Journal of Engineering Science and Technology
Vol. 6, No. 2 (2011) 215 - 227
© School of Engineering, Taylor's University

OPTIMAL TOUR CONSTRUCTIONS FOR MULTIPLE MOBILE ROBOTS

AMIR A. SHAFIE

Autonomous Agent Group, Faculty of Engineering, International Islamic University Malaysia (IIUM), P.O. Box 10, 50728 Kuala Lumpur, Malaysia
E-mail: aashafie@iium.edu.my

Abstract

The attempts to use mobile robots in a variety of environments are currently being limited by their navigational capability, thus a set of robots must be configured for one specific environment. The problem of navigating an environment is the fundamental problem in mobile robotic where various methods including exact and heuristic approaches have been proposed to solve the problem. This paper proposed a solution to the navigation problem via the use of multiple robots to explore the environment employing heuristic methods to navigate the environment using a variant of a Traveling Salesman Problem (TSP) known as Multiple Traveling Salesman Problem (M-TSP).

Keywords: Mobile robot, TSP, Autonomous systems.

1. Introduction

Technological robotic hardware advancement has enable manufacturers to build mobile robot to be used in a variety of environments including areas where the mobile robot has to serve many service points. The tasks of the mobile robot might be to deliver components and parts to areas with various geographical [1], hospital services [2] and farming application in controlled environments [3]. In many of these applications teams of robots can offer more advantages over single robots as they offer greater flexibility through dynamic team coordination and reorganization, greater efficiency through parallel task execution and greater reliability through resource redundancy [4].

For a single mobile robot, motion planning algorithms are used to construct navigation paths that are safe and optimal. Basic motion planning algorithms deal with planning of motion of a robot between starting location and a terminal location

Nomenclatures

A	All possible connection between V and C
C	Set of service points
G	Maximum number of evolutionary generation
H	Number of chromosomes in the pool
K	A specific robot in V
NP Complete	Nondeterministic polynomial time
p_m	Mutation rate
R	Data sets where points are randomly dispersed
RC	Data set that has a mix of random and clustered data points
u	Number of copies for tournament selection
V	Set of identical robots
Abbreviations	
M-TSP	Multiple Traveling Salesman Problem
TSP	Traveling Salesman Problem
VRP	Vehicle Routing Problem
VRPTW	Vehicle Routing Problem with Time Windows

Existing approaches plan an initial path based on known information about the environment, then modify the plan locally as the robot travels or re-plan the entire path as the robot discover obstacles with its sensors, sacrificing optimality and computational efficiency respectively. Motion planning for a team of mobile robots will also need to include the solution for coordination of movement among team members as the tasks will comprise missions that are spatially distributed over a geographical area.

The problem of routing robots over available paths between target locations (corresponding to the specific tasks) is known as multi-robot routing. The objective of the routing algorithm is to find a route for each robot so that each target location is visited once by exactly one robot (no waste of resources), all target locations are eventually visited by some robot (mission completeness) and the entire mission is accomplished successfully (optimization of performance).

As the mobile robot is required to service many locations and return to the starting location, then the problem turns to a tour construction problem and can be considered as a Traveling Salesman Problem (TSP). TSP deals with finding a tour (route) for a salesman who starts from a home location, visits a prescribed sets of cities (locations) and returns to the original location in such a way that the total distance travelled is minimum and each city is visited exactly once [5]. TSP, which falls under the categories of hard problem is one of the most famous and well-studied problem in the area of combinatorial optimization thus many possible solutions have been proposed [6].

