VI. Conclusion

The rate at which learning algorithms converge has been a limiting factor in their implementation. Discretizing provides a general method of improving the performance of VSSA. Discretizing VSSA restricts the probability of choosing an action to a finite number of values. Discretizing reduces computation time by replacing floating-point multiplication with integer addition. It also reduces the number of iterations needed for an algorithm to converge. Finally, discretizing eliminates the need for precise random number generators.

The CPA is among the quickest stochastic learning automata known to date. As such, it represents a natural candidate for discretization. Fortunately, ϵ-optimality is preserved when the CPA is discretized. As well, in some difficult two-action environments the DPA requires only about 50% of the number of iterations required for its continuous counter part. It required only 69% of the iterations required by the CPA when learning in a ten-action environment. Indeed, the DPA is probably the fastest nonestimator or pursuit automaton reported in the literature to date.

We are currently working on discretizing the more general family of estimator algorithms introduced by Thathachar and Sastry [27]. We believe that the conclusions made in the comparison between the continuous and the discretized versions of the PA generalize to the entire family of estimator algorithms.

Acknowledgment

We are grateful to Mr. Valiveti, Prof. Thathachar and an anonymous referee who provided us with many helpful comments. We are also grateful to Prof. Dixon from Carleton University who brought to our attention an error in the original proof of convergence.

References

[1] S. Baba, S. T. Soeda, and Y. Sawaragi, "An application of stochastic automata to the investment game," *Int. J. Syst. Sci.*, vol. 11, no. 12, pp. 1447–1457, Dec. 1980.

[2] Y. A. Flerov, "Some classes of multi-input automata," *J. Cybern.*, vol. 2, pp. 112–122, 1972.

[3] D. L. Isaacson and R. W. Madson, *Markov Chains: Theory and Applications*. New York: Wiley, 1976.

[4] S. Karlin and H. M. Taylor, *A First Course on Stochastic Processes*, second ed. New York: Academic Press, 1974.,

[5] S. Lakshmivarahan, *Learning Algorithms Theory and Applications*. New York: Springer-Verlag, 1981.

[6] ____, "ϵ-optimal learning algorithms—Non-absorbing barrier type," Tech. Rep. EECS 7901, Feb. 1979, School of Elec. Eng. and Computing Sci., Univ. Oklahoma, Norman, OK.

[7] ____, "Two person decentralized team with incomplete information," *Appl. Math. and Computation*, vol. 8, pp. 51–78, 1981.

[8] S. Lakshmivarahan and M. A. L. Thathachar, "Absolutely expedient algorithms for stochastic automata," *IEEE Trans. Syst. Man Cybern.*, vol. SMC-3, pp. 281–286, 1973.

[9] M. R. Meybodi, "Learning automata and its application to priority assignment in a queueing system with unknown characteristics," Ph.D. Thesis, School of Elec. Eng. and Computing Sci., Univ. Oklahoma, Norman, OK.

[10] K. S. Narendra and M. A. L. Thathachar, *Learning Automata*. Englewood cliffs, NJ: Prentice-Hall, 1989.

[11] ____, "Learning automata—A Survey," *IEEE Trans. Syst. Man Cybern.*, vol. SMC-4, pp. 323–334, 1974.

[12] ____, "On the behavior of a learning automaton in a changing environment with routing applications," *IEEE Trans. Syst. Man Cybern.*, vol. SMC-10, pp. 262–269, 1980.

[13] K. S. Narendra, E. Wright, and L. G. Mason, "Applications of learning automata to telephone traffic routing," *IEEE Trans. Syst. Man Cybern.*, vol. SMC-7, pp. 785–792, 1977.

[14] K. S. Narendra and S. Lakshmivarahan, "Learning automata: A critique," *J. Cybern. Inform. Sci.*, vol. 1, pp. 53–66, 1987.

[15] B. J. Oommen and E. R. Hansen, "The asymptotic optimality of discretized linear reward-inaction learning automata," *IEEE Trans. Syst. Man Cybern.*, pp. 542–545, May/June 1984.

[16] B. J. Oommen and J. P. R. Christensen, "Epsilon-optimal discretized linear reward-penalty learning automata," *IEEE Trans. Syst. Man Cybern.*, vol. SMC-18, pp. 451–458, May/June 1988.

[17] B. J. Oommen and M. A. L. Thathachar, "Multiaction learning automata possessing ergodicity of the mean," *Inform. Sci.*, vol. 35, no. 3, pp. 183–198, June 1985.

[18] B. J. Oommen, "Ergodic learning automata capable of incorporating *a priori* information," *IEEE Trans. Syst. Man Cybern.*, vol. SMC-17, pp. 717–723, July/Aug. 1987.

[19] ____, "Absorbing and ergodic discretized two-action learning automata," *IEEE Trans. Syst. Man Cybern.*, vol. SMC-16, pp. 282–296, 1986.

