# Articulated Robot Motion Planning Using Ant Colony Optimisation

Mohd Murtadha Mohamad, Nicholas K. Taylor, and Matthew W. Dunnigan, *Member, IEEE*

***Abstract*—A new approach to robot motion planning is proposed by applying Ant Colony Optimization (ACO) with the probabilistic roadmap planner (PRM). The aim of this approach is to apply ACO to 3-dimensional robot motion planning which is complicated when involving mobile 6-dof or multiple articulated robots. An ant colony robot motion planning (ACRMP) method is proposed that has the benefit of collective behaviour of ants foraging from a nest to a food source. A number of artificial ants are released from the nest (start configuration) and begin to forage (search) towards the food (goal configuration). During the foraging process, a 1-TREE (uni-directional) searching strategy is applied in order to establish any possible connection from the nest to goal. Results from preliminary tests show that the ACRMP is capable of reducing the intermediate configuration between the initial and goal configuration in an acceptable running time.**

***Index Terms*— Ant colony, robot path planning, search technique**

## I. Introduction

This paper describes a novel application of swarm intelligence to robotic arm manipulator motion planning. The probabilistic roadmap (PRM) is among the most efficient methods for planning robot motion. A PRM is a discrete representation of a continuous configuration space (C-space) generated by randomly sampling the free configurations of the C-space and connecting the points into a graph. The Single query Bi-directional with Lazy collision checking PRM (SBL-PRM) [11], [13] is a variation of the PRM which, instead of pre-computing the roadmap, uses two input query configurations as seeds to explore the space. It explores the robot's free space by concurrently building a roadmap made of two trees rooted at the query configurations (bi-directional). It delays collision tests along the edges of the roadmap until they are needed (lazy collision checking). Ant Colony Optimisation (ACO) is a swarm intelligence approach to solving optimisation problems. ACO has been successfully used to optimise the travelling salesman problem (TSP) and the quadratic assignment problem (QAP) [5]. To date, the applications of swarm intelligence in robotics have been primarily with collections of robots or with sets of transfer functions in factory production lines to solve one major task. References [16], [17] managed to solve the robot path planning for 2-dimensional space which involved a mobile robot of 2 to 3 degrees of freedom (dof). Applying ACO to a robot motion planner in which a mobile robot has 6-dof or more in 3-dimensional space, it is not straightforward. This paper introduces a new approach by applying ACO to robot motion planning, of manipulator arm type robots. It focuses on applying ant colony behaviour to search the robot's C-space. Preliminary experiments have been undertaken using SBL-PRM multi-goal motion planning as a benchmark for this new algorithm. Our aim is to get a faster planner and reduce the number of intermediate configurations between the two query configurations start and goal).

M. M. Mohamad is a PhD. Student of the Department of Electrical, Electronics, and Computer Engineering of School of Engineering and Physical Sciences, Heriot-Watt University (telephone: 44-131-4513265, e-mail: M.Mohamad @hw.ac.uk).

M. W. Dunnigan, is a Senior Lecturer in the Department of Electrical, Electronics, and Computer Engineering in the School of Engineering and Physical Sciences, Heriot-Watt University (telephone: 44-131-4513346, e-mail: M.W.Dunnigan@hw.ac.uk)..

N. K. Taylor is a Senior Lecturer in the Department of Computer Science in the School of Mathematical and Computer Sciences, Heriot-Watt University, Edinburgh (telephone: 44-131-4513436, e-mail: N.K.Taylor@hw.ac.uk).

## II. Relation to Previous Work

### A. Works on Probabilistic Roadmap

The problem of robot motion planning in known workspaces has been studied extensively over the last two decades [8]. PRM's have been proven to be an effective tool in solving motion-planning problems with many degrees of freedom [7], [13]. PRM applies a roadmap approach where it constructs a roadmap of paths in the free configuration space [8]. A roadmap is a pre-computed undirected graph covering the entire free space. Once the roadmap is constructed, it is used as a set of standardized paths. Path planning is then applied by connecting the initial and goal configurations to points in the roadmap and then searching the roadmap for the path between these points. Kavraki and Latombe presented the randomized techniques of PRM in [6], [7].