Mobile robots usually have limited traveling distance due to the limitation of the power source (battery). Thus efficient use of energy is important [7] and a shorter path is preferable than a longer path when performing a given task. If the robot travels between any two points, a shortest path algorithm may be used to plan the path of the robot. For example, Stentz [8] used D* algorithm to construct paths for robots in partially known and unknown-dynamic environments. An extension of this approach by Cagigas and Abascal [9] uses a hierarchical graph to model large scale stationary environments and constructs paths using a new hierarchical extension of the D* algorithm. An extension of the algorithm includes a multi-criteria path planner that provides an efficient and natural way of both defining and solving problems in which conflicting criteria are involved [10]. In other studies [11, 12], a neural dynamics based approach is proposed for real-time motion planning with obstacle avoidance of a mobile robot in a non-stationary environment. Some other approaches in mobile robot motion planning include genetic algorithms [13] and fuzzy logic [14].

A time window associated with each of the service points is also a limitation for the mobile robots where at each point a time frame where a particular service or task must be completed, such as loading or unloading (service time). A robot might arrive early but it must wait until start of service time is possible. The objective of the problem then extends further than the original TSP where the robot now must travel to service the points without violating the capacity (power) and time window constraints. In many of the proposed solution the capacity limitation is only the distance traveled [15].

The implications of implementing the TSP towards a team of robots further create an extension of TSP known as the multiple travelling salesman problem (M-TSP) with time windows. The M-TSP with time windows (also known as Vehicle Routing Problem with Time Windows (VRPTW) have also been generalized to a wide variety of routing and scheduling problems, for example, the School Bus Routing Problem [16] and the Pickup and Delivery Problem [17]. However, M-TSP with time windows is also an NP-complete problem for which optimal solutions can only be found for small size problems. It is known that classical optimization procedures are not adequate for this problem. Good heuristic techniques are necessary for solving M-TSP due to its high computational complexity. Modern heuristic techniques, namely genetic algorithms as applied here can be good candidates for this problem.

Section two (2) of this paper provides the description of M-TSP with time windows specifications and section three (3) presents the proposed heuristic techniques used to find acceptable tours for the problem. Experimental details and results are presented in section four (4) while section five (5) concludes the paper with an outline of further work.

2. Background

2.1. Description of M-TSP with time windows

The M-TSP with time windows is represented by a set of identical robots denoted by V, and a directed graph $G = (C, A)$, which consist of a set of points, C. The nodes 0 and $n + 1$ represent the base point, i.e., exiting point, and returning point respectively. The set of n vertices denoting service points is denoted N. The arc set A denotes all possible connections between the nodes (including node denoting base point). No arc terminates at node 0 and no arc originates at node $n + 1$ and all routes start at 0 and end at $n + 1$.We associate a cost C_{ij} and a time t_{ij} with each arc $(i, j) \in A$ of the routing network.

The travel time $t_{i,j}$ may include service time at point i. Each robot has a distance limit q and each point i has a distance d_i , $i \in C$. Each point i has a time window, $[a_i,b_i]$, where a_i and b_i are the respective opening time and closing times of i. A robot may arrive before the beginning of the service time window (i.e., a_i) meaning incur waiting time until service is possible. However, no robot may arrive past the closure of a given time interval, b_i. Robots must also leave the service point within the service point time window $[a_0, b_0]$ and must return before or at time b_{n+1}. Assuming waiting time is permitted at no cost, we may assume that $a_0=b_0= 0$; that is, all routes start at time 0.

The model has two types of decision variables x and s. For each arc (i,j), where $i = j$, $i = n + 1$, $j = 0$, and each robot k, the decision variable x_{ijk} is equal to 1 if robot k drives from vertex i to vertex j, and 0 otherwise. The decision variable s_{ik} denotes the time robot $k,k \in V$ starts to service point $i,i \in C$. If vehicle k does not service point i , then s_{ik} has no meaning. We may assume that $a_0 = 0$ and therefore $s_0k = 0$, $\forall k$. The objective of the M-TSP with time windows is to service all the C service points using the V robots such that the following objectives are met and the following constraints are satisfied.

Objectives

- Minimize the total number of robots used to service the service points.
- Minimize the distance traveled by the robots.

Constraints

- Robot capacity constraint is observed.
- Time window constraint should be observed.
- Each service points are serviced exactly once.
- Each robot route starts at vertex 0 and ends at vertex n+1.