[20] ____, "A learning automaton solution to the stochastic minimum spanning circle problem," *IEEE Trans. Syst. Man Cybern.*, pp. 598–603, July/Aug. 1986.

[21] B. J. Oommen and D. C. Y. Ma, "Deterministic learning automata solutions to the equi-partitioning problem," *IEEE Trans. Comput.*, vol. 37, pp. 2–14, Jan. 1988.

[22] ____, "Fast object partitioning using stochastic learning automata," in *Proc. 1987 Int. Conf. Research Development in Inform. Retrieval*, New Orleans, LA, June 1987.

[23] R. Ramesh, "Learning automata in pattern classification," M.E. thesis, Indian Institute of Science, Bangalore, India, 1983.

[24] P. S. Sastry, "Systems of learning automata: Estimator algorithms applications," Ph.D. thesis, Dept. Elec. Eng., Indian Institute of Science, Bangalore, India, June 1985.

[25] M. A. L. Thathachar and B. J. Oommen, "Discretized reward-inaction learning automata," *J. Cybern. Inform. Sci.*, pp. 24–29, Spring 1979.

[26] M. A. L. Thathachar and P. S. Sastry, "A new approach to designing reinforcement schemes for learning automata," in *Proc. IEEE Int. Conf. Cybern. Syst.*, Bombay, India, Jan. 1984.

[27] ____, "A class of rapidly converging algorithms for learning automata," *IEEE Trans. Syst. Man Cybern.*, vol. SMC-15, pp. 168–175, Jan. 1985.

[28] ____, "Estimator algorithms for learning automata," *Proc. Platinum Jubilee Conf. on Syst. Signal Processing*, Dept. Elec. Eng., Indian Institute of Science, Bangalore, India, Dec. 1986.

[29] M. L. Tsetlin, "On the behavior of finite automata in random media," *Automat. Telemekh.* (USSR), vol. 22, pp. 1345–1354, Oct. 1961.

[30] ____, *Automaton Theory and the Modelling of Biological Systems*. New York: Academic, 1973.

[31] Y. Z. Tsypkin and A. S. Poznyak, "Finite learning automata," *Eng. Cybern.*, vol. 10, pp. 478–490, 1972.

[32] V. I. Varshavskii and I. P. Vorontsova, "On the behavior of stochastic automata with variable structure," *Automata Telemekh.* (USSR), vol. 24, pp. 327–333, 1963.

[33] S. Mukhopadhyay and M. A. L. Thathachar, "Associative learning of boolean functions," *IEEE Trans. Syst. Man Cybern.*, vol. SMC-19, pp. 1008–1015, 1989.

[34] J. K. Lanctôt, "Discrete estimator algorithms: A mathematical model of machine learning," M.Sc. thesis, Carleton Univ., Ottawa, ON, Canada, Fall 1989.

Task-Level Tour Plan Generation for Mobile Robots

J. BORENSTEIN, MEMBER, IEEE, AND Y. KOREN, SENIOR MEMBER, IEEE

***Abstract*—A tour plan generator (TPG) specifically adapted for mobile robots is described. The TPG computes the itinerary of a tour passing through a number of locations. While solutions to similar problems exist (e.g., the well-known traveling salesman problem), special constraints apply to mobile robot applications. One typical constraint is that a single-armed mobile robot is usually unable to carry more than one object at a time in its gripper. This constraint requires the TPG to generate a tour in which each pick-up location is visited immediately prior to the corresponding drop-off location. A model is introduced that allows us to reduce this problem to a special case of the traveling**

Manuscript received June 13, 1989; revised March 6, 1990. This work was supported in part by the Department of Energy, under Grant DE-FG02-86NE37967.

The authors are with the Department of Mechanical Engineering and Applied Mechanics, University of Michigan, Ann Arbor, MI 48109.

IEEE Log Number 9035711.

salesman problem with asymmetric cost matrix and non-Euclidean distances. Another constraint in many mobile robots is the need to periodically visit a home location (e.g., for the purpose of battery charging or position updating). The TPG introduced in this paper automatically creates multiple subtours such that a predefined maximal length for each subtour is not exceeded. The algorithm obtains near-optimal solutions with short computation times by combining different heuristic tour-construction rules into a heuristic team approach. With this method, new heuristic rules can be added in order to improve the accuracy of the algorithm for particular applications. An index of performance has been defined to evaluate the performance of individual members of the heuristic team.

I. Introduction

A task-oriented navigation system for mobile robots is currently under development in the Robotic Systems Division at the University of Michigan. This hierarchically organized system [2] comprises four functional levels of performance, described below.

A. The Hierarchical Navigation System

The task planner (TP) represents the system's highest level (Level IV). Interfacing with the human operator, the TP receives a list of tasks that need to be performed by the robot. The TP then extracts from this list all locations pertinent to robot travel and sends them to the next lowest level.

