m)$

2) $v(n) \geq v(m) + 1$, $\alpha(n) \geq \alpha(m)$

3) $v(n) \geq v(m) + 1$, $\alpha(n) < \alpha(m)$

Cases 1 and 2 are trivial. As for case 3, note that

$$\frac{v(n)}{v(m)} \geq 1 \text{ and}$$

$$\frac{v(m)+\alpha(m)}{v(n)+\alpha(n)} \leq \frac{v(m)+1}{v(n)} \leq 1$$

since $0 \leq \alpha(n) < 1$ for all n.

Therefore,

$$\frac{v(n)}{v(m)} \geq \frac{v(m)+\alpha(m)}{v(n)+\alpha(n)}$$

or

$$v(n)^2 + v(n)\alpha(n) \geq v(m)^2 + v(m)\alpha(m)$$

So

$$h(n) \geq h(m)$$

Corollary 1: Let n^* be the node corresponding to the value of the optimal tour, that is, let

$$v^* = v(n^*) = \max v(n)$$

where n runs over all possible tours. Then,

$$h(n^*) = \max h(n)$$

Now let the root of the search tree have the value v(root) equal to the sum of the values of all available cities (that may or may not eventually participate in the optimal tour). Also let

$$v(\text{next\_node}) = \begin{cases} v \text{ (node), whenever an available city from the node is included in the next\_node and} \\ v(\text{node}) - \text{value(city), whenever an available city from the node is excluded from the next\_node} \end{cases}$$

Then,

$$v(n) \geq v(n^*) \quad \text{and} \quad h(n) \geq h(n^*)$$

for all partially explored (i.e., having nonempty list of available cities) nodes. The choice of the foregoing heuristic allows for ordering the nodes of the search tree in the descending order of value and cost.

Theorem: The heuristic h is an admissible heuristic.

Proof: Trivial.

Corollary 2: The following algorithm performs A^* search for the best tour:

start from root including only the home city;
repeat
select next city;
compute next tour including next city;
if constraints are satisfied, then begin
update next city selection list;
check optimality
end;
compute next tour excluding next city;
if v(next tour) > maximal value, then check optimality;
remove node = arg max $\{h(\text{node})\}$
until v(node) ≤ maximal value

The procedure "check optimality" verifies if all the cities have been considered. In such a case, the optimal tour and maximal value may be updated; otherwise, the heuristic node value $h(n)$ is computed and the node inserted into a priority queue.

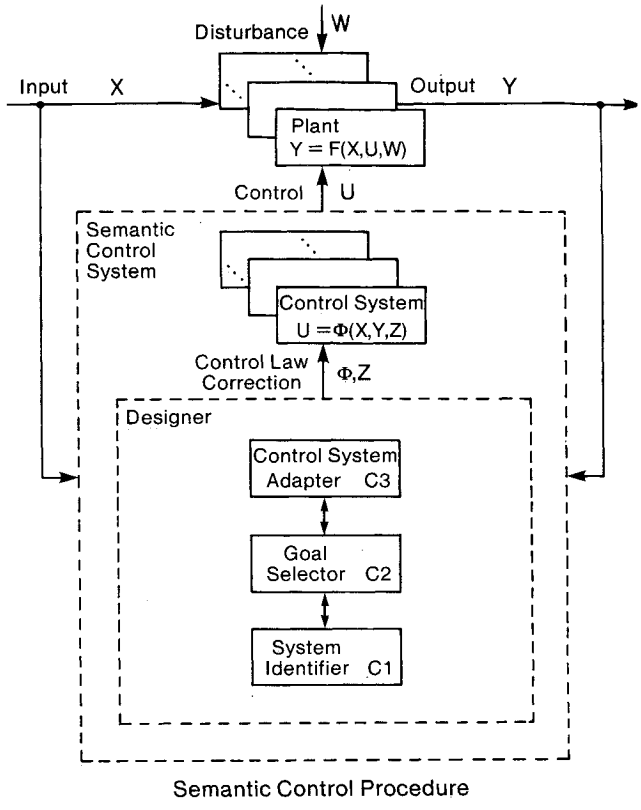

Fig. 1 Semantic control procedure.