Figure 1 shows a simple graphical model of the M-TSP with time windows and its solution. In this example, there are two routes, route 1 with 4 service points and route 2 with 5 service points.

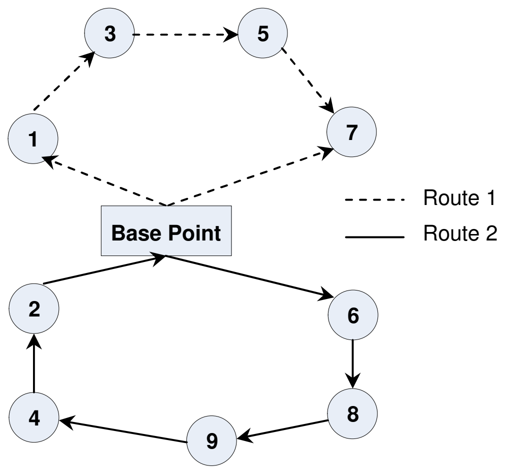

Fig. 1. Sample Solution for M-TSP (2 robots 1 Base Points).

The M-TSP with time windows model can be mathematically formulated as shown below:

$$\text{Min} \quad \sum_{k\in V}\sum_{i\in N}\sum_{j\in N} c_{ij}x_{ijk} \quad \text{such that} \tag{1}$$

$$\sum_{k\in V}\sum_{j\in N} x_{ijk} = 1 \qquad \forall i \in C \tag{2}$$

$$\sum_{i\in C} d_i \sum_{j\in N} x_{ijk} \le q \qquad \forall k \in V \tag{3}$$

$$\sum_{j\in N} x_{0jk} = 1 \qquad \forall k \in V \tag{4}$$

$$\sum_{i\in N} x_{ihk} - \sum_{j\in N} x_{hjk} = 0 \qquad \forall h \in C, \forall k \in V \tag{5}$$

$$\sum_{i\in N} x_{i,n-1,k} = 1 \qquad \forall k \in V \tag{6}$$

$$s_{ik} + t_{ij} - k\left(1 - x_{ijk}\right) \le s \qquad \forall i \in N, \forall j \in N, \forall k \in V \tag{7}$$

$$ot_i \le s_{ik} \le ct_i \qquad \forall i \in N, \forall k \in V \tag{8}$$

$$x_{ijk} \in \{0,1\} \qquad \forall i \in N, \forall j \in N, \forall k \in V \tag{9}$$

$V=\{1,2,...,k\}$ robots	C=$\{1,2,...,n\}$ service points
d_i service point i demand	a_i service point i open time
b_i service point i close time	$0,n+1$ station
$N=\{0,1,...,n,n+1\}$ node size	q_k robot i capacity
t_{ij} service point i from j time	s_{ik} service point i take k service time

The objective function (1) states that costs should be minimized. The constraint set (2) states that each service point must be visited exactly once by one robot, and constraint set (3) states that the robot capacity should not be exceeded. The next set of constraints (4), (5) and (6) gives the flow constraints that ensure that each robot leaves base points 0, departs from a service points it visited and finally returns to the base point, given by node n + 1. The nonlinear inequality (7) (which can be easily linearized, see [1]) states that a robot K cannot arrive at j before s_{ik} + t_{ij} if it travels from from i to j. Constraint (8) ensures that time windows are observed and (9) gives the set of integrality constraints.

3. Solution Procedure

Generalized TSP solution procedure generally prescribed is an iterative interaction between the main problem and many sub-problems as illustrated in Fig. 2 [18]. The main problem determines a clustering result that divides all service points into m groups by minimizing the total cost function. This clustering information is then passed to sub-problems, and each sub-problem optimizes its own routing sequence. The summation of the objective values of all sub-problems is returned to the main problem as a performance index for evaluating the current clustering result.