Level III consists of the tour plan generator (TPG), which plans a sequence of trips among all specified locations. This is not a trivial task, since some locations must be visited in a certain order and some tours may be too long and have to be broken up into smaller ones.

The global path planner (GPP) resides in Level II of the system. The GPP contains a world model that includes information about stationary obstacles (e.g., walls), off-limit zones (e.g., stairs), and recently detected unexpected obstacles. Based on this information, the GPP plans an optimal path between the robot's current location and its destination [1]. The path produced by the GPP is expressed as a linked list of via-points, typically spaced about 1–10 m apart.

At the lowest level (Level I), the local path planner (LPP) drives the mobile robot from one via-point to the next. The main task of the LPP, however, is obstacle avoidance [2], [3], [4]. In our mobile robot system, the LPP uses ultrasonic sensors to detect and avoid obstacles without interrupting the robot's progress toward the designated target.

This paper describes a TPG specifically designed for mobile robot applications. The heart of the TPG is an algorithm that computes an optimal (or near-optimal) tour comprising a start location and any number of intermediate destinations. It is assumed that a tour is closed—that is, the robot returns to its original start location.

B. Tour Plan Generation for Mobile Robots — Required Features

The tour planning problem falls into the class of vehicle scheduling problems, which encompass a variety of transport applications. However, mobile robots represent a specific class of transportation devices and therefore require special considerations, outlined as follows.

1) Distribution: This is the most basic feature of the TPG. A distribution task requires the mobile robot to pick up a supply of items at a central depot and then distribute these items to a number of drop-off points. For this purpose, the TPG must calculate an optimal tour (usually in terms of distance), starting at the depot and passing each specified drop-off point exactly once, before returning to the depot. For distribution tasks, the order in which drop-off points are visited is unimportant and the robot's transport capacity is considered unlimited—that is, it is assumed that the carrying capacity will accommodate all items scheduled for distribution. This basic case is also known as the traveling salesman problem (TSP).

2) Tour Length Constraint: A mobile robot's maximal tour length between visits to its home depot may be limited. Typical reasons for this limitation are the need to recharge batteries or to recalibrate an onboard positioning system (e.g., a gyro). To accommodate this constraint, the TPG must automatically insert home depot visits into the planned tour, effectively dividing the planned tour into one or more subtours. The overall tour length, comprising all such created subtours, is again optimal. While not explicitly addressed in this paper, this feature can easily be modified to deal with load-capacity or travel time limitations.

3) Ordered Pairs: This feature addresses situations in which only one item can be transported at a time, which is characteristic for mobile robots equipped with an arm (since usually only one item can be held in the robot's gripper). Consequently, goal locations should not be treated equally but as ordered pairs, where each source location (where an object is acquired) must be followed immediately by its respective goal location (where the object is released) before the robot can perform the next task.

4) Mixed Tasks: In the TPG, distribution and ordered pair tasks must be handled concurrently. This enables the operator to issue a list of mixed commands, such as (command keywords are printed in upper-case):

> "BRING item_1 FROM place_1 TO place_2 AND DISTRIBUTE item_2 TO place_3 AND DISTRIBUTE item_3 TO place_4 AND [more activities] AND [more activities] PLEASE."

The TPG then computes a sequence of trips that connects all locations such that the overall traveling cost is optimal.

II. The Traveling Salesman Problem

The features listed above are related to a well-known problem in the fields of operations research and artificial intelligence, the TSP. The task is to find an optimal tour through n cities, starting at city 1, then visiting the $n-1$ remaining cities once and only once before returning to the origin. An optimal solution is usually defined in terms of minimum distance or cost.

The problem is of great interest because of its many real-life applications such as fuel oil delivery, newspaper distribution, or the delivery of goods from a central depot to a number of outlets. Various solution to the problem are based on branch-and-bound algorithms that can yield exact (optimal) solutions. Unfortunately, the combinatorial nature of the problem tends to increase computer storage and time requirements dramatically with increasing n. As a matter of fact, the problem belongs to the class of *NP*-hard problems [12], which require an amount of time exponential in n for an exact solution.

As a result, much effort has been put into the development of heuristic algorithms, which, by their very nature, are not necessarily optimal, but require only a polynomial amount of computation time. The trade-off is justified since a "good" solution is acceptable for most practical applications. In fact, Golden and Assad [8] point out that most heuristic solutions are asymptotically optimal; i.e., their relative error goes to zero with large n. For this reason, we will use the term "optimal" in the sense of "near-optimal"—i.e., the best solution the heuristic algorithm can produce. By contrast, when referring to the mathematically exact optimal solution, we will use the term "absolute optimal."

