Evolutionary Mission Planning

Rahul Kala, Abeer Khan, D. Diksha, S. Shelly, Surabhi Sinha
Robotics and Artificial Intelligence Laboratory
Indian Institute of Information Technology, Allahabad
Allahabad, India
{rkala001, abeerunscore96, darora742, shelly09kashyap, surv.surabhi}@gmail.com

Citation: R. Kala, A. Khan, D. Diksha, S. Shelly and S. Sinha, "Evolutionary Mission Planning," 2018 IEEE Congress on Evolutionary Computation (CEC), Rio de Janeiro, 2018, pp. 1-8.

Full Text Available At: https://ieeexplore.ieee.org/document/8477952

***Abstract*— The problem of mission planning enables a robot to solve for complex missions and thereafter execute the missions. The problem is seen as an advancement of the classical problem of motion planning wherein the task is to find a trajectory for a robot to go from a configuration A to a configuration B avoiding obstacles. The popular mechanism to solve the problem is using Temporal Logic specifications to specify a mission, and to thereafter use search strategies to solve the mission. The same requires an exponential complexity in terms of propositional variables to search and verify the mission plan. Further, optimality is usually not a criterion in finding a solution, and the transition costs are usually ignored leading to sub-optimal mission plans. As missions get more complex using a large number of propositional variables for specification, it may no longer be possible to use exponential complexity algorithms. The paper proposes the use of Evolutionary Computation to solve the same problem. We first develop a new restricted language for mission design. Even though the language is restrictive, it can specify a large number of missions of real life service robotics. Then we design an evolutionary computation framework to solve the mission. Probabilistic Roadmap technique is used to get the transition system and transition costs between regions of interest. The mission planner takes the mission specification and these transition costs to compute a mission plan. The mission plan is executed using a reactive navigator that can avoid any dynamic obstacle, other people and robots.**

Keywords—Evolutionary Robotics, Genetic Algorithm, Robot Motion Planning, Service Robotics

I. INTRODUCTION

The classic problem of robot motion planning [1] is defined as computation of a trajectory from a given point *A* to a given point *B*, avoiding obstacles. The robot workspace is mapped to a configuration space (*C*). Any configuration of the robot wherein the robot collides with an obstacle or itself constitutes the obstacle configuration space (C^{obs}), while the set of configurations recording no collision constitutes the free configuration space ($C^{free}=C\backslash C^{obs}$). The trajectory $\tau[0,1]\rightarrow C^{free}$, defines a way to go from the source ($\tau(0)=A$) to the goal ($\tau(1)=B$), avoiding all obstacles on the way ($\tau(s) \in C^{free}$). The problem has been largely researched in the literature and a very large number of solutions have already been proposed that work well in some context that is practically observed.

However a large number of planning problems that the robots face are neither a classical planning problem, nor can be conveniently decomposed into the classical planning problem paradigm. This forms a higher layer of abstraction problem called as the problem of mission planning. The problem of mission planning deals with a complex formulation of the goal that the robot must achieved, rather than a single, clearly identified goal.

One of the most popular methods to solve the problem of mission planning is by using Temporal Logic (TL). TL gives an excellent language to specify the missions in a formal way. The environment is decomposed into regions using triangulation or similar techniques, which gives a transition map between the regions. Each region has properties specified by the proportional variables. The TL mission specification is converted into an automaton, which is operated upon the transition system, to get a feasible transition automaton. This automaton is used to search for a solution or an operating strategy from the source to a state that satisfies the TL mission specification [3]. The strategy is then used by an executor to make the robot travel and carry the necessary actions.

With the same methodology, many works have been done. Kress-Gazit [4] identified the different regions of interest and used them to make a roadmap. The goal was a temporal logic specification for the motion of the robot. Lahijanian et al. [5] additionally included the concepts of probability to ensure that the robot solves all sub-missions satisfying the probabilities set by the user. The probabilities were themselves learnt using Reinforcement Learning. Svorenova et al. [6] solved the problem for the case of multiple robots. The environment was partitioned to get the transition costs, wherein an automaton based approach was used for search and verification.

The major problem associated with the approach is the exponential complexity with respect to the propositional variables encountered when the TL mission is converted to the automaton. Further searching the solution is still exponential. A direct mechanism of the application of the evolutionary computation is to use evolutionary algorithms for the search, which makes no betterment because verification of the solution will still be an exponential task. Further the solutions often disregard optimality of the solution. The intent is usually to find one solution rather than the best solution. The transition costs themselves are normally ignored. This can be fatal for a

robot executing a mission, since a very poor solution may exist for a very complicated mission, but the very poor cost of the solution may itself negate the utility of the mission. The human controller may instead, in such cases, volunteer to key in a solution computed by his/her expertise, which may still be better.

This motivates to use a restricted language to specify the mission, which is easier for the human operators to specify task, has high formalism to facilitate searching a solution, and is generalized enough for a high degree of scenarios. The limitations of the TL as noted above is largely due to the fact that the language is made for a very large class of systems, rather than being specific to the robot missions. The aim in this paper is hence to design a new language specifically for robot missions, using which the verification of the mission can be linear instead of exponential in terms of propositional variables.

Specifically, in this paper we will make use of sampling based approach of Probabilistic Roadmap (PRM) [7] to make the transition system. The approach is generalizable to high dimensions and does not assume an explicitly known structure of the obstacles in the configuration space, or the knowledge of the complete configuration space. A very small amount of work exists in the use of sampling based approaches in this problem. The low level execution involved continuous time navigation. Bhatia et al. [8] used a multi-layered approach for solving a mission with temporal goals, wherein the higher layer explored for information and made higher order plans. The work was extended [9] to include hybrid nonlinear dynamics for the high level temporal goals. They further used a lazy high level search for computational speedup. A tree based approach was used by McMahon et al. [10], wherein the authors used discrete abstractions of the environment to create transitions and designed low cost trajectories that satisfy Co-Safe Linear TL. Low level execution was attained using a Physics based engine.

