Sampling based Mission Planning for Multiple Robots

Rahul Kala, *Member, IEEE*
Robotics and Artificial Intelligence Laboratory,
Indian Institute of Information Technology, Allahabad,
Deoghat, Jhalwa, Allahabad, INDIA
rkala001@gmail.com, http://rkala.in/

***Abstract*— Robots of the future would be sophisticated enough to do all kinds of tasks for the human master, while operating a team of robots would ensure that the tasks are done as efficiently as possible. This paper presents a solution to a very common mission wherein a team of robots needs to visit a number of mission sights and carry some operation there. The sights may have their own preference of robots. The real world workspace is first converted into a roadmap using the Probabilistic Roadmap algorithm. The adopted technique for roadmap generation uses a hybrid of narrow corridor sampler, obstacle based sampler and uniform sampler; with hybrid edge connection technique that aims to first connect disconnected roadmaps and then introduces redundant cycles and also ensures a good coverage of the configuration space. The roadmap is used to compute a cost matrix between every mission sight. This cost matrix is used by an optimization algorithm which deputes different mission sights to different robots as well as dictates the order of visit of the missions by each robot. Local optimization is used to quickly compute the optimal plan. The navigation of the robots is done using a Fuzzy Logic based navigator. Experiments show that the robots are able to coordinate with each other and complete the mission as a team very effectively.**

***Keywords—Evolutionary Robotics, Robot Mission Planning, Fuzzy navigation, Mission Optimization, Sampling based Robotics*.**

I. Introduction

It is increasingly becoming likely to have robots in home and office environments doing many important tasks like carrying files, conveying messages, inspecting sights, getting the things that the human master needs, picking up items in a shopping store, cooking and serving dishes, and a complex combination of these. Multiple robots [1] enable more efficient operation and timely completion of the required task. The problem of motion planning [2, 3] deals with all kinds of decision making. The classical problem of motion planning deals with a solution to the problem "Go from *A* to *B* avoiding obstacles" and helps a robot travel the workspace for carrying the required task. A common robot workspace may be occupied by different kinds of robots, humans and other static or dynamic obstacles. The problem of multi-robot motion planning deals with the problem of devising a collision-free plan for each of the robots from its source to its goal. The plan must make sure that neither do the robots collide with each other, nor with any other static or dynamic obstacle. It is commonly seen that a robot may have to sacrifice its optimal path to facilitate some other robot take a near-optimal path. This is called as cooperation [4] and is offered as long as it makes the overall plan optimal.

Modern day problems are very complex and cannot be easily decomposed into the classical motion planning problem. The robots are asked to execute complex missions. The problem of mission planning makes the robots plan and execute such higher level missions. A common approach towards mission planning is to use Temporal Logic to specify the mission. Temporal Logic [5] is built over logic based systems and enables representation of temporal relations between axioms. Typically the temporal logic mission specification is converted into an automaton, a transition system of the robot's workspace is used to compute valid transitions, which is used to search for a feasible strategy. Lahijanian et al. [6] added the element of probability in temporal based search to account for the uncertainty in the actions of the mobile robots, which was computed using Reinforcement Learning. In contrast Kress-Gazit [7] built a roadmap identifying the different regions of interests and their transitions. The roadmap can then be used for the temporal logic based navigation of the robot. The construction of roadmap may be computationally expensive, however once done, the environment is reduced to a small abstraction making it easy to plan a team of robots.

The advantages of using temporal logic based mission planning is that nearly all types of missions can be formulated and solved using the grammar of temporal logic. However the search is very slow as all types of mission solutions are generated and assessed. Hence there is a lot of interest towards identifying different kinds of commonly found missions and to develop solutions for the same. Missions are highly complicated and therefore may require a large number of robots to work for the mission in order to aim a timely mission execution. The centralized solutions result are extremely time consuming and the approach cannot be used for a large number of robots. The decentralized solutions however may not be able to compute a solution as the actions of the robot highly depend upon each other. It is hence important to identify the different kinds of missions that robots of the future would be subjected to, and to further make robust and scalable algorithms for solving those mission with a team of robots.

In this paper a specific type of mission is considered, wherein a team of robot is asked to visit a set of mission sights to carry some operation. However some mission sights may be serviceable by only some of the robots. Consider the example that you need a list of items from a supermarket, which already has a team of robots to pick and get them to you. Similarly consider that an invitation needs to be sent to all employees of an office by robotic peons. This is a very practical and important mission problem to study.

In a related work, Xidias et al. [8] used a centralized Genetic Algorithm (GA) based technique to make a group of autonomous vehicles cover a set of landmarks and return to their positions. Being a centralized solution, its utility is restricted to a few robots only. On the other hand Sud et al. [9] showed navigation of a very large number of robots in a densely occupied navigation scenario. The navigation was made using an adaptive roadmap at the coarser level, wherein the roadmap could adapt to the changes in the environment. The work can simulate a large number of robots, but cannot take higher order specifications. Similarly Kala [10] proposed a mechanism to grow a roadmap iteratively, first by connecting the different regions of interest, and then by adding valuable edges to reduce the path cost. The roadmap was used for the problem of multi-robot motion planning.

In this paper the problem of mission planning is solved by using a team of robots. The first problem is that the configuration space is continuous and too large The configuration space is thus sampled out to generate a roadmap. The roadmap facilitates computation of cost and path between every mission sight. Since the number of vertices and edges in the roadmap are small, the problem is easily solvable in small computation times. The cost matrix is used by an optimization algorithm which makes the mission plans, specifying the sights and the order of visit of sights by each robot. The local navigation is done using a fuzzy logic based navigator which caters to the presence of dynamic obstacles and other robots.