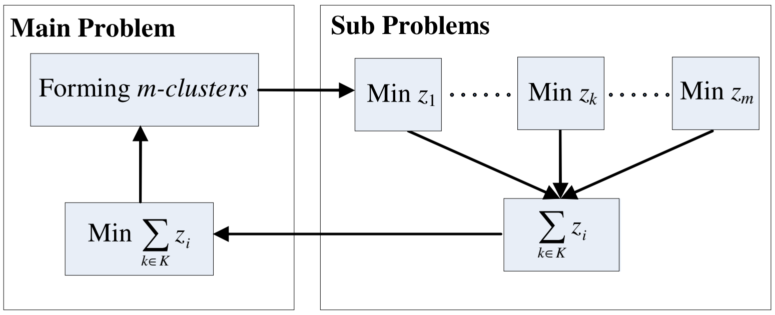

Fig. 2. Interaction between Main Problem and Sub-problem.

Although the decomposition of the original M-TSP with time windows has resulted in a set of smaller problems, these new problems are still difficult to solve. Thus, in the present study, the main problem is solved by a genetic algorithm, and each sub-problem is solved by a heuristic algorithm.

The flow chart of the algorithm is depicted in Fig. 3. The decision variables, y_{ik} of the main problem are first encoded as a string called chromosome. As an example, assuming that the base point has three robots and seven service points to serve, the chromosome can be represented by the following string in Table 1.

Table 1. Chromosome String Representation.

Service Points	1	2	3	4	5	6	7
Chromosome	2	2	3	1	2	3	3

In the above example, the chromosome means that node 4 will be visited by robot 1, nodes 1, 2 and 5 by robot 2, and nodes 3, 6 and 7 by robot 3. Equivalently, the decision variables y_{ik} are determined from the chromosome as y_{41}=y_{12}= y_{22}=y_{52}=y_{33} =...=y_{63}=y_{73}= 1, and y_{ik} = 0 for all the remaining *i* and *j*. The encoding of decision variables in this manner rather than using their original binary value can greatly shorten the length of chromosomes.

The fitness function in the genetic algorithm is used to evaluate the performance of a chromosome, i.e., the optimality of a set of decision variable values. Thus, the fitness function of the algorithm is defined by the objective function of the clustering problem. In the beginning of the algorithm, a pool of chromosomes is randomly generated, where each chromosome represents a clustering result. Each chromosome's fitness is evaluated by solving a set of sub-problems as discussed earlier. If current chromosomes are satisfactory, then the algorithm terminates; otherwise, a new pool of chromosomes is generated through a guided evolution. This evolutionary process contains operations of reproduction, crossover, and mutation. The reproduction operation is carried out by the method of tournament selection [19] which first selects some chromosomes with better fitness in the pool and makes copies of these selected chromosomes, then uses the roulette wheel selection or uniform selection to pick chromosomes from the pool for reproduction. The well-known two-point crossover is used to generate new chromosomes by swapping designated bits of a pair of chromosomes. The order-based mutation [20] is also adopted to produce heterogeneous chromosomes in the pool to avoid early convergence of the algorithm. This mutation method is to randomly select a chromosome from the pool, and then randomly picks two bits in this chromosome and swaps them.