A small step was taken in the same direction in the previous work of the authors [11], wherein the task was for a team of robots to visit a set of mission sites with some preferentiality constraints. The algorithm used a centralized optimization algorithm to allocate mission sites and visit order for the robots. The algorithm could be executed for a very large number of mission sites, however the representation capability was very small and the algorithm was too specific. Based on the same, this paper attempts to make a language that can be used by the evolutionary planner.

The choice of evolutionary algorithm for the specific problem is natural. Exact search techniques will require exponential time as the search space is clearly exponential in terms of the propositional variables, which is unacceptable. Normally heuristics play a major role in making the search effective, yet not compromising on the optimality of the solution to a good extent. However the generality of the language normally implies nullity of the specific heuristics to solve the problem, in which case the exact search methods go completely unguided. Meta-heuristic techniques like evolutionary algorithms offer excellent choice in such a context. The algorithm is anytime in nature, and keeps improving the solution with time which is very good for robotic scenarios. Further the algorithm can search high dimensional search spaces very effectively to compute feasible solutions first, and to thereafter improve the solutions.

Evolutionary approaches have not been used for mission planning to the best of the knowledge of the authors, baring the literatures with a very specific definition of a mission, like the Travelling Salesman problem. Xidias et al. [12] used a centralized Genetic Algorithm for planning a group of autonomous vehicles. Their mission specification was simply to visit all of the mission sites. Similarly Kala [13] used an evolutionary framework to move multiple mobile robots, initially planning them in a decentralized manner, and then minimally deviating the plans only of conflicting robots to generate a feasible plan. The problem was simply to go from one point to the other. Many of the works on evolutionary motion planning of mobile robots are based on the works of Xiao et al. [14].

The paper is organized as follows. Sec. II describes the language used to specify the missions of the robot. Sec. III gives the evolutionary framework to solve the problem. Sec. IV gives the results while Sec. V gives the conclusion remarks.

II. Language for Mission Specification

While the natural English language is the most convenient for the user to specify the task of the robot, formalism in the language is needed in order to develop a computational framework that can be used for search and verification of the mission. Overall, the role of the language is to equip the human operator to give high level specifications which needs to be ascertained in the planned mission which is given to the robot for operation.

A mission site σ is defined as a specific place where the robot needs to go. Unlike some literatures, here the mission site is a specific site and not an entire region. We believe modelling a specific site is more practical, wherein the robot needs to perform a specific task. As an example, when a robot is asked to turn on the server at the robotics lab, in our specification, it needs to go to the specific spot and pose wherein manipulation needs to be performed, rather than go anywhere at the robotics laboratory. That said, in our planning, we do not currently consider any aspect of manipulation including the required sensing, uncertainties or costs. These will be handled in our future research.

A. Task

A task φ is defined as a Boolean expression of the mission sites. The task is achieved if a mission plan satisfies the task specification. The task can be easily given by the grammar (1).

$$\varphi ::= \sigma \tag{1}$$

$$\sigma ::= \sigma \wedge \sigma \mid \sigma \vee \sigma \mid \neg \sigma \mid (\sigma)$$

$$\sigma ::= \Diamond \sigma_1 \mid \Diamond \sigma_2 \mid \Diamond \sigma_3 \mid \Diamond \sigma_4 \mid \ldots\ldots \Diamond \sigma_n$$

$$\varphi ::= \varphi_1 \mid \varphi_2 \mid \varphi_3 \ldots\ldots \varphi_m$$

Here $\wedge$ is the 'and' operator, $\vee$ is the 'or' operator, $\neg$, is the 'not' operator and $\Diamond$ is the 'eventually' operator from temporal logic. *n* is the number of mission sites defined by the user, and *m* is the number of tasks defined by the user.

Consider a string τ_ψ consisting of a sequence of mission sites visited by the robot. Here the sub-script ψ is used to demarcate the mission plan as a sequence of mission sites with the actual trajectory of the robot. Let $\tau_\psi(i)$ denote the i^{th} item in the sequence and let $\tau_\psi(i,j)$ denote the sub-sequence from *i* to *j*. Let $\tau_\psi(i,end)$ denote the sub-sequence from *i* till the end of sequence. $\Diamond\sigma$ is said to be true if σ exists (or that the mission site σ is visited by the robot) in the plan sequence $\tau_\psi(i,j)$. If σ does not exist in the sequence, it is regarded as false. This is given by (2).

$$\Diamond\sigma \models \tau_\psi(i,j) \; iff \; \exists s : \tau_\psi(s) = \sigma, i \le s \le j \; . \tag{2}$$

In order to verify if a plan sequence $\tau_\psi(i,j)$ satisfies a task φ, the expression given by the task needs to be evaluated. First the truth of the ground variables $\Diamond s$ is checked using (2). The same can be done by using a hash table of size equal to the number of mission sites $|\sigma|$, initially all initialized to false, and marking all sites visited as the plan string is read. The algorithm hence has a memory complexity of $O(|\sigma|)$ and a time complexity of $O(|\sigma|+|\tau_\psi(i,j)|)$.