II. Problem Description and Approach

Let L={L_i} be the set of mission sights which a robot must visit in order to perform the necessary operation desired by the user. Let ρ={ρ_j} be the set of robots available to perform the mission. Let $\zeta_i = \left[\zeta_i^j\right]$ be the set of robots which can service the mission sight L_i. Let $M_i = \left(M_i^j\right)$ be the mission of the robot ρ_i consisting of an ordered list of mission sights that the robot must visit. Every mission sight must be serviced by at least one robot, i.e. $\cup_{i,j} M_i^j = L$. Further no mission sight must be serviced by two robots or twice by the same robot, i.e. $M_i^j \neq M_k^l \forall \neg(i = k \wedge j = l)$. Also every mission sight must be serviced by a robot in its preference list, i.e. $\rho_i \in \zeta\left(M_i^j\right)$.

Let C_{free}^{static} denote the obstacle-free configuration space of the robot, i.e. $C_{free}^{static} = C \setminus C_{obs}^{static}$, where C is the configuration space of the robot and C_{obs}^{static} is the obstacle prone configuration space with all configurations of the robot which are prone to collision due to the static obstacles. Dynamic obstacles are not considered here. Let $\tau_{a,b}$:[0,1]→ C_{free}^{static} denote the trajectory from a to b passing through all collision-free points in between, i.e. $\tau_{a,b}(0)=a$, $\tau_{a,b}(1)=b$, $\tau_{a,b}(s) \in C_{free}^{static}$, $0 \leq s \leq 1$. Here the objective is taken simply as the length of the trajectory, i.e. $\|\tau_{a,b}\|$. The trajectory may even be parameterized by time, thus also giving the velocity profile. The trajectory $\tau_{a,b}:[t_1,t_2] \rightarrow C_{free}^{static}$ represents the trajectory of a robot from a at time t_1 to b at time t_2.

Let χ: $\{L \cup S\} \times L \rightarrow \tau$ denote the trajectory from every mission sight or a robot source to every other mission sight, i.e. $\chi(L_i, L_j)$ denotes the trajectory from mission sight L_i to mission sight L_j and $\chi(S_i, L_j)$ denotes the trajectory from source S_i to mission sight L_j. The cost matrix $\|\chi\|$: $\{L \cup S\} \times L \rightarrow R+$ denotes the cost from every mission sight or a robot source to every other mission sight i.e. $\|\chi(L_i, L_j)\|$ denotes the cost of trajectory from mission sight L_i to mission sight L_j

Each robot starts from its source, S_i and then visits all the places listed in its mission specification in the same order. The trajectory of the robot may hence be given by (1).

$$\tau_i = \tau\left(S_i, M_i^1\right) + \sum_{j=1}^{|M_i|-1} \tau\left(M_i^j, M_i^{j+1}\right) \tag{1}$$

Here '+' denotes concatenation of the trajectories. For the trajectory to be feasible, it is essential that no two robots collide which each other, given by (2).

$$R(\tau_i(t)) \cap R(\tau_j(t)) = \emptyset \forall t, i \neq j \tag{2}$$

Here $R()$ is the function which maps a robot from its configuration space to the volume occupied in the workspace space. The cost of the mission of a single robot is given by (3), while the cost of the overall mission is given by (4).

$$\|M_i\| = \left\|\tau\left(S_i, M_i^1\right)\right\| + \sum_{j=1}^{|M_i|-1} \left\|\tau\left(M_i^j, M_i^{j+1}\right)\right\| \tag{3}$$

$$\|M\| = \Sigma_i \|M_i\| \tag{4}$$

The aim of the approach is to find an algorithm which computes a mission with the least mission cost, i.e.

$M^* = \mathrm{argmin}_M \|M\|$. The approach used to solve the problem is described in Fig. 1. The approach first builds a roadmap $G(V,E)$ in the free configuration space C_{free}^{static}, considering only the static obstacles. The roadmap is used to compute the path cost matrix χ considering only the static obstacles. Since the number of vertices and edges are kept small, this can be quickly computed. This matrix is used by an optimization algorithm to give a mission for each robot M_i^{static}, where the suffix static means that only static obstacles were considered in the formulation of the mission**.** This mission is used by a fuzzy navigator to traverse the robots considering the presence of the other robots, previously unseen obstacles and other dynamic obstacles.

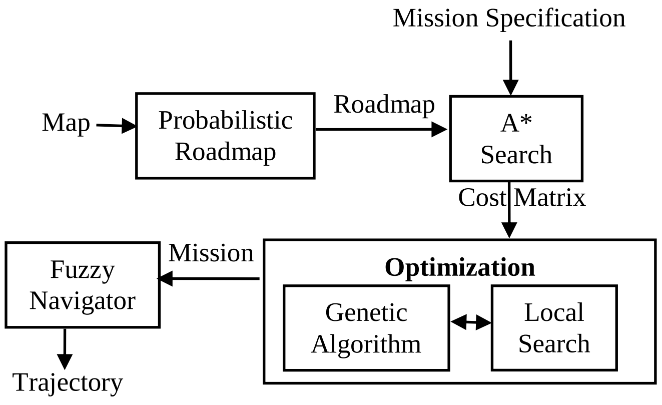


Fig. 1. Overview of the algorithm

III. Roadmap Generation

The first task is to sample out the configuration space and make a roadmap using the Probabilistic Roadmap (PRM) algorithm. It is assumed that all robots have a common configuration space. Since primarily the mobility is of concern, the robots are assumed to be operating in a common configuration space of SE(2). This enables making one roadmap and subsequently one cost matrix for the entire problem, rather than repeating this for all robots which can be computationally expensive.