The work in [1] introduced PRM with lazy collision-checking (lazy-PRM). This planner assumes all nodes and paths are collision-free during the pre-computation phase. It then searches the roadmap at hand for the shortest path. The nodes and path are checked for collisions afterwards. If collisions have occurred, the corresponding nodes and edges are removed. The planner either finds the new shortest path, or first updates the roadmap with the new nodes and edges, then searches for the shortest path.

The approach in [11], [13] introduced single-query bi-directional path planning with lazy collision-checking (SBL-PRM). This single-query approach, instead of pre-computing a roadmap covering the entire free space, uses the two input query configurations to explore as little space as possible. The bi-directional approach explores the

1-4244-0195-X/06/$20.00 ©2006 IEEE

robot's free space by building a roadmap made of two trees rooted at the query configurations. Lazy collision-checking means that the collision tests are delayed along the edges of the roadmap until they are absolutely needed. The SBL-PRM is applied to multi-robot path planning in [12].

A continuation of SBL-PRM is performed in [10] a problem of multi-goal motion planning which combines two problems: the shortest path and traveling-salesman problem. The planner uses a greedy algorithm that assumes the number of goals are relatively small and the computational cost of finding a good path between two goals dominates that of finding a good tour in a graph with edges of given costs.

### *B. Ant Colony Optimization*

Ant Colony Optimization (ACO) studies the artificial systems that take inspiration from the behaviour of ant colonies which is one of the swarm intelligence approaches. Swarm intelligence is the property of the system whereby the collective behaviour of ants (agents) interacting locally with their environment collectively carry out distributed problem solving. As far as this is paper concerned, the ACO is a new approach to the robot motion planning field. The current applications of ACO are on routing, assignment, scheduling, subset and network type problems.

Works in [2]-[5] describe the most common problem in routing as the famous traveling salesman problem. In assignment type problems, [9] apply ACO in the quadratic assignment.

## III. BACKGROUND

In general terms, PRM is a two stage motion planner, the first is the pre-processing phase and the second is path planning. In the pre-processing phase, a set of collision-free configuration spaces are generated and interconnected onto a network using a simple path planning technique applied to pairs of neighbouring nodes. This path planning technique is a local planner that does not store the information of connections in the roadmap since it is fast, although not so powerful, and the call to the planner is inexpensive. This kind of planner is called a deterministic local planner. The network may contain one or more connected components depending on the robot free space and pre-processing time. At the end of inter-connections between nodes phase, the narrow parts of C-space are enhanced by adding a few nodes to cover the difficult parts and gives the planner greater probability to form a large component. The path planning phase is the process of connecting between two configurations to two nodes, $A$ and $B$ in the network. Then the network is searched for a sequence of edges connecting $A$ and $B$. A smoothing technique can be used to improve the path found. The planner is considered to fail if it cannot connect two input configurations to the network or if $A$ and $B$ lie in two different connected components.

During the generation of a random configuration, each degree of freedom (dof) is drawn uniformly from its allowed range. The resulting configuration is checked for collision with obstacles and self collision. A local planner that has been used is a straight line in C-space. The other planner is one that can be used for the robots which can move a few of its dof's in a certain direction and adjust the others. It works in the following manner. The odd joints are simultaneously translated each time period along a straight line in the workspace that connects its workspace configuration $p$ to its workspace at configuration $q$. Then, the position of even joints is adjusted by computing the inverse kinematics of the robot. For roadmap enhancement, a number of nodes between 1/3 and 1/2 of the initial nodes are generated. A node $x$ that is to be expanded with a probability distribution function is selected. To expand configuration $x$, let each dof take a random value in an interval centered on its value at $x$ about 1/6 of range of the dof.

In the ACO, an ant is described as an agent that has the following characteristics:

- Lays a trail on edge ($i$, $j$) when moving between town $i$ and town $j$.
- Chooses to go to the town with the probability function of town distance and the amount of trail present on the connecting edge.
- To force ants to make legal tours, transitions to already visited towns are inhibited till a tour is completed.

Trail intensity $\tau$ is dependent on two parameters which are:

- $\rho$ that represents the evaporation of trail.
- $\delta$ quantity unit of length of trail substance (pheromone in real ants) laid on edge of $k$-th ant at a period of times.

Tabu list is an ant's data structure that memorizes the towns already visited up to a certain time and forbids ants to visit them again before the tour has been completed. Tabu list is emptied when tour is finished and the ant is free to choose its own way.