The heuristic algorithms for the TSP fall into three classes:

1) Tour Construction Procedures: The heuristics are applied during the construction of the tour. Various heuristic rules for the selection of the next city to be included in the growing subtour have been suggested in the literature. Examples for this

method are: nearest neighbor, discussed and analyzed in [15]; Clarke and Wright savings, discussed and improved in [7]; nearest, arbitrary, cheapest, or farthest insertion, analyzed and tested in [15]; and convex hull, discussed in [9].

2) Tour Improvement Procedures: A randomly constructed initial tour is improved by exchanging one subset of links in the tour with another subset, until no further improvement is achieved. Examples of this method are the 3-opt method, introduced in [13], and the k-opt method, introduced in [14].

3) Composite Procedures: This is a simple yet effective refinement. First, a tour construction procedure is applied to produce a "good" initial tour. Second, one or even two tour improvement procedures are applied. Several variations of this method have been tested and compared by Golden [9].

A thorough comparison of all the above methods, with many experimental results (concerning accuracy and timing), is given in [9]. A theoretical analysis of some of the tour construction procedures appears in [15].

A. Limitations and Assumptions

The basic form of the TSP includes the following two assumptions:

1) Symmetrical Distance Matrix: This assumption applies to the common case where the distance between location i and location j is the same in both directions. An example of a case that does not produce symmetrical distances is a distance matrix of intra-urban connections with one-way streets. Here the actual distance traveled between two locations is not necessarily the same in both directions.

2) Triangle Inequality: The triangle inequality is fundamental in euclidean geometry and states that the sum of the lengths of any two sides in a triangle is greater than the length of the third side. One example of a case in which the triangle inequality does not necessarily hold are interurban roads in hilly areas.

Not all of the heuristic algorithms above are capable of handling exceptions to the basic TSP. Unfortunately, experimental results [9] relate only to the basic form of the TSP, and so do most other papers on the subject.

B. Vehicle Routing

A related class of problems is known as the vehicle routing problem (VRP). In the VRP, a set of routes must be designed, again, at minimal total cost. However, now each route leaves from and eventually returns to a depot or home base, while satisfying capacity constraints and meeting customer requirements. The problem can be seen as an (at times much more demanding) extension of the standard TSP, with some or all of the following requirements and restrictions added [8]:

- An optimal tour between one central depot and many customers must be found for many vehicles traveling concurrently;
- Vehicles have maximum capacities;
- Vehicles have maximum travel time or time window constraints;
- Multiple depots, multiple capacities, multiple demands, etc., may exist.

Recent research introduces branch-and-bound based methods that solve VRP's with different constraints. For example, Gavish and Srikanth [6] solve large problems with up to 500 locations and multiple subtours, and Kolen et al. [11] solve problems with additional time window constraints, but with much fewer locations ($n = 15$). This and other evidence in the literature indicate that computation time rises dramatically as more constraints are addressed.

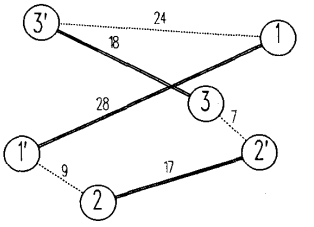

Fig. 1. The triangle inequality does not hold when using the ordered pair model.

For this reason, many researchers [5], [16] base their approach on reducing the complexity of the VRP (by means of some heuristics) to the basic TSP, which may then be solved by any of the heuristic methods discussed previously. Clearly, this approach yields only an approximate solution.

III. A Tour Plan Generator for Mobile Robots

We have developed two complementary models for mobile robot applications that allow us to implement our TPG as a special case of the TSP.

Model I: The foremost problem in tour planning for mobile robot applications stems from the fact that single-arm mobile robots can usually carry only one item at a time. If the robot is required to transport several objects from source locations to goal locations (one at a time) then a source must always be visited immediately prior to visiting a goal—a constraint conventional TSP algorithms cannot handle. Nevertheless, this problem is TSP-related and is known in the literature as the stacker crane problem [12].

As a solution to this problem, we treat each ordered pair (source and goal) as one location in the TSP. This concept can be visualized by picturing the vehicle entering a location i at source(i) and emerging from that location at goal(i') (see Fig. 1). The $n \times n$ cost matrix $\boldsymbol{C}$ comprises of elements $c_{i',j}$, which are the distances between goal(i') and source(j). As seen in Fig. 1, the distance between goal(i') and source(j) differs from the distance between source(i) and goal(j'), and therefore $\boldsymbol{C}$ is asymmetric. Please note that for this model, the internal distance between source(i) and goal(i') may be considered as zero. However, since internal distances may be needed to compute the overall length of a tour (see Model II, below), they are stored as the diagonal elements $c_{i,i'}$ in $\boldsymbol{C}$.

It should also be noted that tasks involving only one physical location at a time (e.g., distribution tasks) can easily be adapted to the above model by simply assigning the same coordinates to artificially defined sources and goals. This is why ordered pair and distribution task commands can be issued in the same command sequence (feature (d) in Sec. I-2.).