A. Generating Vertices

The first task is to generate the vertices *V* of the roadmap. A hybrid sampling technique is used for the task. Three sampling techniques are used. The first technique is narrow corridor sampling, which generates samples inside a narrow corridor. Narrow corridors are the hardest to sample out and are a source of concern for any sampling based motion planning technique. First uniform sampling is used and the samples generated inside the obstacles are retained, which are promoted to the obstacle free configuration space at the obstacle boundary. A sample further apart at a small distance in the same direction is generated. If the new sample is collision-prone, it means that a transition from obstacle-prone to obstacle-free configuration space was recorded by the traversal of the sample, and further that a transition from obstacle-free to obstacle-prone configuration space was recorded as a result of this sample. This as per the definition of bridge-test [11] is a narrow corridor. A small local search is used to take the vertex in the mid-point of the narrow corridor. The sampling technique uses a mixture of obstacle-based sampling and bridge test to sample out narrow corridors.

The second strategy is obstacle-based sampling strategy which attempts to generate samples near the obstacle boundary. Areas near the obstacles mark a good candidate for the robot to turn and avoid the obstacle, and therefore should be sampled out nicely. The sampler generates samples inside the obstacles and promotes them to the obstacle-free configuration space or the obstacle boundary. The sample is further promoted by a small distance to maintain a required clearance which helps in better connecting the sample to the roadmap and also enables the robots to maintain a safety distance.

The last strategy is uniform sampling strategy wherein the samples are uniformly generated in the collision-free configuration space. The samples are only admitted when they maintain a respectable distance between each other. This limits the number of vertices in the roadmap.

B. Generating Edges

The next task is to compute the edges *E* that connect the initial set of vertices *V* of the roadmap. Most of the vertices of the roadmap may be easily connectable to each other by straight lines. Thus the initial basic strategy for edge connection attempts connection between every vertex *u* and the *k*-closest vertices. If a straight line connection is possible, the corresponding edge is added. If however a straight line connection is not possible, an additional vertex is placed near the obstacle which makes the connection impossible. The region around the obstacle marks promising area and growing a small roadmap abound the area can facilitate connectivity. This results in a fairly connected roadmap.

The roadmap so produced may still be in the form of multiple disconnected roadmaps. Let κ(*a*) denote the vertices to which the vertex *a* is connected directly or indirectly, i.e. all vertices of the set κ(*a*) are connected to each other by a path and no path between any vertex of κ(*a*) exists with any vertex not a member of κ(*a*). Let κ denote the set of all such disconnected roadmaps.

Four strategies are used to add more edges, maybe with the addition of more vertices in the process. The first strategy attempts to connect two disconnected roadmaps κ(*a*) and κ(*b*). The first roadmap κ(*a*) is selected randomly from κ. The disconnected roadmaps in the vicinity of κ(*a*) are more likely to be easily connected, as compared to roadmaps which are very far off. The distance between the disconnected roadmaps is taken as the least distance between the vertices of the two roadmaps, given by (5).

$$|\kappa(a),\kappa(b)| = \min_{x\in\kappa(a),\,y\in\kappa(b)}\|x-y\| \tag{5}$$

The choice of the second roadmap is done using a roulette wheel selection with the selection possibility of κ(*b*) given by the inverse of the distance between κ(*a*) and κ(*b*). The attempt is to connect the vertices *x* and *y* which record the minimum distance. A vertex is generated at a small distance from *x*, producing a new bridging vertex. If the bridging vertex can be connected to *y*, the connections are made and the roadmaps are merged. Otherwise the bridged vertex (with connections to *k* nearest neighbors if feasible) is retailed for future attempts. Attempts to connect pair $\kappa(a)$ and $\kappa(b)$ can only be made if the number of failed connection attempts do not cross a modest threshold.

The first strategy is only useful once the roadmap is not fully connected. Subsequently attempts may still be given to add redundant cycles and redundant edges to get better estimate of the costs. The second strategy attempts to grow the roadmap covering in the configuration space, exploring newer areas. The leaf nodes are identified, which have a unitary connectivity, and they are expanded in a random direction. This gives an Expansive Spaces Trees (EST) style expansion of the roadmap.

The third connection strategy attempts to spread the roadmap in the entire configuration space, especially areas which may not be represented so far. A lower dimensional memory hash is made, which is fitted with cells. The occupancy of every cell and the volume of collision-free configuration space can be computed. The ratio of the occupancy to the volume is the desirability ratio of a cell. A cell is selected by Roulette Wheel selection scheme with the possibility of selection given by the inverse of the desirability ratio. A sample is randomly taken in the cell, and is used to pull the roadmap towards itself. The nearest point in the roadmap is taken and extended towards this sample in a Rapidly-exploring Random Trees (RRT) style expansion [12]. The last connection strategy is a random connection strategy which makes the roadmap grow in an RRT-style expansion. A random sample is taken. If the sample is collision-prone, it is promoted to the obstacle-free configuration space. The nearest sample in the roadmap is chosen and extended towards this sample.