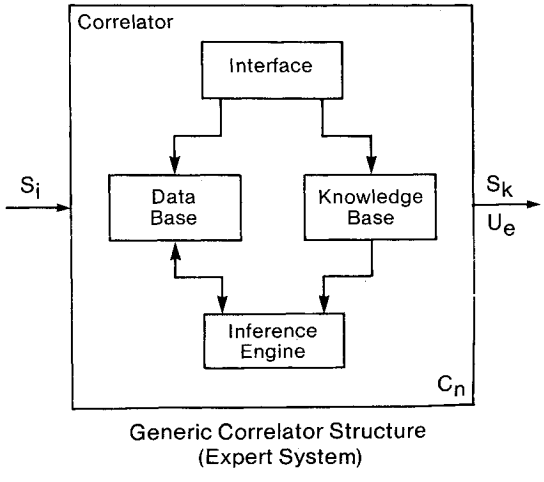

Fig. 2 Export system.

Heuristic Next City Selection

The next city selection is performed according to the following rule: next_city = arg min$H$(city), over all available cities, where

$$H(j) \triangleq [\min_i d(i,j) + \min_k d(j,k)]/\text{value}(j)$$

and i,k belong to the set of already included cities in the tour. This formulation allows for the recursive computation of $H(j)$ using the following definition:

$$\text{min\_from}_{t+1}(j) = \min[\text{min\_from}_t(j), d(j,k)]$$

where t represents the inclusion stage and k is the number indicating the last included city. Similarly, min_to(j) can be computed. Then,

$$H(j) = [\text{min\_to}(j) + \text{min\_from}(j)]/\text{value}(j)$$

Heuristic Minimal Tour Computation

We employ the method of simulated annealing[4] to compute the minimal tour of the given set of cities. The general scheme of always taking a downhill step while sometimes taking an uphill step in the series of successive efforts to minimize the energy of a system is known as the metropolis algorithm. The following elements must be provided in order to use such an algorithm: a description of possible system configurations, a generator of random changes in the configuration, an objective function E whose minimization is the goal of the procedure, a control parameter T, and an annealing schedule that defines the changes of the control parameter.

Configuration

The cities are numbered $i = 1,\ldots,N$. A configuration is a permutation of these numbers interpreted as the order in which the cities are visited.

Rearrangements

Following Lin,[5] two types of moves are allowed: *reverse*, when a section of the path is replaced by the same cities running in the opposite order, and *transport*, when a section of the path is removed and then inserted in between two cities on another, randomly chosen part of the path.

Objective Function

Our objective function is the cost of the path.

Annealing Schedule

The initial value of T is found experimentally to be budget/N, where N is the number of cities in the tour. The step size of T is chosen to be T/N. The simulation is performed for every value of T from some $f(N)$ number of reconfigurations, or for $g(N)$ number of successful reconfigurations, whichever comes first. We found experimentally that it is best to have

$$f(N) \triangleq N^2, \qquad g(N) \triangleq N$$

We stop the annealing procedure when the efforts to reduce the total cost becomes unsuccessful. Note that this stopping rule implies that the better the initial path, the earlier we will return. It is at this point that the information learned and accumulated at the previous steps is exploited to its greatest extent.

Constraint Satisfaction

Two kinds of constraints are introduced in our problem: the global probabilistic constraint and local Los Angeles–Boston constraint. We deal with these constraints separately at two different levels.

Los Angeles–Boston Constraint

The local Los Angeles–Boston constraint satisfaction is dealt with by the knowledge base. We postpone the description of the knowledge base until the next section.