The transition probability of moving from $i$ to $j$ at certain times is dependent on four parameters:

- Trail intensity, $\tau$.
- Visibility, $\eta$.
- $\alpha$ and $\beta$ ( user's control parameters) which chooses the importance between trail versus visibility, i.e. transition probability is considering the visibility (closest town should be chosen) and trail intensity (if the edge has been passed by lots of ants, it is highly desirable).
- $N_i^k$, next city is allowed to be visited (feasible neighbourhood of ant $k$ when being at city $i$).

During the initialization phase the ants are positioned on different towns and initial values for trail intensity are set on edges. The first element of each ant tabu list is set to be equal to the starting town. Thereafter, every ant moves from town $i$ to town $j$ choosing the town to move to with a probability that is a function (parameters $\alpha$ and $\beta$) of two measures: the first is trail, that gives information about how many ants in the past have chosen that same edge. The second is visibility the closer a town the more desirable it

is. Each time an ant makes a move, the trail it leaves on edge $(i, j)$ is summed to trail on the same edge in the past. When the ant has moved, transition probabilities are computed using new trail values.

After a number (of city) moves (or a tour is complete) the tabu list of each ant will be full, the shortest path found by the number of ants is computed and memorized and all lists are emptied. This process is iterated until the tour counter reaches a maximum (user-defined) number of cycles or all ants make the same tour. Computation complexity can be expressed as a function of the number of ants, number of towns and number of cycles.

## IV. Ant Colony Robot Motion Planning

In this section, the ACO is proposed to be implemented for robot motion planning. The idea is taken from the combination of the ant behaviour of ACO and the roadmap characteristic of PRM.

The problem of existing multi-goal path planning is modelled in the hierarchical form. A roadmap (RM) is built by nodes and edges in C-space. The Single query Bi-directional Lazy algorithm is applied to find a path between the initial and goal configurations. A multi-goal problem is tackled by using the traveling salesman problem (TSP) solving approach.

- The basic program is (PRM multi-goal) :

RM $\rightarrow$ SBL $\rightarrow$ TSP

- The objective is to apply the swarm algorithm in the PRM and replace the PRM (if possible).
- The first stage is to replace PRM with foraging-ants (F-ANTS).

F-ANTS $\rightarrow$ SBL $\rightarrow$ TSP

- The second stage is to replace the SBL with trail-ants (T-ANTS).

F-ANTS $\rightarrow$ T-ANTS $\rightarrow$ TSP

- The third stage is to optimize the top level by applying ACO to TSP

F-ANTS $\rightarrow$ T-ANTS $\rightarrow$ ACO-TSP

In this approach, there is the global table which will record all ants, trails dropped, and the distances. There are two groups of ants, ants located in nest and ants located at food. Both locations have the same amount of ants. The start location determines the initial configuration of the robot, $s$, and the food location is the robot goal configuration, $g$. Each ant has a limited number of trails to be dropped.

In the first iteration T-ANTS is applied where an ant is released from the nest and moves randomly away from the nest for a distance of $d$. This distance in the robot space is determined by the configuration of all the robot joints from the initial configuration to the new randomly generated configuration. Then the path along the nest to the new configuration is checked for collisions as well as a new configuration being found.

If the location (configuration) is forbidden because of the existence of obstacles (or the configuration is colliding against the robot body) a random location is generated and it is checked for collisions again until it is in an allowed location (configuration free). The number of times collisions are checked is counted and recorded in the global table for the $k_s$-th ant ($k$-th ant of the nest, $s$) at the $q$ position (configuration), the trail is dropped at this location $q(k_s$-th) (configuration of robot at $q$ for the nest's $k$-th ant). If it reaches the food it will stop, if its distance is closer to the food (less than $d$) than the previous trail, the ant will move to the food, if it doesn't, the ant will move randomly and drop the trails after the same period of time before.

Each time the ant drops the trail, the global table is updated with information about $k$-th ant, its trail position, and the number of trails that particular ant has dropped. Ants will stop moving either if the food is reached, or the trail is finished.