C. Computing Cost Matrix

The next task is to compute the trajectory between all mission sights and sources and the associated cost. Since it is assumed that the configuration space of all robots is the same, a single cost matrix needs to be computed. Computation of this matrix is a classical motion planning problem, repeated for all pairs of mission sights and sources. Thus the number of vertices and edges in the roadmap need to be very small so that the paths can be computed without much computational overhead. The computation still does not account for the presence of dynamic obstacles and other robots to make the computation easier.

To compute $\chi(L_i,L_j)$, where L_i may be either of the mission sights or source, a graph search over the roadmap is used. First L_i and L_j are temporarily added as additional vertices in the roadmap $G(V,E)$. A straight line connectivity to the nearest k-neighbors is checked. This gives an extended roadmap with L_i and L_j as vertices. A* algorithm is used to compute the shortest path between L_i and L_j. Since the roadmap maintains all straight line connections, the distance between L_i and L_j is taken as the weight of the edge. The historic cost is the summation of all step costs. The heuristic cost is taken as the straight line distance to the goal L_j. The algorithm is terminated on reaching the goal. The algorithm gives both the trajectory $\chi(L_i,L_j)$ and the associated cost $\|\chi(L_i,L_j)\|$.

IV. Multi-Robot Mission Assignment

The overall problem of mission planning needs to be solved in a centralized manner considering the joint plans of all robots combined. The centralized approaches are extremely time consuming and thus the computation is restricted to static obstacles only which were fully dealt with in Sections II and III. Here the plans need to be made considering the cost matrix alone, whose dimensionality is limited to the number of robots and the number of mission sights.

A. Objective Function

The objective is to find the best mission plan $M^*_{static} = \text{argmin}_{M^{static}} \left\|M^{static}\right\|$. The mission and trajectory so generated may not be feasible due to inter-robot collisions and collisions with other dynamic obstacles. The assumption is that such collisions can be eliminated by minor changes in the trajectories of the robots, thus making the dynamic trajectories nearly same to the trajectories considering the static obstacles only, thus making the overall approach near-optimal. This is widely seen in human mission planning as well. While planning for picking a number of items in home one usually does not consider the presence of other people and avoids them while executing the plan. Similarly a supervisor while deputing work to employees rarely considers the influences of other people in the environment. The objective function, to be minimized, is thus given by (6) and (7) which clearly resemble (3) and (4).

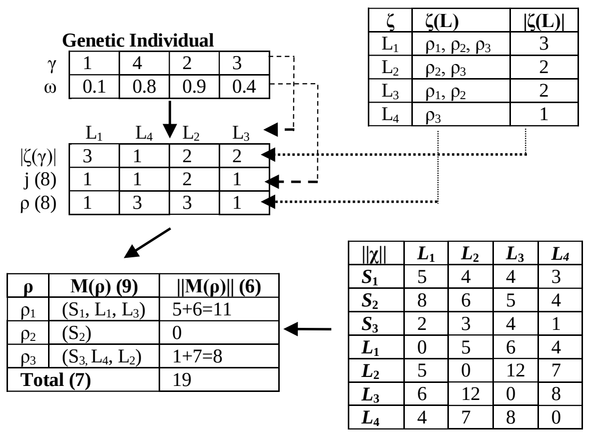
$$\left\|M_i^{static}\right\| = \left\|\chi\left(S_i, M_i^1\right)\right\| + \sum_{j=1}^{|M_i|-1} \left\|\chi\left(M_i^j, M_i^{j+1}\right)\right\| \tag{6}$$

$$\left\|M^{static}\right\| = \sum_i \left\|M_i^{static}\right\| \tag{7}$$

B. Genetic Algorihtm

The genotype consists of two parts <γ, ω>, each of length $|L|$, the number of mission sights. The genotype is shown in Fig. 2. The first part γ is an ordered list of mission sights to be visited, $1 \le \gamma_i \le |L|$, $\gamma_i \ne \gamma_j \; \forall \; i \ne j$, $1 \le i \le |L|$. The second part (ω) is a specification of the robot corresponding to the mission sight. There is a hard constraint on the robot that every mission sight must be serviced by a robot in its preference list $\zeta(\gamma_i)$. A representation needs to be adopted such that for any assignment, this constraint is always valid. This saves the algorithm from doing constraint checking and thus significantly eases the optimization. ω_i is a representative of the robot vising the mission sight γ_i. There are a total of $|\zeta(\gamma_i)|$ robots available, any of which may be made to visit the mission sight γ_i. Let ω_i be a real number, $0<\omega_i<1$, $1 \le i \le |L|$. A linear scale from 0 to 1 is divided into non-overlapping regions, each of length $1/|\zeta(\gamma_i)|$ and each pointing to one of the valid robots which can visit the sight γ_i. Let $\zeta_j(\gamma_i)$ denote the j^{th} item of the set $\zeta(\gamma_i)$. The robot visiting the sight γ_i will hence be given by (8).

$$\rho_i = \zeta_j(\gamma_i),\ j = ceil(\omega_i \times |\zeta(\gamma_i)|) \qquad (8)$$