The detailed steps of the algorithm for solving the main clustering problems follows the norm for genetic algorithm [21] with minimum modification to suit M-TSP with time windows are as follows:

```
Step 0. Initialization.
        Total number of chromosomes in the pool, H.
        The maximum number of evolutionary generation, G.
        Number of copies for tournament selection, u.
        Crossover rate $p_c$; mutation rate $p_m$.
        Generation counter g 1.
Step 1. Initial gene pool.
        Step 1.1. Randomly generate a chromosome.
        Step 1.2. Check the satisfaction of capacity constraint for this chromosome.
                  If Eq. (3) is not satisfied then discard this chromosome and go
                  back to Step 1.1
        Step 1.3. If the total number of chromosomes is H then go to Step 2;
        otherwise go back to Step 1.1
Step 2. Fitness evaluation
        Step 2.1. For each chromosome in the pool, solve its corresponding set
        of sub-problems, and obtain its fitness value $P_{k2K}z_k$
        Step 2.2. Find the minimum fitness value, F*, of all the chromosomes in
        the current generation
        Step 2.3. Find the minimum fitness value, F**, of all the generations so far
Step 3. Stop criteria
        If F__ is satisfactory or g=G then stop; otherwise go to Step 4
Step 4. New chromosome generation g g + 1
        Step 4.1. Reproduction by tournament selection
        Step 4.2. Crossover
                  Randomly pick chromosomes from the pool with a probability pc
                  Apply the two-point crossover on the picked chromosomes
        Step 4.3. Mutation
                  Randomly pick chromosomes from the pool with a probability pm
                  Apply the order-based mutation on the picked chromosomes
Step 5. Chromosome sifting
        Check the satisfaction of capacity constraint for each chromosome in the
        new pool
        If Eq. (3) is not satisfied for a chromosome then discard it

Step 6. Go to Step 2.
```

In step 2 of M-TSP with time window algorithm, the fitness evaluation of a chromosome is obtained through solving a set of sub-problems, where each of them is an independent TSP with time window problem. The TSP with time window is also an NP-hard problem; thus, a heuristic algorithm is being used to find an acceptable solution the problem. This algorithm determines the sequence of Service points to visit according to their latest acceptable arriving time, l_i. The rationale behind this rule is to reduce the likelihood of violating time window constraints in the later portion of the routing sequence. The steps of this algorithm for solving the k_{th} sub-problem are described as follows:

Step 1. *Sort service points by the ascending order of their latest acceptable arriving time*

Step 1.1. *For all i, je N_k, position i before j if $l_i < l_j$; otherwise, go to Step* 1.2

Step 1.2. *If $l_i = l_j$ $(i \neq j)$ and $l_i – e_i < l_j – e_j$ then position i before j; otherwise position j before i*

Step 3. *Compute the return time*

$$T_k = d_k + \sum_{i \in N_k} \sum_{j \in N_k} \left[x_{ijk} \max\{t_j, i_j\} + s_j \right]$$

If $T_k > r_k$, then return the information of infeasibility to the main problem; otherwise go to Step 4.

Step 4. *Compute*

$$z_x = \sum_{i \in N_1} \sum_{j \in N_2} C_{ij} x_{ijk} + \sum_{j \in N_2} p_i(t_i)$$

Return z_k to the main problem.

In Step 1.2 of the algorithm, the rule is to prioritize the service points with a tighter time window when two service points have the same latest acceptable arriving time.

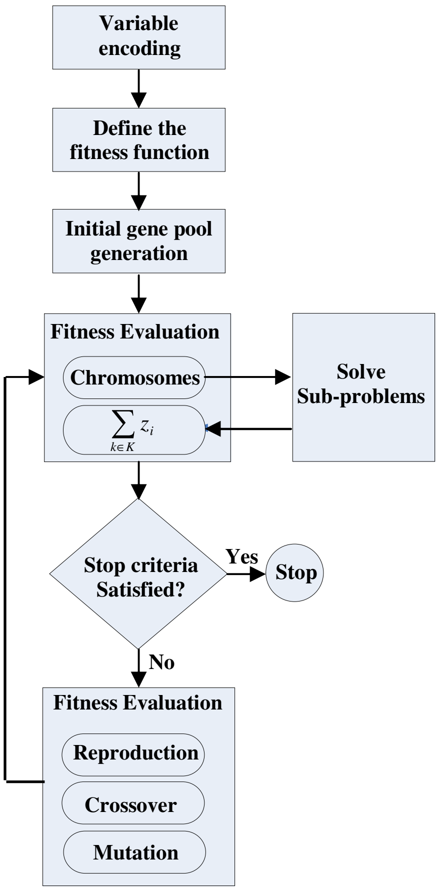

Fig. 3. Heuristic Algorithm for M-TSP.