When all ants from the nest (s-ants) have stopped, the global table is updated. The second iteration now is started from the food location, $g$. This stage is where the T-ANTS is applied. First the $k_g$-th ant will move randomly within a distance $d$ from food. Then it will check in the global table for any closest trail with less-attempts and within $d$ distance. The $k_g$-th ant will choose the trail if it satisfies the function of less-attempts and distance, or if none of them are in the $d$ range, the ant will move randomly to a new free-collision location and check the table again for the trails within the function of less-attempts and distance. An $l$ number of attempts are set for the attempts upper bound, restricting the ants from doing a mass number of attempts. If the $k_g$-th ant fails to get to the new location, the $k_g$-th ant will be terminated, and the $(k+1)_g$-th is released and does the same thing but on a same or different free-location from the nest.

Then, if the $(k+1)_g$-th ant managed to get to the new location the trail is released and the table is updated. The range within its new location is checked again for the existing trails and this will repeat until the $g$-ants find the nest. The $g$-ants might use, or not use, the same path from the $s$-ants but the paths connected from the food, $g$, to any of $s$-ants trail will definitely give the path to the food, $s$. However, the $g$-ants will get the shortest path by choosing the appropriate trails with the heuristic provided in the global table. Finally when all $g$-ants have made the journey to the nest, and the table is updated, the shortest path can be checked against the global table.

## V. Experimental Results

The preliminary experiments aimed to investigate reducing the size of the search trees in the existing roadmap in SBL-PRM in order to get to F-ANTS stage. The existing trees are grown from both start and goal configurations of the robot manipulator. After tree reduction, only one tree remains - grown from the start configuration. Reduced tree SBL-PRM is called 1-TREE planners.

There are four different cases in which one or two manipulators move from the starting to goal configurations. These cases were tested on the 1-TREE and the original SBL-PRM algorithms for comparison.

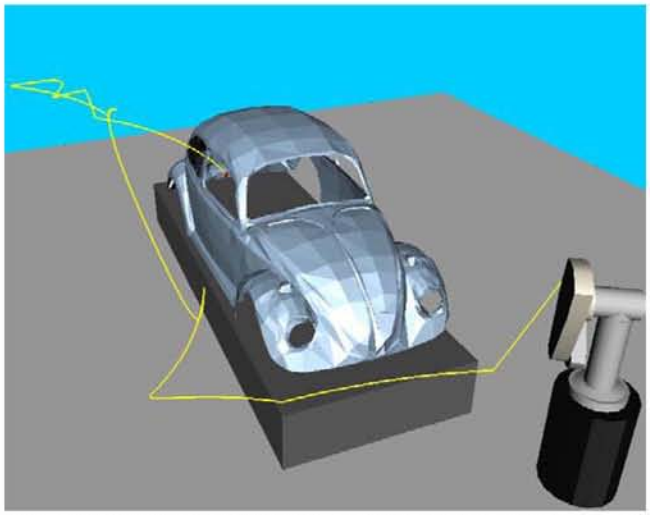

Fig.1 Case A, a PUMA 560 robot mounted on a mobile platform and a car on fixed platform as the obstacles.

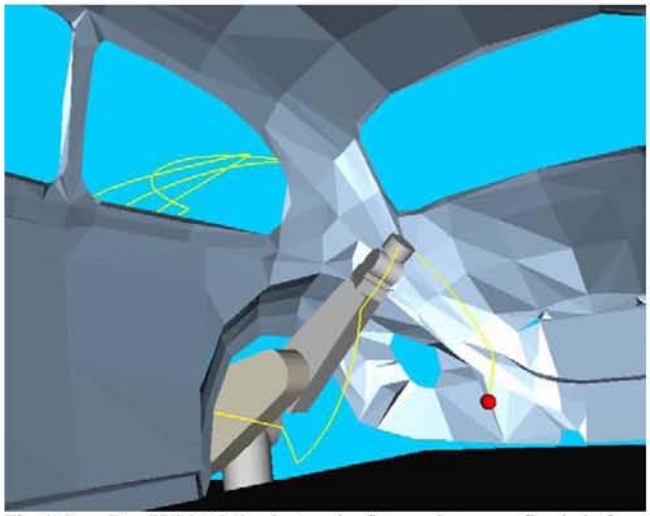

Fig. 2 Case B, a PUMA 560 robot on the floor and a car on fixed platform as the obstacles. This view is from the inner part of the car.