γ	1	4	2	3
ω	0.1	0.8	0.9	0.4

	L₁	L₄	L₂	L₃
$\|\zeta(\gamma)\|$	3	1	2	2
j (8)	1	1	2	1
ρ (8)	1	3	3	1

ζ	ζ(L)	|ζ(L)|
L_1	ρ_1, ρ_2, ρ_3	3
L_2	ρ_2, ρ_3	2
L_3	ρ_1, ρ_2	2
L_4	ρ_3	1

ρ	M(ρ) (9)	||M(ρ)|| (6)
ρ_1	(S_1, L_1, L_3)	5+6=11
ρ_2	(S_2)	0
ρ_3	(S_3, L_4, L_2)	1+7=8
Total (7)	19	

||χ||	L_1	L_2	L_3	L_4
S_1	5	4	4	3
S_2	8	6	5	4
S_3	2	3	4	1
L_1	0	5	6	4
L_2	5	0	12	7
L_3	6	12	0	8
L_4	4	7	8	0

Fig. 2. Genetic Algorithm Individual Representation

The phenotype is the mission for each robot M_i^{static}, which can be obtained simply from the genotype by (starting from the source) noting all the mission sights in order visited by the robot, given by (9). Here '+' is the ordered list append operator.

$$M_i^{static} = S_i + \sum_{j=1,\rho_j=i}^{|L|} \gamma_j \qquad (9)$$

For optimization, a population pool of such individuals is taken. The initial population pool is generated randomly. The optimization of γ is a conventional combinatorial optimization problem, while the optimization of ω is a conventional parametric optimization problem. Accordingly mutation and crossover operations are applied to generate the next generation of individuals. The best individual found in the search process is returned as the solution.

C. Local Search

The GA solution is assisted with a local search (LS) to ease the optimization in a master-slave memetic architecture. Every individual of the master GA is subjected to a few iterations of LS. The LS places the individuals in better areas in the vicinity of the fitness landscape, thus giving better indications to the master GA about the goodness of the solution.

The LS is applied for every individual, which is a <γ, ω> pair. The LS is different for the mission sight ordering part γ and the robot assignment part ω. For γ, two random genes are chosen and swapped. For ω, a random gene is taken and the robot corresponding to it is changed to a random robot. If the generated solution is better than the original solution, the better solution is retained. The optimization of both γ and ω takes place with the same probability. At every iteration one of these is altered to explore the possibility of a better solution.

V. Navigation Using Fuzzy Logic

The mission generated is a specification of the order of visit of mission sights for every robot. The actual navigation is done using a Fuzzy Inference System. The system accounts for new obstacles that may have been previously unseen, dynamic obstacles and other robots operating in the workspace. The fuzzy system used here is taken from the author's previous work [13].

A. Hybrid Deliberative and Reactive Navigation

Each robot needs to visit the mission sights, one after the other, while starting from the source. Other robots are dynamically avoided by the fuzzy navigation system, if they happen to come on the way. This reduces the problem to navigation between every consecutive pair of nodes $\langle M_i^j, M_i^{j+1}\rangle$ in the robot's mission. The navigation may be done using the pre-computed path $\chi(M_i^j, M_i^{j+1})$, adapting it to suit the dynamic environment and other moving robots, or using the fuzzy based navigation system.

The fuzzy based navigation system is reactive in nature. Reactive systems are very fast to react to new, unseen and dynamic obstacles, however they often get trapped. They lack optimality. The deliberative algorithms on the contrary are near-optimal and near-complete, however take very long to compute and can therefore rarely handle dynamic obstacles or changes in the working scenario. The roadmap constructed using the PRM and hence the path is not smooth as well, and hence un-navigable by the robot.

Hence a hybrid system is used. The path $\chi(M_i^j, M_i^{j+1})$ computed by the PRM is deliberative in nature and the same is used as a guideline for the robot. The deliberative path consists of a set of points one after the other. At any instance of time, the robot would be approximately around the deliberative path. A point in $\chi(M_i^j, M_i^{j+1})$ is chosen that is at least β distance away from the current position of the robot [14]. This is given as a goal for the fuzzy logic system. As soon as the robot reaches near its immediate goal, the goal is chanced to a furtherer point in $\chi(M_i^j, M_i^{j+1})$. In this manner by constantly changing the goal of the fuzzy inference system to a point in the deliberative path, the robot is made to move while avoiding other robots and dynamic obstacles. The change of goal does not happen when the immediate goal is the mission sight M_i^{j+1}, which the robot pursued to reach. On reaching the mission sight M_i^{j+1}, the robot is guided by the next deliberative path $\chi\left(M_i^j, M_i^{j+1}\right)$. The robot stops upon reaching the final mission sight.

B. Fuzzy Based Reactive Navigation

The fuzzy system is a reactive system that works in a decentralized mode for each robot. The system assesses the current situation and decides the best immediate move to be taken by the robot. On reaching the final mission sight, the navigation stops. The fuzzy system is so trained such that on their way if two robots face each other and are likely to collide, the fuzzy system gives opposing corrections to make the robots escape the collision.

The first problem in the design of a fuzzy system for robot navigation is the inputs. The inputs must be rich enough to represent the operational scenario, while be less in number to enable easy construction of the fuzzy system. A total of 10 fuzzy inputs are taken. Six inputs are for obstacle avoidance and goal seeking behaviors. These include the distances from the obstacles at front and front-diagonal directions, angular deviation, distance to goal, and preferred turn (left or right) to avoid the obstacle directly ahead. The last input is a label input which is heuristically set based on the guidelines presented in [13].