Probabilistic Constraint

The probabiltiy of budget excess is computed after every invoke of anneal according to the following recursive formula:

$$P(Y_n + X_{n+1} > b) = P(Y_n > b - x_{n+1})q + P(Y_n > b - x'_{n+1})p \qquad \text{for } n \geq 2$$

$$P(X_1 > b) = \begin{cases} 1, & x_1 > b \\ p, & x'_1 > b \\ 0, & \text{otherwise} \end{cases}$$

where

X_i = a random variable taking the values x_i and $x'_i = (1+\alpha)x_i$ with probabilities q and p, $(q+p=1)$, respectively
x_i = cost of the ith link in the path
α = first-class premium
p = probability of the premium

$$Y_n \triangleq \sum_{i=1}^{n} X_i$$

b = budget limit

The state of the search space is included in the search tree only if it satisfies the global probabilistic constraint. Note that such an explicit computation of the probability value is extremely time-consuming and should be replaced by more clever techniques suggested in other papers in this issue. The local Los Angeles–Boston constraint, once initiated, is propagated all the way to the leaves that belong to the subtree rooted at the local constraint initiation node.

Knowledge Base

The knowledge base contains the expertise to handle different situations in order to support the computations of the inference engine. In particular, its knowledge pertains to

1) Computation of the optimal order and the associated cost of any given subset of cities.

2) Computation of the still achievable value of any given partially constructed tour.

3) Discovery and treatment of the "hidden" cities (a city C is called a hidden city if there exists a pair of adjacent cities A and B in the given tour such that

$$\text{fare}(A,B) \geq \text{fare}(A,C) + \text{fare}(C,B)$$

recall that neither the triangle inequality nor symmetry conditions are satisfied).

4) Ordering of the cities to be considered later for inclusion or exclusion in any given tour.

5) Inclusion of the next most promising city in the partially constructed tour.

The knowledge base is organized in the rule-based manner and divided into five subsections: Optimal-order rules, value rules, hidden city discovery rules, next city priority rules, and inclusion rules. We list some of the rules in the sequel.

Optimal-Order Rules (OOR)

OOR1: If total cities ≤ 5
then apply brute force order

OOR2: if $5 <$ total cities ≤ 8
and no Los Angeles–Boston constraints
then apply Little's order[6]

Table 1 Output of 10 sample runs of the expert system

Budget	Value	Expected cost	Excess probability	Time, s
1550	46	1433.63	0.03	2.86
	DTT CHI DFW MSY ALT			
1800	52	1680.00	0.01	8.85
	DTT CHI MSP DFW MSY ALT			
2050	62	1814.40	0.01	18.45
	DTT CHI LAX MSP BOS			
2300	68	2094.40	0.01	15.00
	DTT CHI LAX PHX MSP BOS			
2550	74	2273.60	0.00	30.70
	DTT CHI LAX PHX DEN MSP BOS			
2800	82	2632.00	0.02	32.84
	DTT CHI MSP LAX PHX DFW ATL BOS			
3050	88	2923.20	0.03	31.15
	DTT CHI MSP LAX DEN DFW MSY ALT BOS			
3300	94	3091.20	0.00	13.84
	DTT CHI MSP LAX DEN DFW MSY ALT BOS			
3550	94	3248.00	0.00	11.65
	DTT CHI LAX PHX DEN DFW MSY ALT MSP BOS			
3800	100	3606.40	0.01	14.39
	DTT CHI MSP DEN SEA LAX PHX DFW MSY ATL BOS			

OOR3: if total cities > 8
or Los Angeles–Boston constraint is on
then apply anneal order

Value Rules ($\bar{VR}$)

VR1: if a city is included
then achievable value = achievable value

VR2: if Boston is excluded
then achievable value = achievable value − value (Boston) − value (Los Angeles) if a city is excluded

VR3: and city ≠ Boston
then achievable value = achievable value − value (city)

VR4: if Boston is discovered
and Boston was excluded
and Los Angeles belongs to path
and Los Angeles before Boston
then achievable value = achievable value + value (Boston) + value (Los Angeles)

VR5: if Los Angeles is discovered
and Los Angeles was excluded
and Boston belongs to path
and Los Angeles before Boston
then achievable value = achievable value + value (Los Angeles)

VR6: if a city is discovered
and city was excluded
and city ≠ Boston
and city ≠ Los Angeles
then achievable value = achievable value + value (city)

Hidden City Discovery Rule (*HCDR*)

This rule uses the adjacency matrix that is constructed at the preprocessing stage using the Floyd-Marshall[7] algorithm.