Noting the truth of every $\Diamond s$, the validity of a plan $\tau_\psi(i,j)$ for every task can be checked by the usual laws of Boolean algebra. For computation, the in-fix expression as specified by the operator is first converted into a post-fix expression in a computation complexity linear to the length of the plan. Then the post-fix expression is evaluated again in a computation time linear to the length of the plan. Hence the entire algorithm has a time and memory complexity of $O(|\sigma| +|\tau_\psi(i,j)|)$.

Some simple examples of task are as follows: The task asking the robot to inspect 5 different laboratories (L_1, L_2, L_3, L_4, L_5) is given by (3). The task asking the robot to go to either two of the three assistants (A_1, A_2, A_3) and either of the two robotics professors (R_1, R_2) to get some approval (that requires signatures of minimum one professor and two assistants) and at all times avoiding going near the laboratory area (L) because of restricted use is given by (4). Similarly, the task asking the robot to get some envelope from any of the two stores (S_1, S_2), a photocopy from any of the two machines (P_1, P_2), and snacks from the cafeteria (C) is given by (5). In all the examples the variables have their guessable meanings.

$$\varphi_1 = \Diamond L_1 \wedge \Diamond L_2 \wedge \Diamond L_3 \wedge \Diamond L_4 \wedge \Diamond L_5 \; . \tag{3}$$

$$\begin{aligned} \varphi_2 &= \left((\Diamond A_1 \wedge \Diamond A_2) \vee (\Diamond A_1 \wedge \Diamond A_3) \vee (\Diamond A_2 \wedge \Diamond A_3)\right) \\ &\wedge (\Diamond R_1 \vee \Diamond R_2) \wedge \neg\Diamond L \end{aligned} \; . \tag{4}$$

$$\varphi_3 = (\Diamond S_1 \vee \Diamond S_2) \wedge (\Diamond P_1 \vee \Diamond P_2) \wedge \Diamond C \; . \tag{5}$$

Algorithm 1 gives the process of verification if a sub-sequence satisfies the task φ. Here ζ is a hash table that stores all positive literals.

Algorithm 1: Verify φ
Inputs: plan $\tau_\psi(i,j)$, task φ
Outputs: Whether $\tau_\psi(i,j)$ satisfies φ

$\zeta(\sigma)$=false $\forall$ σ
for a = i to j, $\zeta(\tau_\psi(a))$=true
for every $\Diamond\sigma$ in φ, replace $\Diamond\sigma$ in φ by $\zeta(\sigma)$
convert φ from infix to postfix
result ← Evaluate postfix
return result

B. *Mission*

The task is a rich problem specification technique and is easily solvable as visible from the examples, however, the problem with task is that it virtually does not enable one to specify any temporal constraints of some event happening before or after the other event, for which the language is extended. A mission (ψ) is defined as a temporal order in which the tasks need to be carried out, and can be easily specified as the sequencing operation of the temporal logic. Consider φ_1, φ_2, φ_3 … φ_k to be *k* tasks that must happen one after the other, the mission is specified as (6).

$$\psi = \Diamond(\varphi_1 \wedge \Diamond(\varphi_2 \wedge \Diamond(\varphi_3 \wedge ...))) \; . \tag{6}$$

Some simple examples of task are as follows: The mission asking the robot to first inspect three robotics laboratory (L_1, L_2, L_3), then report the findings to any of the two robotics professor (R_1, R_2), and then report to the head of robotics (H) is given by (7). The mission to first get a document signed by all the three robotics assistants (A_1, A_2, A_3), and then by either of the two robotics professors (R_1, R_2) is given by (8). The mission to get envelope from either of the two stores (S_1, S_2) and a photocopy from any of the two machines (P_1, P_2), then to get stamp from any of the two offices (M_1, M_2), and then to post the letter at the post office (O) is given by (9). In all the examples the variables have their guessable meanings.

$$\begin{aligned} &\varphi_1^1 = \Diamond L_1 \wedge \Diamond L_2 \wedge \Diamond L_3, \varphi_2^1 = \Diamond R_1 \vee \Diamond R_2, \varphi_3^1 = \Diamond H, \\ &\psi_1 = \Diamond\left(\varphi_1^1 \wedge \Diamond\left(\varphi_2^1 \wedge \Diamond\varphi_3^1\right)\right) \end{aligned} \; . \tag{7}$$

$$\begin{aligned} &\varphi_1^2 = \Diamond A_1 \wedge \Diamond A_2 \wedge \Diamond A_3, \varphi_2^2 = \Diamond R_1 \vee \Diamond R_2, \\ &\psi_1 = \Diamond\left(\varphi_1^2 \wedge \Diamond\varphi_2^2\right) \end{aligned} \; . \tag{8}$$

$$\begin{aligned} &\varphi_1^3 = (\Diamond S_1 \vee \Diamond S_2) \wedge (\Diamond P_1 \vee \Diamond P_2), \varphi_2^3 = \Diamond M_1 \vee \Diamond M_2, \\ &\varphi_3^3 = \Diamond O, \psi_1 = \Diamond\left(\varphi_1^1 \wedge \Diamond\left(\varphi_2^1 \wedge \Diamond\varphi_3^1\right)\right) \end{aligned} \; . \tag{9}$$

The main task is to verify the satisfaction of a string τ_ψ for a specific mission specification ψ with a small computational complexity. Let υ_i be the index of the first item in the sequence τ_ψ such that $\tau_\psi(1, \upsilon_i)$ satisfies φ_1, φ_2, φ_3 … $\varphi_{i..}$ Given that υ_{i-1} satisfies the sequence φ_1, φ_2, φ_3 … $\varphi_{i-1,}$ we can say that the satisfaction of φ_i is due to subsequence $\tau_\psi(\upsilon_{i-1}, \upsilon_i)$. Hence υ_i may be given by (10-11).