From Fig. 1 it is also evident that the triangle inequality does not hold in this case: the triangle tour 1' to 2; 2' to 3; 3' to 1, constituting a complete tour according to our model, does not yield $c(1'-2)+c(2'-3) > c(3'-1)$. In practical mobile robot applications the triangle inequality does not hold for yet another reason: a distance in the cost matrix is actually the length of a trip, which is computed as the shortest possible connection between two locations, while avoiding known obstacles. Therefore, a trip may be considerably longer than the straight (euclidean) distances between two locations.

As is seen from the above discussion, the requirements in mobile robot applications exceed the two assumptions of the basic TSP. With regard to the three types of heuristic solutions mentioned in Section II, tour improvement procedures cannot be used for this problem because they require symmetric distance matrices [9]. Similarly, composite procedures cannot be

used since they make use of tour improvement procedures. Therefore, the most suitable heuristic approach is based on tour construction procedures.

Model II: In order to insert a visit to a home depot after a preprogrammed tour length L_{max} has been traveled, the tour must be broken up into subtours, each of which starts and terminates at the home location. The length of the subtour must not exceed L_{max}. The TPG described below includes a mechanism for breaking up the complete tour into subtours. With this mechanism it is easy to adapt the algorithm to other physical constraints of similar type (i.e., where a physical resource is in limited supply and is exhausted during travel). Examples for such constraints include charge in a battery-powered vehicle; time in school bus scheduling or newspaper dispatching; capacity in a vehicle with capacity limitations; or position accuracy of a position sensor (e.g., a gyro), which drifts with increasing tour length or time. TSP's with more than one subtour are also known as multiple traveling salesmen problems (MTSP). Solutions to MTSP's are given in [6] and [10]. However, these solutions assume that the number of subtours to be created is known in advance.

IV. The TPG Algorithm

In mobile robots, onboard computation power is usually limited, both in memory size and in computation speed. For this reason, a TPG algorithm that is small in size and fast is preferred over larger and slower programs with higher accuracy. A simple heuristic algorithm was therefore considered the most suitable for onboard implementation. Out of the three classes of heuristic TSP algorithms (see Section II), we have chosen to work with tour construction procedures, since they offer a straightforward way to accommodate both models (I and II, above).

A tour construction procedure (also called insertion procedure) usually starts with the smallest possible subtour comprising two locations. Then, according to some heuristic rule, the procedure inserts new locations into this subtour until all locations are included and the tour is complete. After each new location has been inserted, the tour length constraint can be tested. This is done by comparing the accumulated tour length (of the current subtour) to L_{max}. If L_{max} is exceeded, the last inserted location is removed and the remaining subtour (now complying with L_{max}) is saved. Subsequently, a new subtour is constructed until all locations have been included. Note that a subtour is near-optimal at any stage of this process, since the newly inserted location was selected by the heuristic selection rule of the algorithm.

The following is the generic form for the closest (or farthest, or arbitrary) insertion algorithm for the standard TSP:

1) Choose a starting location m.
2) Choose a location n such that $c(m,n)$ is minimal (or maximal) and form subtour $m \rightarrow n \rightarrow m$.
3) Find location k, not yet in the subtour, that is closest (or farthest, or arbitrary) to any location in the subtour.
4) Find trip (i,j) in the subtour that minimizes $c(i,k)+c(k,j)-c(i,j)$. Insert k between i and j.
5) Repeat steps (3) and (4) until all locations are included in the tour.

Calculation times for such algorithms are $O(n^2)$ [9]. Typically, one would run the same algorithm n times, choosing each one of the n locations once as the starting location, comparing costs of each run and selecting the tour with the lowest cost as the result. This changes the order of the resulting algorithm to $O(n^3)$, but considerably improves the quality of the solution. In our application, however, subtours are likely to be created because of the tour length constraint. Since subtours must start and end at the same location, the algorithm cannot be run with different starting locations, rendering this means of improvement useless.

Nevertheless, we were able to improve results by running several similar algorithms for the same problem, and selecting the best run as the result. We run the generic insertion algorithm (comprising steps (1)–(5) above) with variations to step 3 (the selection rule). The following six selection rules were employed:

SELECT1: Find location k, not in the subtour, farthest from the starting location.
SELECT2: Find location k, not in the subtour, closest to the last chosen location.
SELECT3: Find location k, not in the subtour, close to the last chosen location, but far from the first location, maximizing $c(1,k)-c(k-1,k)$.
SELECT4: CLOSEST INSERTION: find location k, not in the subtour, closest to any location in the subtour.
SELECT5: FARTHEST INSERTION: find location k, not in the subtour, farthest from any location in the subtour.
SELECT6: CHEAPEST INSERTION: for (i,j) in the subtour find location k, not in the subtour, minimizing $c(i',k)+c(k',j)-c(i',j)$.