HCDR: if adj$[C, j] \neq 0$ (adj stands for adjacency matrix)
and C was excluded
then C is discovered

Next City Priority Rules (*NCPR*)

NCPR 1: if Boston belongs to path
then increase priority to Los Angeles

NCPR 2: if Boston is excluded
then exclude Los Angeles

NCPR 3: if Boston does not belong to path
and Boston is not excluded
then decrease priority of Los Angeles

NCPR 4: if Los Angeles belongs to path
then increase priority of Boston

Inclusion Rules (*IR*)

IR 1: if selected city is Los Angeles
and Boston belongs to path
then include Los Angeles before Boston

IR 2: if selected city is Boston
and Los Angeles belongs to path
then include Boston after Los Angeles

IR 3: if IR 1 not "fired"
and IR 2 not "fired"
then include selected city anywhere

Implementation

The system is written in Turbo-Pascal and takes about 1000 lines of code. The main data structures are the following: a tour record consisting of the current best path, priority queue of the available cities, cost, and achievable value, and a search tree (priority queue) of the tour records.

Both priority queues are implemented as heaps.[8] The advantages of this implementation are threefold: A heap is guaranteed to be balanced, remove and insert operations are of 0 (log$N$) time complexity and, when represented as an array, in particular, the largest key is always in the first position in the array.

Computational Results

Table 1 represents the values, cost, and the average computation times of 10 sample runs for the data published in Ref. 3, and 10 different budget limits on the IBM-PC/AT computed with an 80287 coprocessor.

Conclusions

The expert systems approach for solving multiobjective control problems posed in the environment of high complexity, uncertainty, and global and local constraints is exemplified via the solution for a generalized traveling salesman problem.

The heuristic next node evaluation function is defined and proved to be admissible under the assumption of exact minimal tour cost availability. The learning mechanisms are defined at the levels of next subset construction and best tour computation.

The advantage of the expert systems approach is mainly in the case of implementing various local and global constraints. Also, the expert systems functional subdivision allows for considering the search algorithm (inference engine) separately from the constraints (part of the knowledge base), thus additionally contributing to the ease of software implementation.

The largest average experimental running time on the IBM PC/AT solving a sample problem was less than 40 s.

Acknowledgment

The author is grateful to B.K. Ghosh and E.Y. Rodin for many stimulating discussions during the development of this project.

References

[1]Lirov, Y., "Artificial Intelligence Methods in Decision and Control Systems," PhD. Dissertation, Washington Univ., St. Louis, MO, 1987.

[2]Winston P.H., *Artificial Intelligence,* 2nd ed., Addison-Wesley, Reading, MA, 1984.

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

[4]Kirkpatrick, S., Gellatt, C.S., and Vecchi, M.P., "Optimization by Simulated Annealing, *Science,* Vol. 220, May 1983, pp. 671-680.

[5]Lin, S., "Computer Solutions of the Traveling-Salesman Problem," *Bell Systems Technical Journal,* Vol. 44, 1965, pp. 2245-2269.

[6]Little, J.S.C., et al., "An Algorithm for the Traveling Problem," *Operations Research,* Vol. 11, 1963, pp. 972-989.

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

[8]Brown, M.R., "Implementation and Analysis of Binomial Queue Algorithms," *SIAM Journal of Computing,* Vol. 7, Aug. 1978.