Four inputs are additionally taken to cater to multi-robot coordination mechanisms, enabling the robots avoid each other. All the robots are projected to travel by the current speed in the current direction till the projected travel plan is feasible. The derived inputs include least distance from any robot directly ahead, least distance from any robot in the front left quadrant, least distance from any robot in the front right quadrant and preferred turn (left or right) to overcome the robot directly ahead. The last input is a label input which is set using heuristics presented in [13].

Fuzzy rules are made so that the robot tries to follow the deliberate path as closely as possible, avoids obstacles, mutually avoids the other robots and maintains a respectable clearance from all obstacles and robots. The fuzzy system gives as output the steering movements to make. The speed is controlled independently by assessing the obstacles in front and computing the maximum safe speed to disallow any collisions and to further keep respectable safety distance.

VI. RESULTS

A. Simulations

The approach was tested by simulations. In order to understand the working of the algorithm from an experimental view, first a very simple example is taken. A simple map is taken with a lot of wide open spaces. Even though wide open spaces make the task of roadmap construction very simple, the spaces result in multiple robots near enough to compete for a mission sight and thus complicate the optimization. Further they result in greater possibilities of robots colliding with each other. Only four robots are taken to better understand the algorithm. The location of the robots is kept as wide as possible, that is the four corners of the map. 15 mission sights are chosen at random. The scenario is shown in Fig. 3(a). The color of the mission sight denotes the robot which must cater to the mission sight. Empty circles indicate that any robot can cater to that mission sight.

The first task is generation of the roadmap, which for the scenario is shown in Fig. 3(b). The roadmap is used to compute the cost matrix which is used by the optimization algorithm. The path generated by the optimization algorithm is shown in Fig. 3(c). The actual path traced by the robots is using Fuzzy navigator is shown in Fig. 3(d). The results are also available as a video at [15]. It must be noted that one of the robots was not used in the task, which can often happen with such problems.

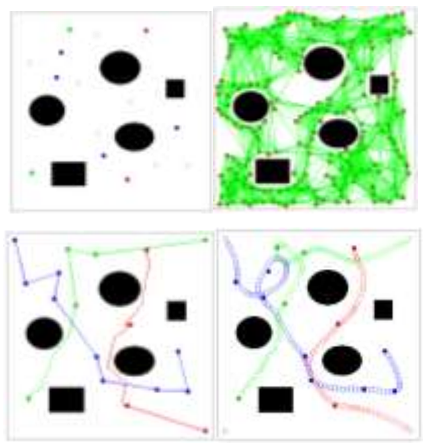


Fig. 3. Results for scenario 1

Based on the lines of simple scenario 1, the simulations are repeated for more number of robots and more mission sights. A map is taken which can accommodate and create option of interaction between numerous robots. The number of robots are taken as 6, generated at the extreme ends of the maps. The number of mission sights are increased to 25. The results are shown in Fig. 4. Due to open spaces and a large number of robots, a congestion is created which necessitates the robots to avoid each other while following their mission. The results are best visualized in the video given at [15].

Subsequently, in the last scenario, the experimentation is done with a mixed preferentiality of the mission sights. Every mission sight prefers one or more robots. The specification cannot be graphically represented and hence all mission sights are shown by the same color in Fig. 5(a) denoting the specification for the problem. There are 25 mission sights and 6 robots. The roadmap is shown in Fig. 5(b) while the output of the optimizer is shown in Fig. 5(c). The path traced by each robot is shown in Fig. 5(d). It can be seen that in this scenario every robot primarily goes to the mission sight for which it is the most required, and a few neighboring sights in the vicinity.

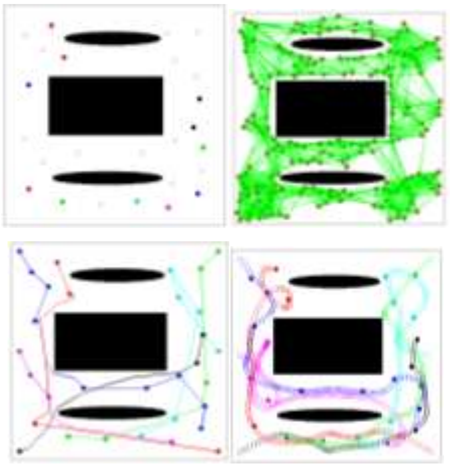


Fig. 4. Results for scenario 2

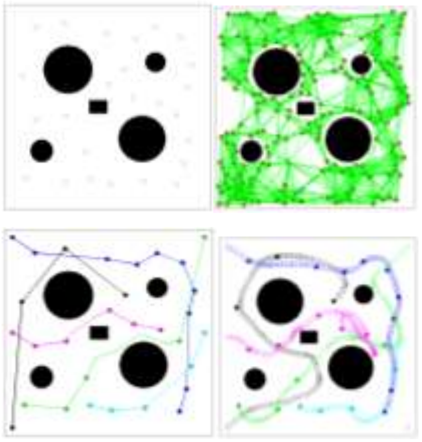


Fig. 5. Results for scenario 3

B. *Analysis of Parameters*

After the simulation results, the parameters of the algorithm are assessed. The second scenario is used in all analysis. One of the most important parameters of the algorithm is the number of vertices in the roadmap |*V*|. The complexity of roadmap construction is O(|*V*|) as all other factors in the roadmap construction are constants. The complexity of computing the cost matrix ||χ|| is O(|ρ|.(|ρ|+|*L*|).|*V*|), where |ρ| is the number of robots and |*L*| is the number of mission sights. In the worst cases, higher number of vertices in the roadmap result in larger time to search between any pair or nodes, and hence a higher computation time for construction of the cost matrix ||χ||. However larger number of vertices also result in smaller paths in the search and thus reducing the search effort. In other words, the heuristic function used in the A* algorithm is the Euclidian distance to goal, which gets better as the number of vertices increase and thus facilitate near optimal paths.