4. Computational Experiments

In this section, the algorithms described in Section 3 is tested on randomly generated problem set that have been formulated according to the standard set developed by Solomon [22]. Three different types of problems are named *Rx*, *Cx*, and *RCx* in which *x* designates the number of service points in the problem. The problems with the same number of service points in the same type are further denoted by an index preceded by an underline score. Such as *R*10_1 and *R*10_2 are two different problems both have ten service points in type *R*. In the problem set of type *R*, the service points' coordinates are distributed by a random uniform distribution. Service points are clustered in problems of type *C* and semiclustered in problems of type RC. The semiclustered problem is the one that contains a mix of randomly generated and clustered data. All graphical coordinates in test problems are generated within a $[0, 100]^2$ square.

To justify the performance of the proposed approach, we conducted computational experiments in two perspectives. First, the performance of M-TSP with time windows is evaluated using different problem sizes. Then the effects of changing the number of robots for a set of service area are then evaluated.

4.1. Effect of problem size

Figures 4 to 6 show some of the network topologies obtained after running M-TSP, where the proposed solution are presented for 100 service points using 3 mobile robots. Figure 4 represents data sets (*R*) where service points are randomly dispersed. Figure 5 represents service points that are clustered (*C*) while Fig. 6 the service points are mixed (*RC*). It should be noted that nodes in the *R* category networks are much harder to solve than in *C* category. Due to space limitation only three network topologies are shown here, however, the general behavior is representative of the respective data sets.

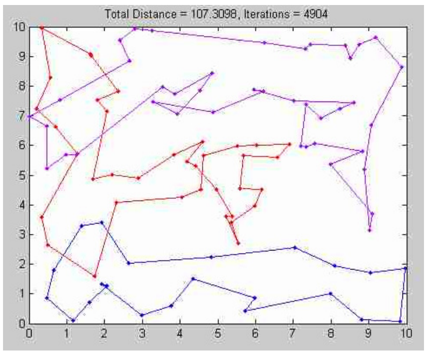

Fig. 4. Solution for 100 Service Points for Random (*R*) Distribution.

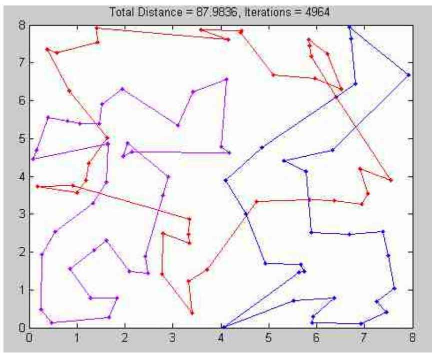

Fig. 5. Solution for 100 Service Points for Clustered (*C*) Distribution.

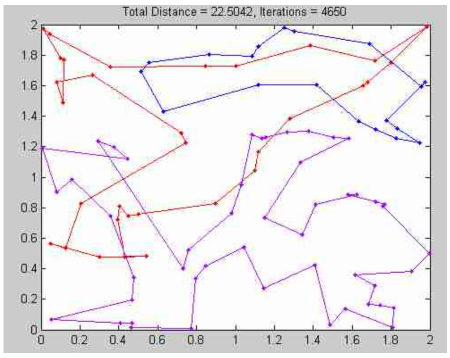

Fig. 6. Solution for 100 Service Points for Random Clustered (*RC*) Distribution.

In order to evaluate the performance of our algorithm on different problem sizes, the original problem of the distribution center is expanded artificially in terms of the number of vehicles and the number of retailer nodes. The computational results are shown in Table 2, where we can see that all problems are solved in acceptable time. These results suggest that our approach is robust for different problem sizes.

Table 2. Computational Result between Single and Multiple TSP.