Please note that selection rules *1*, *2*, and *3* are our own heuristics, while rules *4*, *5*, and *6* have been suggested in the literature. Also, rules 1–5 are all of $O(n^2)$, while rule 6 is of $O(n^2 \log n)$ [12].

TABLE I
Sample-Run Coordinates

RUN#32 COORDINATES				
#	X'	Y'	X	Y
1	718	279	718	279
2	507	312	705	171
3	33	171	620	233
4	201	11	561	164
5	650	258	391	279
6	595	343	297	330
7	186	143	373	327
8	532	170	479	339

TABLE II
Asymmetric Cost Matrix for the Sample Run

RUN#32 ASYMMETRIC COST-MATRIX C								
#	1	2	3	4	5	6	7	8
1'	0	108	108	194	327	424	348	246
2'	213	243	137	157	120	210	134	38
3'	693	672	590	528	373	308	374	476
4'	582	528	474	391	328	333	359	429
5'	71	102	39	129	259	360	285	189
6'	138	204	112	182	213	298	222	116
7'	549	519	443	375	246	217	262	352
8'	215	173	108	29	178	284	223	177

V. Experimental Results

In order to obtain relevant data for empirical evaluation, 100 random problems, each comprising $n = 15$ locations (one starting location and seven start/goal pairs), were created. Table I gives the coordinates for one such problem, and Table II shows the associated asymmetric cost matrix C. The diagonal elements of C hold the internal distance between the start and goal location of an ordered pair. This information is not needed for

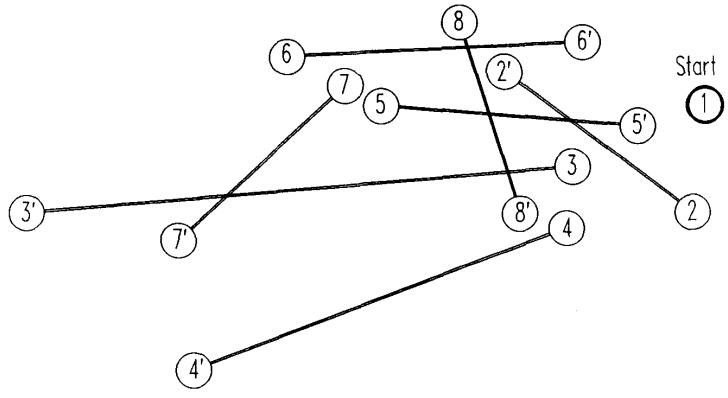

Fig. 2. Sample problem with one start location and seven ordered pairs.

TABLE III
ABSOLUTE OPTIMAL TOUR WITH ONE AND TWO SUBTOURS

RUN#32	
With one subtour:	
Absolute optimal tour: $1 \rightarrow 2 \rightarrow 8 \rightarrow 4 \rightarrow 7 \rightarrow 5 \rightarrow 3 \rightarrow 6 \rightarrow 1$	$C_{opt} = 3485$
$L_{max} = 2613$ (set to $0.75^{*}C_{opt}$ to force 2nd subtour)	
With two subtours:	
Absolute optimal tour: $1 \rightarrow 3 \rightarrow 6 \rightarrow 1 \rightarrow 2 \rightarrow 8 \rightarrow 4 \rightarrow 7 \rightarrow 5 \rightarrow 1$	$C'_{opt} = 3625$

TABLE IV
RESULTS OF INDIVIDUAL SELECTION RULES

RUN#32 ($L_{max} = 2613$, $C'_{opt} = 3625$)			
Selection Rule	Tour	Cost	E_{rel}
SELECT1	$1 \rightarrow 3 \rightarrow 7 \rightarrow 6 \rightarrow 5 \rightarrow 1 \rightarrow 2 \rightarrow 4 \rightarrow 8 \rightarrow 1$	4112	13.43%
SELECT2	$1 \rightarrow 8 \rightarrow 4 \rightarrow 7 \rightarrow 6 \rightarrow 5 \rightarrow 1 \rightarrow 2 \rightarrow 3 \rightarrow 1$	4293	18.42%
SELECT3	$1 \rightarrow 2 \rightarrow 8 \rightarrow 4 \rightarrow 7 \rightarrow 5 \rightarrow 1 \rightarrow 3 \rightarrow 6 \rightarrow 1$	3625	0.00%
SELECT4	$1 \rightarrow 2 \rightarrow 8 \rightarrow 3 \rightarrow 7 \rightarrow 5 \rightarrow 1 \rightarrow 4 \rightarrow 6 \rightarrow 1$	3830	5.65%
SELECT5	$1 \rightarrow 2 \rightarrow 8 \rightarrow 3 \rightarrow 7 \rightarrow 6 \rightarrow 1 \rightarrow 4 \rightarrow 5 \rightarrow 1$	3796	4.71%
SELECT6	$1 \rightarrow 7 \rightarrow 6 \rightarrow 8 \rightarrow 4 \rightarrow 5 \rightarrow 1 \rightarrow 2 \rightarrow 3 \rightarrow 1$	4267	17.71%
TPG's best	$1 \rightarrow 2 \rightarrow 8 \rightarrow 4 \rightarrow 7 \rightarrow 5 \rightarrow 1 \rightarrow 3 \rightarrow 6 \rightarrow 1$	3625	0.00%