Fig. 6(a) shows the computation time for construction of the roadmap *G*(*V*,*E*), Fig. 6(b) shows the computation time to compute the cost matrix ||χ||. Fig. 6(c) shows the total computation time before optimization. The larger number of vertices |*V*| however result in better paths which is a better representation of the cost matrix. Fig. 7(a) shows the percent of queries needed for the computation of the cost matrix ||χ|| which can be answered as at least one path is represented in the roadmap. The average cost of the cost matrix ||χ|| is shown in Fig. 7(b) for cases resulting in a complete matrix. It can be easily seen that the metrics improve with the increase in the number of vertices, also requiring more computation time.

The next important task is optimization using GA along with the LS. Both the algorithms are iterative in nature that is more number of iterations of each of these searches results in a better individual. While the LS is likely to get trapped in a local optima, the GA mostly results in convergence at the global optima. The cost of the individual is plotted with the number of iterations. The results for GA are shown in Fig. 8(a) and the results for LS are given in Fig. 8(b).

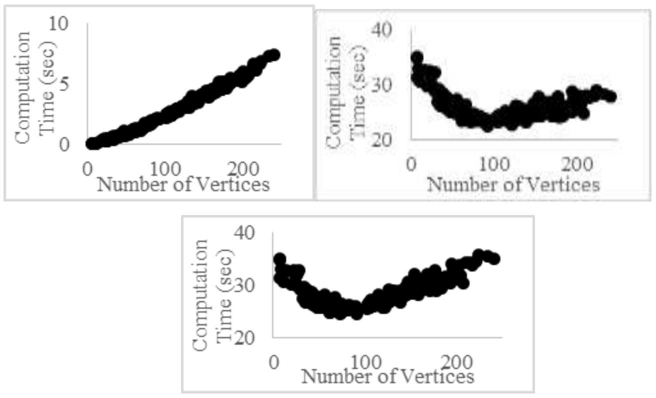


Fig. 6. Computation Time before Optimization (a) Roadmap Construction, (b) Cost Matrix Computation (c) Total

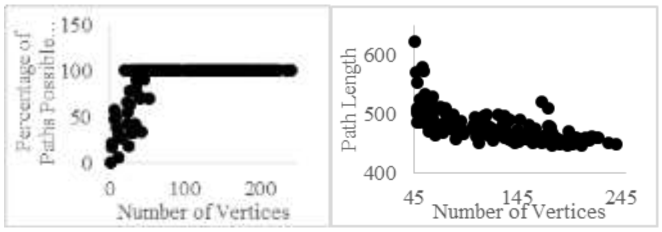


Fig. 7. Optimality of the Roadmap (a) Percentage of paths computatble, (b) Average Path Length

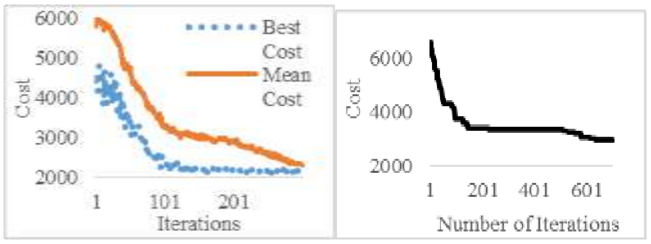


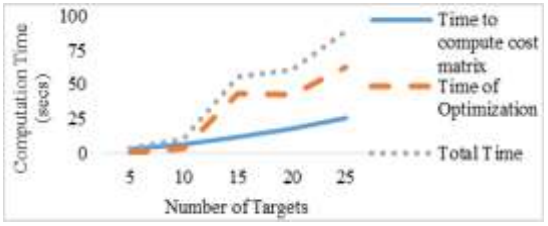
Fig. 8. Optimization (a) Genetic Algorithm, (b) Local Search

C. *Scalability*

The computational complexity of the roadmap construction is O(|*V*|) and is independent of the number of targets and the number of robots. The computation complexity of cost computation matrix is O(|ρ|.(|ρ|+|*L*|).|*V*|). The computation time of optimization is dependent upon the number of targets and robots while the algorithm is iterative in nature. The genomic length is 2|*L*| and more be the genomic length, more is the computation time. The computation time and cost are studied for increasing number of mission sights and robots.

First experiments are done for increasing number of mission sights. The optimization is first done for prolonged duration of time to get the optimal results. The computation time of optimization is taken as the time needed in optimization for generating a result which is at least 90% as good as the optimal result. The computation time of generation of cost matrix, optimization and the total computation cost are plotted in Fig. 9(a). The little drop of computation time between 15 and 20 targets is due to the stochastic nature of the algorithm. The cost of the solution is plotted in Fig. 9(b).

The next set of experiments are on increasing the number of robots. The number of robots is not represented in the genomic length of the optimization individual, however it does determine the domain of the variables used and hence the factor is of importance. The results for increasing the number of robots are shown in Fig. 10. It can be seen that there is only a minor addition in the computation cost on addition of more robots. More robots however mean that the entire mission can be solved more easily and hence the solution cost decreases on increasing the number of robots. The increase in cost in the case of 2 robots and 6 robots is due to the stochastic nature of the algorithm. It must though be mentioned that not all robots participate in all missions.