Instance	Num. Service Points	M-TSP	Instance	Num. Service Points	M-TSP
R10_1	10	27.972	**R10_2**	10	27.132
	100	116.154		100	107.301
	1000	1562.080		1000	1222.354
C10_1	10	27.138	**C10_2**	10	24.354
	100	87.984		100	97.166
	1000	1119.823		1000	1175.966
RC10_1	10	26.648	**RC10_2**	10	26.648
	100	82.504		100	80.655
	1000	1705.966		1000	1880.181

4.2. Effect of mobile robots number variation

The present study suggested that by dividing the original problem into multi traveling salesman problems then for each of the sub-problem can be handled satisfactorily by ordinary mobile robots. The advantages of this decomposition are to reduce the problem size and to solve sub-problems that are simpler than the original problem. To assess this argument, a genetic algorithm that is similar to the one proposed in Section 3 is developed to solve the original M-TSP with time windows problem as shown in Table 3.

Table 3. Computational Result on Different Problem Sizes.

Number of Robots	Service Points	Traveling Cost	Average
1	20	35.427	35.427
	30	48.863	48.863
	40	54.278	54.278
3	20	38.031	12.677
	30	48.941	16.314
	40	59.512	19.837
5	20	39.169	7.833
	30	51.761	10.372
	40	60.669	12.134

5. Conclusions

This study has proposed an approach for solving the robot routing problem with time windows. Our approach takes advantage of the mobile robot special structure which facilitates the decomposition of the original problem into a main clustering problem and many TSP with time window sub-problems. This decomposition not only reduces the problem size but also changes the original problem to many relatively simpler problems. The main clustering problem is solved by a genetic algorithm and each sub-problem is independently solved by a heuristic algorithm. The whole solution procedure contains iterative interactions between the main problem and the set of sub-problems. Through these iterative interactions the solution of the original problem is improved gradually.

Perhaps most significantly, our interpretation of the M-TSP represents a philosophically different view of the problem as the whole. When the multi robot routing is viewed with-out bias towards number of mobile robots or total cost, we are afforded with a more natural multi-objective perspective for this application problem. No unnecessary bias is introduced into the search. This is in stark contrast to most other work in multi robot routing, in which the number of robot used is given implicit priority, and consequently the scoring procedure must prioritize this dimension of the problem.

We claim that there is no theoretical or practical advantage to giving priority to the number of robots, perhaps other than having a common framework from which to compare different researcher's results. Admittedly, there is an associated cost to having more robots. However, there is also an associated cost to the additional fuel and time used in using fewer robots at longer distances to service points. Furthermore, robot counts can be less important when robots and manpower costs are low. By considering minimal cost (distance), we reduce energy consumption. Such ecological considerations are arguably of growing concern in a world of greenhouse gases and a depleted ozone layer.

In any case, the M-TSP is naturally multi-objective, and neither dimension is fundamentally more important than the other from a theoretical perspective and even from a practical aspect, it is arguably debatable as to whether the optimization search should be biased towards minimizing the number of robots deployed as most current research work on M-TSP tends to do. Hence, as can be seen with our results, the proposed approach generates a set of equally valid M-TSP solutions.

Acknowledgment

The Ministry of Higher Education, Malaysia (MOHE) kindly provided funding for the research through the Fundamental Research Grant Scheme (FRGS 0106-15).