choosing an optimal tour (and could be set equal 0), since this distance must be traveled anyway for any possible tour. However, this internal distance must be considered on behalf of the tour length constraint.

Fig. 2 depicts the same example graphically. The starting location is labeled "1", start/goal pairs are labeled "2" through "8", and goal locations are distinguished by primes. Close inspection of Fig. 2 shows the difficulty in finding an optimal tour manually, even for this relatively small problem.

For each one of the 100 problems, the absolute optimal solution was calculated. This was possible because the problem size ($n = 15$) is very small. The cost of the absolute optimal tour (in terms of tour length) is named C_{opt}.

In order to enforce the creation of a tour comprising at least two subtours, the constraint for the maximal tour length L_{max} was then set to 75% (arbitrarily chosen value) of the absolute optimal cost C_{opt}. (Clearly, if the admissible maximal length L_{max} is less than the length of the known absolute optimal tour, then the heuristic algorithm will have to build two or more subtours.) Recalculation of the new optimal cost, now under consideration of the maximal cost constraint L_{max}, yielded a new optimal tour, shown in Table III. The value of C'_{opt} in Table III represents the total cost of the absolute optimal solution of the 2-subtour problem with the tour length constraint.

Subsequently, each one of the six heuristic algorithms was run for each of the 100 random problems. A typical result (for the above example) is shown in Table IV, where the relative error E_{rel} (as compared to C^{*}_{opt}) of each selection rule is shown. Since the TPG always runs all six selection rules for each problem, the TPG can choose the best result out of the six as the representative result. In the example shown in Table IV, selection rule #3 happened to find the exact optimal tour; therefore, the representative result shows $E_{rel} = 0$. Obviously, this is coincidental, and $E_{rel} > 0$ for most representative results. A better indication for the accuracy of the TPG would be an average of the relative

TABLE V
AVERAGE ERROR E_{avrg} PRODUCED BY EACH SELECTION RULE WHEN RUN ALONE

Active rule	Average error E_{avrg} (100 runs)
only SELECT1	7.22%
only SELECT2	7.12%
only SELECT3	8.00%
only SELECT4	7.63%
only SELECT5	7.45%
only SELECT6	10.03%

TABLE VI
AVERAGE ERROR E_{avrg} PRODUCED BY ALL (OR ALL EXCEPT ONE) SELECTION RULES

Active selection rules	Average error E_{avrg} (100 runs)	Contribution to joint performance
All Selection rules	2.63%	
all except SELECT1	3.06%	0.43 ≈ 14.1%
all except SELECT2	2.78%	0.15 ≈ 5.4%
all except SELECT3	3.12%	0.49 ≈ 15.7%
all except SELECT4	2.95%	0.32 ≈ 10.8%
all except SELECT5	2.84%	0.21 ≈ 7.4%
all except SELECT6	2.79%	0.16 ≈ 5.7%

errors of representative results over a large number of problems. We will call such a value E_{avrg}.

VI. INDEX OF PERFORMANCE

As is evident from Table IV, the various selection rules performed quite differently on the same problem. However, none of the selection rules always performed well or poorly on all 100 problems. Additional heuristic selection rules can easily be added to the TPG. Even if the rules were bad, they might produce the best tour every once in a while. However, computation time for the TPG grows with more selection rules to run. A good evaluation method would allow the programmer to identify and include only those heuristic rules that contribute significantly to the accuracy of the algorithm. Therefore, the question arises of how to evaluate the performance of heuristic selection rules. One approach would be to calculate the average relative error produced by each selection rule when run alone [9], [12]. A result of this test is shown in Table V.

However, since the rules function as a team in the actual algorithm, their performance should be evaluated in that context. For this reason, we introduced another test. This time, the TPG was run for all 100 problems, omitting one of the six selection rules each time (i.e., the representative result was chosen out of the remaining five rules). The better E_{avrg} resulting from the remaining selection rules, the lesser the contribution of the omitted selection rule. This approach can be illustrated by imagining a creative think-tank team working on a problem: If only the best idea counts, it is less important how often a team member comes up with another good idea, but rather how often he or she produces the best one.