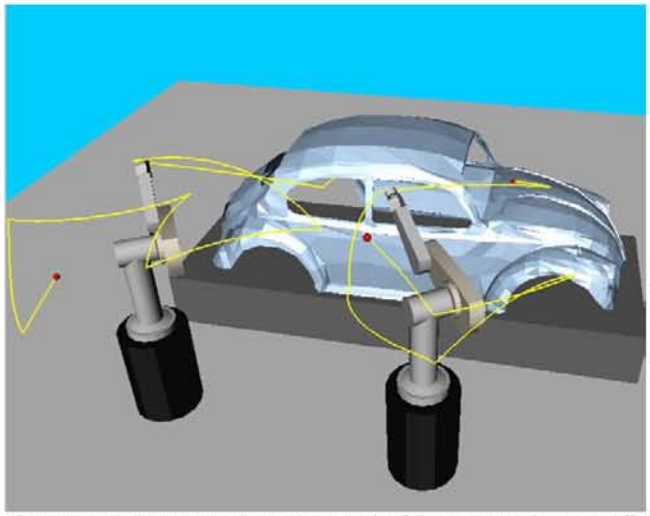

Fig.3 Case C, two PUMA 560 robots, each of them mounted on a mobile platform and a car on fixed platform as the obstacles.

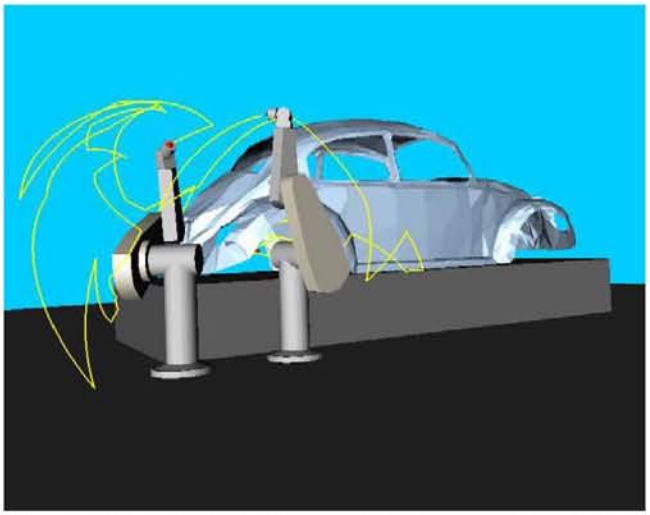

Fig.4 Case D, two PUMA 560 robots on the floor and a car on fixed platform as the obstacles.

TABLE I
NUMBER OF CONFIGURATIONS GENERATED FOR THE DIFFERENT CASES USING SBL-PRM AND1-TREE

| Case | Algorithm | Configurations | | |
|---|---|---|---|---|
| | | Average | Maximum | Minimum |
| A | SBL-PRM | 5 | 11 | 3 |
| | 1-TREE | 5 | 12 | 3 |
| B | SBL-PRM | 20 | 46 | 7 |
| | 1-TREE | 19 | 34 | 5 |
| C | SBL-PRM | 8 | 14 | 5 |
| | 1-TREE | 7 | 12 | 3 |
| D | SBL-PRM | 41 | 87 | 11 |
| | 1-TREE | 39 | 85 | 19 |

TABLE II
RUNNING TIME FOR EACH CASE

| Case | Algorithm | Running time (s) | | |
|---|---|---|---|---|
| | | Average | Maximum | Minimum |
| A | SBL-PRM | 0.1409 | 0.2030 | 0.1100 |
| | 1-TREE | 0.1459 | 0.2500 | 0.1250 |
| B | SBL-PRM | 3.2420 | 11.8750 | 0.1710 |
| | 1-TREE | 3.4169 | 12.1880 | 0.1560 |
| C | SBL-PRM | 0.2178 | 0.4060 | 0.1400 |
| | 1-TREE | 0.2284 | 0.4690 | 0.1250 |
| D | SBL-PRM | 13.8453 | 70.0940 | 0.9380 |
| | 1-TREE | 14.0928 | 69.9530 | 1.2190 |