$$\upsilon_1 = \begin{cases} \min_j\left(j : \tau_\psi(1,j) \models \varphi_1\right) & if \; \tau_\psi \models \phi_1 \\ \infty & otherwise \end{cases} \; . \tag{10}$$

$$\upsilon_i = \begin{cases} \min_j \left(j : \tau_\psi(\upsilon_{i-1}, j) \models \varphi_i \right) & \text{if } \upsilon_{i-1} \neq \infty, \\ & \tau_\psi(\upsilon_{i-1}, end) \models \varphi_i, \\ & i > 1 \\ \infty & otherwise \end{cases} . \quad (11)$$

If υ_k is finite, then it is apparent that the plan τ_ψ is a valid plan that satisfies the mission specification ψ. Moreover, the sequence $\tau_\psi(\upsilon_i+1,end)$ is unnecessary and can be deleted without actually affecting the validity of the plan.

The pseudo-code for checking the validity of a plan τ_ψ for a mission specification ψ is given by Algorithm 2.

Algorithm 2: Verify ψ
Inputs: plan τ_ψ, mission ψ
Outputs: Earliest point of satisfaction of plan υ_k, ∞ if the plan does not satisfy ψ

```
for i = 1 to |τψ|
        if Verifyφ(τψ(1,i), φ1)
                ν1=i, break
        end if
end for
if i=|τψ|+1, νk=∞ , return

for i=2 to k
        for j= νi-1+1 to |τψ|
                if Verifyφ(τψ(νi-1+1,j), φi)
                        νi=j, break
                end if
                if j=|τψ|+1, νk=∞ , return
        end for
end for
```

The algorithm Verify φ has a time and memory complexity of $O(|\sigma| + |\tau_\psi(i,j)|)$ as discussed previously. Accordingly the mission verification algorithm has a time complexity of $O(|\sigma|.|\tau_\psi|+|\tau_\psi|^2)$. This would have required an exponential complexity is using the complete class of temporal logic with an automaton based verification process.

III. Solution Framework

The approach uses a Probabilistic Roadmap (PRM) to construct a roadmap which can quickly answer shortest path queries between any two points. The roadmap is used to compute the path and path cost between every pair of mission sites and source. This cost information is used by an evolutionary framework to search for a mission plan, which is constantly verified for validity by using the validator of Sect. II and computing the corresponding cost using the cost matrix. The evolutionary algorithm is assisted with a local search in a memetic framework. The mission plan thus computed is used by a reactive navigator for execution, which uses a combination of deliberative and reactive planning framework to best avoid the new and dynamic obstacles, while adhering to the non-holonomic constraints. The overall approach is summarized by Fig. 1.

A. Probabilistic Roadmap

First the configuration space C^{free} is sampled out and represented in the form of a roadmap using PRM technique. The mission planning problem relies upon the solution to several classical *A* to *B* motion planning problems, which represent multiple queries over C^{free} and hence PRM is good choice for environments wherein no knowledge of the geometry of the obstacles is available. The PRM approach used here is directly taken from [15] (also [16]) and a small discussion is done here for the sake of completeness.

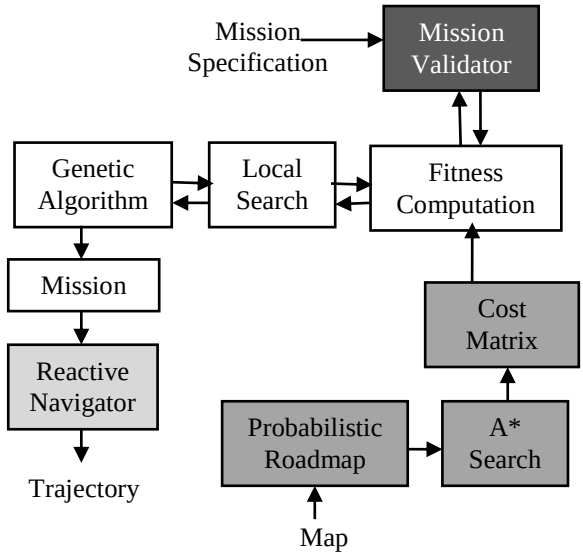

Fig. 1. Algorithm Design

The first problem is generation of the vertices. The approach uses a hybrid of narrow corridor sampling (using obstacle based sample for initial sample generation, promoting it to the free configuration space, and using bride-test to see if the same is inside a narrow corridor), obstacle based sampling, and uniform sampling. The initially constructed roadmap is then used for edge connectivity. First nearest *k* neighbors are attempted to be connected. If connection with the nearest *k* neighbors fails, a new vertex is left to ease subsequent connection attempts. Subsequently, four strategies are used for edge connections. The 'connect connected components' strategy identifies sub-graphs which are disconnected, and attempts connection of closest such graphs by connecting closest pairs of vertices from the disconnected sub-graphs. All selections are made stochastically, biased towards closer sub-graphs and vertices. In every attempt a single vertex is added in-between the two sub-graphs. The second strategy is the Expansive Space Tree style expansion of the leaf vertices. The third strategy creates a low dimensional hash of the configuration space to find the areas un-represented, and uses RRT style expansion to pull the roadmap towards such areas. The last strategy expands the roadmap randomly in a RRT style expansion.

B. Computing Cost Matrix

The PRM technique gives a fast method to compute the trajectory and the cost between any pair of mission sites. The plan is essentially a sequential specification of the mission sites, and hence it is important that for a speedy computation of

the plan cost, the transition cost and trajectory between every pair of sites is easily computable. In this method we store all possible transitions a priori, so that these can be used directly during search.