Table VI shows E_{avrg} for the TPG using all but one selection rule. The first entry, with all selection rules active, produced an average cost of 2.63% above the absolute optimal cost. In each of the succeeding runs, one of the selection rules was omitted.

As can be learned from Table VI, the best contribution comes from selection rule *3*. (To our knowledge, the heuristics of this rule have not been suggested elsewhere in the context of the TSP. However, it must be emphasized that this experiment has been performed with a tour length constraint.) Intuitively, selection rule *3* can be understood as an attempt to fill as many remote locations as possible into one subtour, thereby avoiding the high costs associated with cases where a subtour had to be closed due to L_{max} before covering all locations in a remote area.

From Table VI it can also be learned that selection rules *2* and *6* offer only small contributions. Therefore, one might want to consider eliminating them all together. This would be particularly efficient with rule 6, since this heuristic is computationally more expensive than the others. The remaining rules are all of $O(n^2)$.

VII. CONCLUSION

A TPG optimized specifically for mobile robot applications has been developed. Features of this TPG include the ability to insert home visits into the tour automatically, as well as to deal with ordered source/goal pairs that must be visited in proper sequence. A model has been developed that allows us to represent the problem as a TSP with an asymmetric cost matrix and non-Euclidean distances.

New heuristic tour construction procedures designed specifically for mobile robot requirements have been tested, and they compare favorably with known heuristics. A heuristics team approach was employed to further improve the TPG's performance. The index of performance method introduced in Section VI provides an efficient tool to evaluate the performance of additional selection rules. This method is particularly suited to the heuristic team approach used in this TPG.

REFERENCES

[1] J. Borenstein and Y. Koren, "Optimal path algorithms for autonomous vehicles," presented at the *13th CIRP Manufacturing Systems Seminar*, Stuttgart, West Germany, June 1986.

[2] J. Borenstein, Y. Koren, and R. Weill, "Hierarchically structured multisensor system for an intelligent mobile robot," *CIRP Ann.*, vol. 36/1, pp. 331–334, 1987.

[3] J. Borenstein and Y. Koren, "Real-time Obstacle Avoidance for Fast Mobile Robots," *IEEE Trans. Syst., Man, Cybern.*, vol. 19, pp. 1179–1187, Sept./Oct. 1989.

[4] J. Borenstein and Y. Koren, "Real-time obstacle-avoidance for mobile robots in cluttered environments," *IEEE Int. Conf. on Robot. Automat.*, pp. 572–577, Cincinnati, OH, May 1990.

[5] A. Federgruen and P. Zipkin, "A combined vehicle routing and inventory allocation problem," *Oper. Res.*, vol. 32, no. 5, pp. 1019–1037, Sept./Oct. 1984.

[6] B. Gavish and K. Srikanth, "An optimal solution method for large-scale multiple traveling salesmen problems," *Oper. Res.*, vol. 34, no. 5, pp. 698–717, Sept./Oct. 1986.

[7] B. L. Golden, "Evaluating a sequential vehicle routing algorithm," *AIIE Trans.*, vol. 9, no. 2, pp. 204–208, 1977.

[8] B. L. Golden and A. A. Assad, "Perspectives on vehicle routing: Exciting new developments," *Oper. Res.*, vol. 34, no. 5, pp. 803–810, Sept./Oct. 1986.

[9] B. Golden, L. Bodin, T. Doyle, and W. Stewart, "Approximate traveling salesman algorithms," *Oper. Res.*, vol. 26, no. 3, pp. 694–711, May/June 1980.

[10] R. Jonker and T. Volgenant, "An improved transformation of the symmetric multiple traveling salesman problem," *Oper. Res.*, vol. 36, no. 1, pp. 163–167, Jan./Feb. 1988.

[11] A. W. J. Kolen, A. H. G. Rinnooy Kan, and H. W. J. M. Trienekens, "Vehicle routing with time windows," *Oper. Res.*, vol. 25, no. 2, pp. 266–273, Mar./Apr. 1987.

[12] E. L. Lawler *et al.*, *The Traveling Salesman Problem*. New York: Wiley, 1985.

[13] S. Lin, "Computer solutions of the traveling salesman problem," *Bell System Tech. J.*, vol. XLIV, no. 10, pp. 2245–2269, Dec. 1965.

[14] S. Lin and B. W. Kernighan, "An effective heuristic algorithm for the traveling salesman problem," *Oper. Res.*, vol. 21, no. 2, pp. 498–516, 1973.

[15] J. Rosenkrantz, R. E. Stearns, and P. M. Lewis, "Approximate algorithms for the traveling salesperson problem," in *Proc. 15th Annual IEEE Symp. Switching and Automata Theory*, pp. 33–42, 1974.

[16] M. M. Solomon, "Algorithms for the vehicle routing and scheduling problems with time window constraints," *Oper. Res.*, vol. 35, no. 2, pp. 254–265, Mar./Apr. 1987.