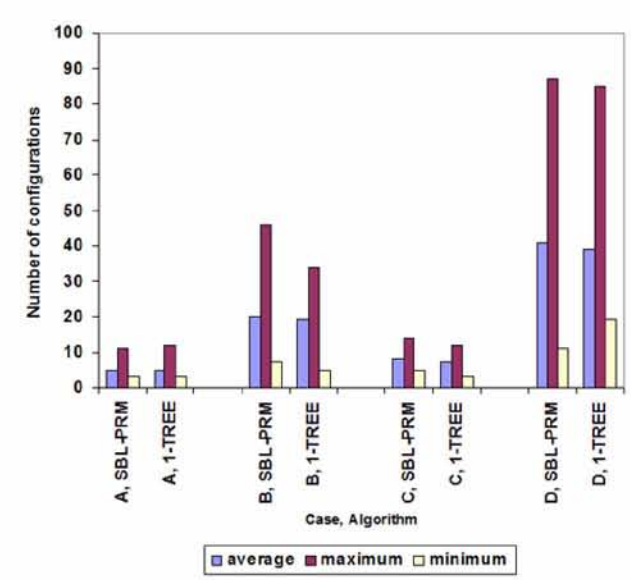


Fig. 5. The comparisons of generated configurations. The average, maximum, and minimum numbers of configurations are shown for each case and the algorithm applied.

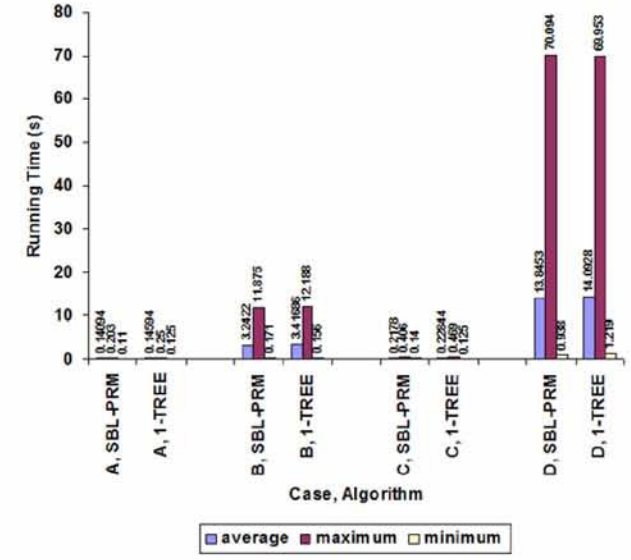


Fig. 6. The comparisons of running time of all cases

In each of the experiments, the number of configurations was calculated and the running time was recorded. The figures in Table 1 and Table 2 shown were obtained on a 2.60 GHz Pentium4 processor with 512 MB of main memory running Microsoft Visual C++ 6.0. The distance threshold $\rho$ was set to 0.15 and the resolution $\varepsilon$ to 0.012. The parameter $\rho$ is used to determine how far the distance of two configurations is, if their distance is less than $\rho$, both of them are considered close. A path in configuration space between two configurations is considered collision free if a series of points on the path in every successive point are closer apart than some $\varepsilon$.

As can be seen in Table 1 and Fig. 5, 1-TREE has shown a smaller number of configurations generated on the robot(s) path than the configurations generated by SBL-PRM. For the situation of applying the algorithms for the robot mounted on the platform, Fig. 1 and Fig. 2, the average path configurations generated between start and goal configurations are less than ten. However, when the robot(s) are fixed on the floor (Fig. 2 and Fig. 4), the averages are 19 and 39 configurations for one robot and two robots respectively. The reason is the robot(s) have to manage with the limited space in their own space instead of moving away from the object as in the mobile-platform-mounted-robot case (Fig. 1 and Fig. 3). This reason also affected the computational time for both algorithms as shown in Table 2 and Fig. 6. The computational times for both algorithms are very close although 1-TREE is slightly slower than SBL-PRM. This is probably because the 1-TREE is the only at the preliminary stage of its application on transition from SBL-PRM.

## VI. Conclusions and Future Plans

The benchmark against this new implementation is the SBL-PRM. It has been tested against SBL-PRM to determine their relative performances. The performance metrics are time of planning and the number of intermediate points (configurations) between two queries (start and goal). The implementation of the modified SBL-PRM to 1-TREE (unidirectional) reduces the number of intermediate configurations. Although the time of planning is slightly greater than the original planner, the difference is relatively small. This can be improved by applying the ACO to replace the PRM.

The future plan of this research is to fully implement the ACO in robot motion planning. It will contribute to the domain of intelligence motion planning. The following is the future plan for 1-TREE. Implementation of F-ANTS – the network or roadmap will be replaced by introducing a new network built by the foraging-ants (F-ANTS). Implementation of T-ANTS – each ant (agent) has the capability of releasing the trail or virtual pheromone during the F-ANTS stage, the trail will be tracked down by trail-ants in order to reduce the distance travel to make the path shorter. Implementation of ACO-TSP – the benchmark program has a solver for multi-goal planning. A complete F-ANTS and T-ANTS will make use of the existing ACO-TSP algorithm to overcome the multi-goal planning problem.