Let $\tau`(\sigma_i, \sigma_j)$ be the trajectory of the robot from a sites $\sigma_i$, to the sites $\sigma_j$, with the associated cost $\|\tau`(\sigma_i, \sigma_j)\|$. In order to compute the same, the sites σ_i and σ_j are temporarily added to the roadmap and connected to the nearest k-vertices, and A* algorithm is used to search the resultant graph. Here ` denotes that it is not the actual trajectory traced by the robot. Similarly the cost from the robot's source ($S$) to all the mission sites is also stored as $\tau`(S, \sigma_i)$ with the associated cost $\|\tau`(S, \sigma_i)\|$. The two matrices are used by the evolutionary search mechanism.

C. Evolutionary Search

The aim is to search for a sequence τ_ψ which is valid as per the formulation stated in Sec. II. Unlike many approaches, here the aim is to search for the optimal cost, as our transition system clearly has costs which may be very different from each other. It is possible to get plans which are valid but have miserably poor costs. The notion of optimality is the basic motivation behind the work. Exact search techniques can be used to search for the optimal solution, however the search will be exponential in terms of the length of the optimal path, which is assumed to be reasonably large for out model. Hence Genetic Algorithm is used to search for the best plan. The algorithm is probabilistically optimal and probabilistically complete, meaning that the solution cost tends to the optimal cost and the probability of getting a solution tends to one as time tends to infinity.

The genotype is encoded as a string of mission sites. A typical genotype is of the form $\tau_\psi=[\sigma_1, \sigma_2, \sigma_3, \sigma_4 \ldots. \sigma_g]$, where g is the maximum genomic length. The genotype represents the plan $[S, \sigma_1, \sigma_2, \sigma_3, \sigma_4 \ldots. \sigma_g]$. In order to calculate the fitness of the genotype, first we check for its validity to see if the motion sequence satisfies the mission specification ψ using Algorithm 2. The algorithm gives υ_k, the index of $\tau_{\psi,}$ which is enough to satisfy ψ. Any representation after that may be ignored. The fitness function is given by (12).

$$F(\tau_\psi) = \begin{cases} \|\tau`(S,\sigma_i)\| + \sum_{i=2}^{\nu_k} \|\tau`(\sigma_{i-1},\sigma_i)\| & if\, \nu_k \neq \infty \\ \infty & otherwise \end{cases}. \qquad (12)$$

The Genetic Algorithm uses Stochastic Universal Sampling technique. There are two types of mutation operators used. The first mutation operator is inspired by the combinatorial optimization techniques. The operator selects two genes and swaps them to make the mutated gene. Even though the robot may have to visit places multiple times for the adherence as per the specification formula, the problem still has characteristics of combinatorial optimization, especially at the level of tasks. The second mutation operator selects a gene and replaces it with some other gene. One point crossover technique is used wherein half the gene is taken from the first parent and the other half is taken from the other parent. An elite count of one is used. Correspondingly the initial population generation uses both a shuffle of the mission sites so as not to have repeated sites in the genotype, and a random assignment wherein a mission site can be repeated any number of times.

It is possible that the plan required to satisfy the mission is exceedingly big. The plan will require a large number of genes, which will adversely affect the GA due to the curse of dimensionality. To solve the problem, a provision is kept that if the complete individual cannot satisfy the mission plan, the genomic sequence may be repeated upto a maximum of 3 times. Currently the transition matrix has an all-connected structure due to limited modelling, however in the future the aim is to use principles of Grammatical Evolution for selecting the possible transition based on feasible transitions and genes as employed in [17].

The algorithm is assisted with a local search to make a memetic framework of the overall approach. The local gene uses both mutation techniques for generation of a new individual. If the new individual is better than the parent, the new individual is retained, otherwise the parent is retained. This gives the capability to very quickly reach the local minima, while the Genetic Algorithm attempts to find the global minima. A few iterations of local search are applied at every generation.

Once the evolution completes and the best plan is obtained, a few post-processing operations are employed to further aim at shortening the path length. The post-processing uses swapping of two genes and replacement of a gene by a new gene as two basic operations. Additionally a deletion operation is performed, wherein a gene is deleted, so as to avoid unnecessary mission site visits.

D. Exact Solver

The exact solver is not a component of the approach and is only used for the purpose of testing as a baseline. The exact solver is a search operation that uses Uniform Cost Search to search the path. The Uniform Cost Search is an optimal search that gives the best path possible, although with the problem of exponential complexity. The search starts from the source to make a search tree. At every iteration the lowest cost node is extracted. If the node, with path as given from the source in the search tree, satisfies the termination criterion laid down in Sec. II, the search terminates. Otherwise the children are generated whose cost is computed and added to the search queue. We use the search to testify the goodness of the planner for cases with small inputs.

E. Reactive Navigation

The plan τ_φ returned by the mission planner is a set of mission sites to be visited in the same order, starting from the source, in the form of $S, \sigma_1, \sigma_2, \sigma_3, \ldots \sigma_k$. The trajectory $\tau`(\sigma_i, \sigma_{i+1})$ between every consecutive pair of vertices and between from the source $\tau`(S, \sigma_i)$ is already known by the PRM planning. However the same trajectory cannot be used for execution as new obstacles may have emerged, dynamic obstacles may emerge and due to sharp turns in the roadmap. Hence a reactive navigation mechanism is used for the actual navigation.