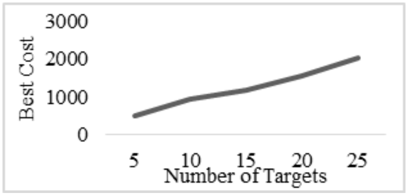
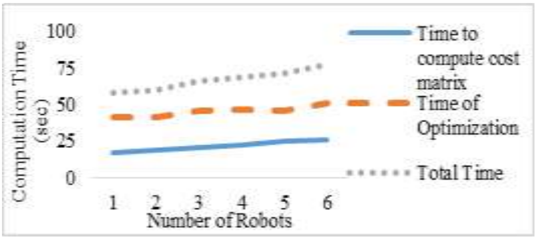


Fig. 9. Experiments with increasing the number of targets



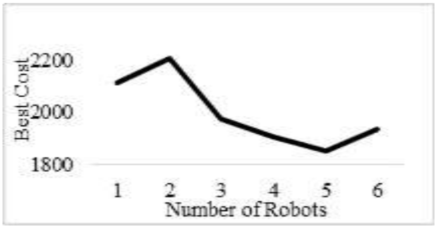


Fig. 10. Experiments with increasing the number of robots

VII. Conclusions

This paper presented the problem of mission planning for multiple mobile robots. A team of robots was asked to visit a set of mission sights. The workspace was sampled using a PRM, which was used to compute the traversal cost between every pair of nodes. The cost matrix was used by a GA running in a centralized mode to construct the mission for every robot. The mission was executed using a fuzzy navigator. Experiments revealed that the robots could coordinate to execute the mission. Increasing the number of vertices in the roadmap, the number of iterations of the optimization, the number of targets and the number of robots resulted in more computation time of the algorithm, however resulted in a better solution. Future research focus will be on using a better local navigation algorithm, making a near-decentralized optimization algorithm for the mission planner, adapting the roadmap construction to the scalability and complexity of the mission specification, continuously adapting missions as they are executed or the specifications are changed, and implementations on a physical robot.

Acknowledgment

The work is supported by the Indian Institute of Information Technology, Allahabad and the Department of Science and Technology, Government of India.

References

[1] T. Arai, E. Pagello, and L. E. Parker, "Guest editorial advances in multirobot systems," *IEEE Transactions on Robotics and Automation*, vol. 18, no. 5, pp. 655,661, 2002.

[2] H. Choset, K.M. Lynch, S. Hutchinson, G.A. Kantor, W. Burgard, L.E. Kavraki, and S. Thrun, *Principles of Robot Motion*, MA: MIT Press, 2005.

[3] R. Tiwari, A. Shukla, and R. Kala, *Intelligent Planning for Mobile Robotics: Algorithmic Approaches*, Hershey, PA: IGI Global Publishers, 2013.

[4] R. Kala, “Coordination in Navigation of Multiple Mobile Robots,” *Cybernetics and Systems*, vol. 45, no. 1, pp. 1-24, 2014.

[5] M. Fisher, An Introduction to Practical Formal Methods Using Temporal Logic, West Sussex, UK: Wiley, 2011.

[6] M. Lahijanian, S.B. Andersson, and C. Belta, “Temporal Logic Motion Planning and Control With Probabilistic Satisfaction Guarantees,” *IEEE Transactions on Robotics*, vol. 28, no. 2, pp. 396-409, 2012

[7] H. Kress-Gazit, G.E. Fainekos, and G.J. Pappas, “Temporal-Logic-Based Reactive Mission and Motion Planning,” *IEEE Transactions on Robotics*, vol. 25, no. 6, pp. 1370-1381, 2009.

[8] E.K. Xidias and P.N. Azariadis, “Mission design for a group of autonomous guided vehicles,” *Robotics and Autonomous Systems*, vol. 59, no. 1, pp. 34–43, 2011.

[9] A. Sud, R. Gayle, W. Andersen, S. Guy, M. Lin, and D. Manocha, “Real-time navigation of independent agents using adaptive roadmaps,” in *Proc. 2007 ACM Symp. Virtual reality software and technology*, New York, ACM. 2007, pp. 99-106.

[10] R. Kala, “Rapidly-exploring Random Graphs: Motion Planning of Multiple Mobile Robots,” *Advanced Robotics*, vol. 27, no. 14, pp. 1113-1122, 2013.

[11] D. Hsu, T. Jiang. J. Reif, and Z. Sun, “The bridge test for sampling narrow passages with probabilistic roadmap planners,” in *Proc. 2003 IEEE Int. Conf. Robotics and Automation*, vol. 3, 2003, pp. 4420-4426.

[12] J.J. Kuffner and S.M. LaValle, “RRT-connect: An efficient approach to single-query path planning,” in *Proc. IEEE Int. Conf. Robotics and Automation*, 2000, pp. 995-1001.

[13] R. Kala, “Navigating Multiple Mobile Robots without Direct Communication,” *International Journal of Intelligent Systems*, vol. 29, no. 8, pp. 767–786, 2014.

[14] R. Kala, A. Shukla, and R. Tiwari, “Fusion of probabilistic A* algorithm and fuzzy inference system for robotic path planning,” *Artificial Intelligence Review*, vol. 33, no. 4, pp. 275-306, 2010

[15] R. Kala, Video Results, Available at: https://youtu.be/9dScGfiiahU Accessed: 15th April, 2016.