## References

[1] R. Bohlin, and L. E. Kavraki. "Path planning using lazy PRM," *IEEE Proc. Int. Conf. on Robotics and Automation,* pp. 521 – 528, 2000.

[2] M. Dorigo and L. M. Gambardella. Ant Colonies for the Traveling Salesman Problem. Technical Report TR/IRIDIA/1996-3, IRIDIA, Université Libre de Bruxelles, 1996.

[3] M. Dorigo, and L. M. Gambardella. "Ant Colony System: A Cooperative Learning Approach to the Traveling Salesman Problem," *IEEE Transactions in Evolutionary Computation,* vol. 1 (1), pp. 53 – 56, 1997.

[4] M.Dorigo, V. Maniezzo and A. Colorni, "Positive feedback as a search strategy," Technical report 91-016 revised. Dipartimento di Elettronica, Politecnico di Milano, Milan, 1991.

[5] M.Dorigo, V. Maniezzo and A. Colorni, "The Ant System: Optimization by a colony of cooperating agents," *IEEE Trans. on Systems, Man and Cybernetics-Part B,* vol. 26 (1), pp.1 – 13, 1996.

[6] L. Kavraki and Latombe, J.-C., "Randomized preprocessing of configuration space for fast path planning," *IEEE Proc. Int. Conf. of* Kavraki, L. and Latombe, J.-C., "Randomized preprocessing of configuration space for fast path planning: Articulated robot," *IEEE/RSJ/GI Proc. Int. Conf. Intelligent Robots and Systems,* pp. 1764 – 1771, 1994.

[7] L. E. Kavraki. *Random Networks in Configuration Space For Fast*

*Path Planning.* PhD thesis, Stanford University, CA, 1995.

[8] J. -C. Latombe. *Robot Motion Planning*. Kluwer, Boston, MA, 1991.

[9] V. Maniezzo, A. Colorni and M. Dorigo, "The Ant System Applied to the Quadratic Assignment Problem," Technical Report IRIDIA/94-28, Université Libre de Bruxelles, Belgium.

[10] G. Liu, T. Li, Y. Peng, and X. Hou, "The Ant Algorithm for Solving Robot Path Planning Problem," *Int. Conf. on Information Technology and Applications, ICITA,* vol. 2, pp. 25 – 27, 2005.

[11] M. Saha, G. Sanchez-Ante, J.-C. Latombe "Planning multi-goal tours for robot arms,", *IEEE Proc. Int. Conf. on Robotics and Automation,* pp. 3797 – 3803, 2003.

[12] G. Sanchez, and J.-C. Latombe, "Using PRM planner to compare centralized and decoupled planning for multi-robot systems," *IEEE Proc. Int. Conf. on Robotics. and Automation,* pp. 2112 -2119, 2002.

[13] G. Sanchez, and J.-C. Latombe, "On delaying collision checking in PRM planning: Application to multi-robot coordination," *Int. J. of Robotics Research,* vol. 21(1), pp. 5 – 26, 2002.

[14] G. Sanchez-Ante. *Single-Query Bi-Directional Motion Planning With Lazy Collision Checking.* PhD thesis. Instituto Tecnologico Y De Estudios Superiores De Monterrey-Campus Cuernavaca, Mexico, 2002.

[15] T. Stützle and M. Dorigo, "ACO Algorithms for the Quadratic Assignment Problem," Technical Report IRIDIA/99-2 Université Libre de Bruxelles, Belgium.

[16] G. Liu, T. Li, Y. Peng, and X. Hou, "The Ant Algorithm for Solving Robot Path Planning Problem," *Int. Conf. on Information Technology and Applications, ICITA,* vol. 2, pp. 25 – 27, 2005.

[17] X. Fan, X. Luo, S. Yi, S. Yang, and H. Zhang Robotics, "Optimal path planning for mobile robots based on intensified ant colony optimization algorithm," *Proc. IEEE Int. Conf. on Intelligent Systems and Signal Processing.* vol. 1, pp. 131 – 136, 2003.