A hybrid of deliberative and reactive planning framework is used. A deliberative path is already available which may not be navigable due to the presence of other people, robots and dynamic obstacles. The deliberative path serves as a guide for

the reactive navigator. The algorithm selects the farthest point in the deliberative path which is known to be collision-free considering only the static obstacles. This serves as the goal of the reactive navigator.

The reactive navigation scheme considers a conical region around the desired direction of to the goal, which is already known. Distances from static obstacles, other robots and people are measured within the range. The direction in C^{free}, which records the maximum distance (under a threshold) is used as the direction of immediate motion. If there is a tie, the largest gap is considered as well, in which case the middle point of the gap is taken. The linear speed is set to safe settings as per the available distance.

Currently the scenario is of a single robot only, and thus the utility of many features of the algorithm, however the navigation scheme is discussed here for completeness as the framework will soon be extended to multiple robots with multiple moving people in the robotic environment.

IV. RESULTS

The approach was tested against a large number of test scenarios and was found to perform well in all of them. The scenarios were made keeping the number of proposition variables very high, when exact results cannot be computed by an exhaustive search. However, the scenarios were generated by random generators, and therefore it is difficult to appreciate the solutions. For presentation in this paper, we stick to presenting very simple scenarios with some practical utility for the ease of presentation. The very large cases with randomly generated specifications will be used for subsequent analysis. This seems to be a rather well-adopted approach in the related publications.

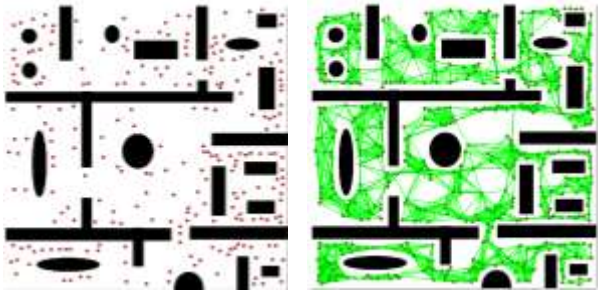

Fig. 2. Generation of Roadmap (a) Vertices (b) Edges

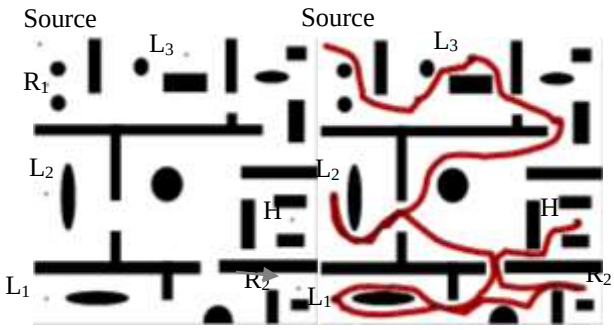

Fig. 3. Scenario 1 (a) Description of places (b) Path traced

All testing is done over the map shown in Fig. 2(a). The map is converted into a roadmap by sampling vertices shown in Fig. 2(a), and thereafter connecting edges as shown in Fig. 2(b). The map shown is the workspace of the robot, and not the configuration space, in which the obstacle gaps are much smaller than they appear. The algorithm is tested against the same examples used to illustrate the notion of tasks, given by (7)-(9). The first scenario is the mission asking the robot to first inspect three robotics laboratory (L_1, L_2, L_3), then report the findings to any of the two robotics professor (R_1, R_2), and then report to the head of robotics (H) given by (7). The places of interest are shown in Fig. 3(a). Fig. 3(b) shows the path traced by the robot. The results are better displayed in the supplementary video available at [18].

The second scenario corresponds to the mission to first get a document signed by all the three robotics assistants (A_1, A_2, A_3), and then by either of the two robotics professors (A_1, R_1, R_2) is given by (8). The places of interest are given in Fig. 4(a). The path traced by the robot is given by Fig. 4(b). The last scenario is corresponding to the mission to get envelope from either of the two stores (S_1, S_2) and a photocopy from any of the two machines (P_1, P_2), then to get stamp from any of the two offices (M_1, M_2), and then to post the letter at the post office (O) is given by (9). The places of interest are given by Fig. 5(a), while the path traced by the robot is given by Fig. 5(b).

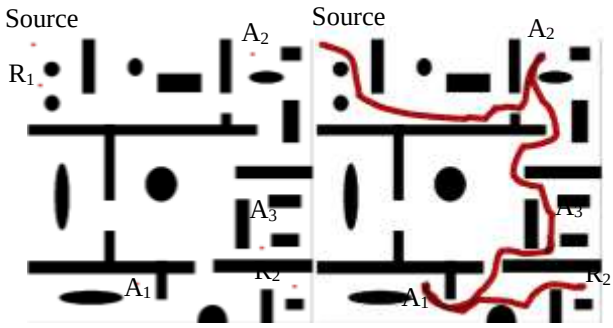

Fig. 4. Scenario 2 (a) Description of places (b) Path traced

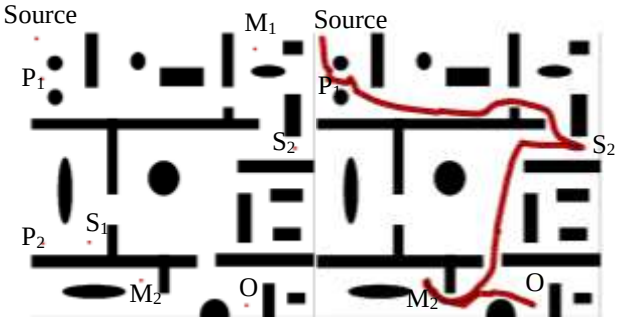

Fig. 5. Scenario 3 (a) Description of places (b) Path traced

Subsequent testing was done to test the performance of the algorithms on much higher problem sizes, wherein the exact solutions do not work. For the same reasons no comparative analysis is possible, as the approaches with exponential time

complexities are naturally unable to handle the problem sizes as experimented here. The construction of the roadmap and its implication, in general, to mission planning is directly taken from [11] wherein these parameters showed rather expected performance. A related analysis is not repeated here to save space.