References

1. Hada, Y.; Gakuhari, H.; Takase, K.; and Hemeldan, E.I. (2004). Delivery service robot using distributed acquisition, actuators and intelligence. *Proceedings of the International Conference on Intelligent Robots and Systems*, 3, 2997-3002.
2. Shieh, M.Y.; Hsieh, J.C.; and Cheng, C.P. (2004). Design of an intelligent hospital service robot and its applications. *Proceedings of the IEEE International Conference on Systems, Man and Cybernetics*, 5, 4377-4382.
3. van Henten, E.J.; Hemming, J.; van Tuijl, B.A.J.; Kornet, J.G.; Meuleman, J.; Bontsema, J.; and van Os, E.A. (2002). An autonomous robot for harvesting cucumbers in greenhouses. *Autonomous Robots*, 13(3), 241-258.
4. Jones, E.; Dias, M.B.; and Stentz, A. (2011). Time-extended multi-robot coordination for domains with intra-path constraints. *Autonomous Robots*, 30(1), 41-56.
5. Punnen, A.P. (2002). The traveling salesman problem applications, formulations and variations, in: Gutin, G.; and Punnen, A.P. (Eds.). *The traveling salesman problem and its variations*. 1st Ed. Springer,1-28.
6. Laporte, G. (1992). The traveling salesman problem: An overview of exact and approximate algorithms. *European Journal of Operational Research*, 59(2), 231-247.
7. Mei, Y.; Lu, Y.H.; Hu, Y.C.; and Lee, C.S.G. (2004). Energy-efficient motion planning for mobile robots. *Proceedings of IEEE International Conference on Robotics and Automation*, 5, 4344-4349.
8. Gonzales, J.P.; and Stentz, A. (2005). Planning with uncertainty in position: an optimal and efficient planner. *Proceedings of the IEEE International Conference on Intelligent Robots and Systems (IROS'05)*, 2435-2442.
9. Cagigas, D.; and Abascal, J. (2005). A hierarchical extension of the D* algorithm. *Journal of Intelligent & Robotic Systems*, 42(4), 393-413.

10. Ossowski, S.; Fernandez, A.; Serrano, J.M.; Hernandez, J.Z.; Garcia-Serrano, A.M.; Perez-de-la-Cruz, J.L.; Belmonte, M.V.; Maseda, J.M. (2004). Designing multiagent decision support system - The case of transportation Management. *Proceeding of the Third International Joint Conference on Autonomous Agent and Multiagent Systems*, 3, 1470-1471.
11. Ge, S.S.; and Cu, Y.J. (2002). Dynamic motion planning for mobile robots using potential field method. *Autonomous Robots*, 13(3), 207-222.
12. Yang, S.X.; and Meng, M.Q.H. (2003). Real-time collision-free motion planning of a mobile robot using a neural dynamics-based approach. *IEEE Transactions on Neural Networks*, 14(6), 1541-1552.
13. Liang, Y.; and Xu, L. (2009). Global path planning for mobile robot based genetic algorithms and modified simulated annealing algorithm. *Proceedings of ACM/SIGEVO Summit on Genetic and Evolutionary Computation*, 303-308.
14. Garibaldi, J.; Barreras, A.; and Castilo, O. (2007). Intelligent control of autonomous mobile robot using fuzzy logic and genetic algorithms. *Studies in Fuzziness and Soft Computing*, 208, 255-265.
15. Sipahioglu, A.; Yazici, A.; Parlaktuna, O.; and Gurel, U. (2008). Real-time tour construction for a mobile robot in a dynamic environment. *Robotics and Autonomous Systems*, 56(4), 289-295.
16. Kolen, A.W.J.; Rinnooy Kan, A.H.G.; and Trienkens, H.W.J.M. (1987). Vehicle routing and scheduling with time windows. *Operations Research*, 35(2), 266–273.
17. Lim, K.H.; and Lee, M.J. (2007). Scheduling trucks in local depots for door-to-door delivery services. *Journal of the Operational Research Society*, 58(9), 1195-1202.
18. Meeran, S.; and Shafie, A. (1997). Optimum path planning using convex hull and local search heuristics. *Mechatronics*, 7(8), 737-756.
19. Goldberg, D.E. (1989). *Genetic algorithms in search, optimization and machine learning*. Addison-Wesley Publishing Company, Inc.
20. Lin, F.T.; Kao, C.Y.; and Hsu, C.C. (1993). Applying the genetic approach to simulated annealing in solving some NP-hard problems. *IEEE Transactions on Systems*, *Man and Cybernetics*, 23(6), 1752-1767.
21. Holland, J.H. (1975). *Adaptation in natural and artificial systems*. Ann Arbor, Michigan: The University of Michigan Press.
22. Larsen, A.; Madsen, O.B.G; Solomon, M.M. (2004). The a priori dynamic travelling salesman problem with time windows. *Transportation Science*, 38(4), 459-472.