The first set of experiments involved studying the convergence of the Genetic Algorithm. The testing was done with a problem size of 100, which is practically high enough to specify very complicated missions. The results for the overall Genetic Algorithm is given in Fig. 6(a), while the results for local search is given in Fig. 6(b). In order to get a near-optimal solution using the local search, very large number of iterations had to be performed, which still performed a little poor than the Genetic Algorithm. The cost metric is normalized for presentation.

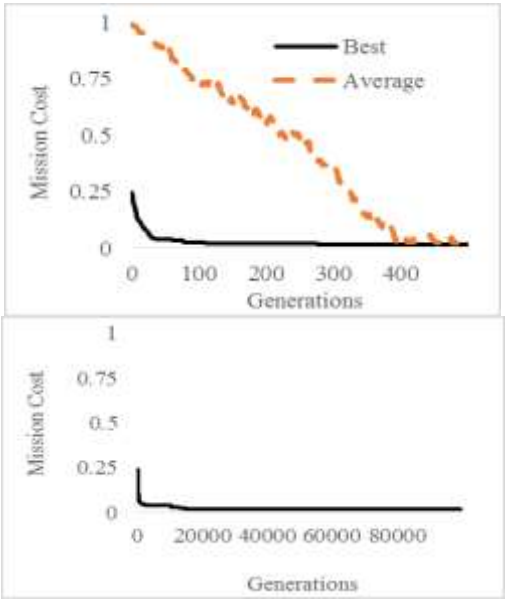

Fig. 6. Convergence of (a) Genetic Algorihtm (b) Local Search

Thereafter experiments were done to test the scalability of the approach. First testing was done on the exact solver. The solver took visibly no time for problems till a size of 12, which is very small. However soon thereafter the system was out of memory, but not before a prolonged duration of time. Linear memory complexity equivalents of search like the Iterative Lengthening algorithm could have not resulted in memory errors, however since the computation time is clearly exponential, there was no motivation behind testing for larger problem sizes. This is a limitation of the search based methods. Modern model checking tools can compromise optimality and run for problems of larger sizes, however, the exponential complexity still restricts their use on problems of very large size, a glimpse of which was seen here.

The experiments were first done to test the computation time of the fitness evaluation function, which has a computation complexity of $O(|\sigma|.|\tau_\psi|+|\tau_\psi|^2)$. The intent is to test its effect in the total optimization, and therefore instead of explicitly checking the computation time of the specific function, the computation time of the entire optimization process is considered, wherein optimization is carried for a fixed number of generations (100) and on a fixed number of individuals (150). The computation time is hence a constant time, plus the time of evaluation of the fitness function. The results are shown in Fig. 7. It is noteworthy that the computation time does not excessively increase with problem size, even though problem sizes upto 100 was tested. The graph is stochastic as the problems were generated randomly, and therefore a trendline is plotted in addition.

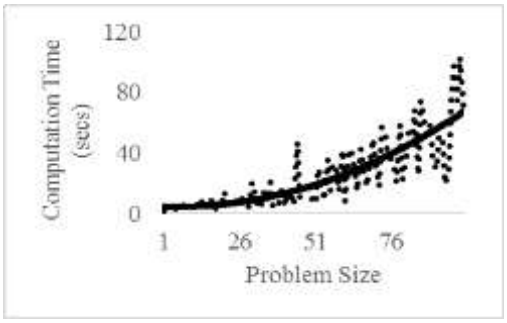

Fig. 7. Computation Time Study for Fixed Genetic Algorihtm Configuration.

Subsequent testing was done to check the performance of the algorithm in regard to optimality and computational time with an increase in the problem size. The first problem is to find the optimal solutions to the randomly generated problems. As it is not possible to get exact solutions, the only way to compute the optimal solutions was to leave the algorithm running for prolonged hours and the same was repeated for 10 times. Thereafter the computation time was plotted against the problem-size for different percent deviations from the optimal solution. The results are shown in Fig. 8. The legends correspond to the deviation from the optimal path, with 0 denoting the time to get the optimal path and 0.4 denoting a 40% deviation. The noisy trend is due to the stochastic nature of the problem, some very large problem generated turn out to be very simple and vice versa. The results are plotted as a trendline for smoothening, using the median filter with a window of 5. It is imperative that larger problem sizes and more nearly-optimal solutions require more computation time as compared to the counterparts.

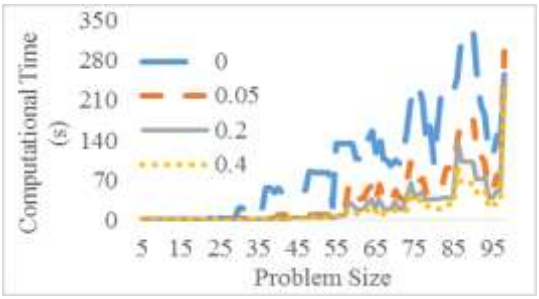

Fig. 8. Computation Time Study v/s Problem Size for different near-optimality (deviation from optmal paths)

In order to better understand the results, the results are also plotted as the deviation from the optimal path with respect to the problem size, for varying computation times. The results are divided into two parts. Fig. 9(a) shows the results when the computation time is very small and therefore the results are

very poor, and Fig. 9(b) when the computation time is large enough to give the near optimal results. The increase in computation time naturally increases the near-optimality, while the optimality is poor when the problem size is very large.

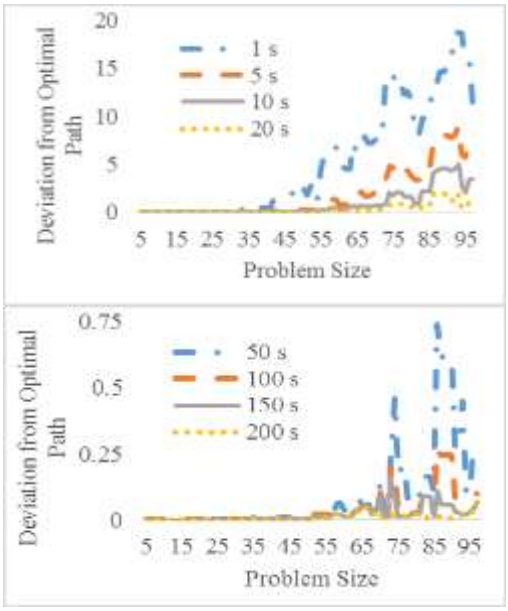

Fig. 9. Near-Optimality (deviation from optmal paths) v/s Problem Size for different computation times

From Fig. 8 and Fig. 9 it is clear that most small problems require a very small time to get respectable solutions. It may be argued that the computation time is not very small for larger problems. However even if you call as assistant to give very challenging tasks, you'll require time to explain to the assistant what needs to be done, who will further require time to analyze the problem and come up with a solution strategy, which may even be higher than the time the robot is taking. As the complexity of missions grow, it is acceptable to take more time. However what we have done here is to devise solutions for a problem, which cannot be solved by the methods available in the literature.

V. CONCLUSIONS

In this paper evolutionary methodology was used to solve the problem of mission planning. In the future robots will be able to perform numerous tasks for the humans, and therefore will be expected to accept complicated commands to carry out numerous tasks. The exact solution methods cannot be used when the number of variables used are very large, which motivates the use of evolutionary search for the problem. Using this technique, very complicated missions upto the size of 100 were solved with reasonable optimality. The computation time does not facilitate a near-real time performance, however, is acceptable considering that the highest degree of planning is being called, and the mission execution will take a prolonged duration of time.

In the future, the aim is to extent the work to multiple robots and the use of sensors wherein some information is only available on reaching a place. Decomposition techniques, attempting to decompose the complex missions into simpler missions and thereafter building an integrated solution, need to be tried upon. The algorithm needs to be extended to account for the probabilistic transitions and costs. The algorithm may be extended to planning the manipulation at the mission site. Also the algorithm needs to be extended to a real mobile manipulator.

ACKNOWLEDGMENT

Research supported by the Indian Institute of Information Technology and the Science and Engineering Research Board, Department of Science and Technology, Government of India through project number ECR/2015/000406.

REFERENCES

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

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

[3] C. Baier, J. P. Katoen, *Principles of Model Checking*, MIT Press, Cambridge, MA, 2008.

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

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

[6] M. Svorenova, J. Tumova, J. Barnat, I. Cerna, "Attraction-based receding horizon path planning with temporal logic constraints," In: *Proceedings of the 2012 IEEE 51st Annual Conference on Decision and Control*, pp. 6749-6754, 2012.

[7] L. E. Kavraki, P. Svestka, J.C. Latombe, M.H. Overmars. "Probabilistic roadmaps for path planning in high-dimensional configuration spaces." *IEEE Transactions on Robotics and Automation*, vol. 12, no. 4, pp. 566–580.

[8] A. Bhatia, L.E. Kavraki, M.Y. Vardi, "Sampling-based motion planning with temporal goals," In: *Proceedings of the 2010 IEEE International Conference on Robotics and Automation*, pp. 2689-2696, 2010.

[9] A. Bhatia, L.E. Kavraki, M.Y. Vardi, "Motion planning with hybrid dynamics and temporal goals," In: *Proceedings of the 2010 49th IEEE Conference on Decision and Control*, pp. 1108-1115, 2010.

[10] J. McMahon, E. Plaku, "Sampling-based tree search with discrete abstractions for motion planning with dynamics and temporal logic," In: *Proceedings of the 2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 3726-3733, 2014.

[11] R. Kala, "Sampling based Mission Planning for Multiple Robots," In: *Proceedings of the IEEE Congress on Evolutionary Computation, IEEE*, 2016.

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

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

[14] J. Xiao, Z. Michalewicz, L. Zhang, K. Trojanowski, "Adaptive Evolutionary Planner/Navigator for Mobile Robots", *IEEE Transactions on Evolutionary Computation*, vol. 1, no. 1, pp. 18-28, 1997.

[15] R. Kala, "Homotopy conscious roadmap construction by fast sampling of narrow corridors," *Applied Intelligenc*e, DOI 10.1007/s10489-016-0808-9, DOI: , 2016

[16] R. Kala, "Homotopic Roadmap Generation for Robot Motion Planning," *Journal of Intelligent and Robotic Systems*, vol. 82, no. 3, pp 555–575, 2016.

[17] R. Kala, "Multi-Robot Path Planning using Co-Evolutionary Genetic Programming," *Expert Systems With Applications*, vol. 39,no. 3, pp. 3817-3831, 2012.

[18] R. Kala, Supplimentary video, Available at: https://youtu.be/tjHMwaQY5kg, Accessed: 1st May, 2018.