Decentralized Multi-Robot Mission Planning using Evolutionary Computation

Sugandha Dumka, Smiti Maheshwari, Rahul Kala
Robotics and Artificial Intelligence Laboratry
Indian Institute of Information Technology, Allahabad
Allahabad, India
sugandha.dumka@gmail.com, maheshwari.smiti@gmail.com, rkala001@gmail.com

Citation: S. Dumka, S. Maheshwari and R. Kala, "Decentralized Multi-Robot Mission Planning Using Evolutionary Computation," 2018 IEEE Congress on Evolutionary Computation (CEC), Rio de Janeiro, 2018, pp. 1-8.
Full Text Available At: https://ieeexplore.ieee.org/document/8477839

***Abstract*— The classic problem of robot motion planning asks the robot to go from *A* to *B* avoiding obstacles. Missions are challenging problems asking the robot to visit a set of sites to accomplish a mission. The mission planning problems are largely studied as a Travelling Salesman Problem involving combinatorial optimization. In this paper the problem is generalized to any Boolean expression, giving more expressing powers to specify missions like "*Visit any one of three coffee machines*" or "*Visit any two of three instructors*", along with other mission sites to be mandatorily visited. The problem is solved using multiple robots in a decentralized manner. The Boolean expression is simplified into an 'OR of AND' format, which gives the flexibility to solve all the AND components and to select the minimum cost solution among them. Each of the AND components is a reduced multi-robot Travelling Salesman Problem solved by using k-medoids clustering and evolutionary computation. The results obtained by this approach are compared with the centralized algorithm and a master slave algorithm which uses a randomized algorithm for robot assignment, and for every such assignment the corresponding optimization problem of visiting the sites is solved for. The comparison depicts that as the problem size and the number of robots increase, the decentralized approach outperforms the rest enormously. The results are also tested on a Pioneer LX robot working in an office environment to carry dummy missions of everyday needs.**

Keywords*— *mission planning, multi-robotic systems, decentralized systems, evolutionary robotics, evolutionary computation.

I. Introduction

The robots are increasingly getting sophisticated and are able to do most of the low level jobs otherwise done by the humans. The day is not far when robots will be able to give services like getting a cup of coffee from the machine, getting items of interest from a store, fetching and getting basic items from a cupboard, etc. For doing the same operations the robot needs to be instructed by the humans about the jobs to be performed. Such complex problem specifications are called as missions.

The classic problem of motion planning [1,2] finds solution to the problem of "Go from *A* to *B* avoiding obstacles." Previously, many researchers believed that most missions can be easily decomposed into unit motion planning problems. So getting a coffee from the machine could be looked as a motion planning problem to go till the machine, a pick-and-place problem to fetch coffee, and another motion planning to come back. In case many such operations need to be performed, an expectation is that the human would enlist all such operations, while each operation may be a motion planning problem. However the humans may give complex missions to be performed by the robot that are not easily decomposable.

Typically, there are two schools of learning for solving the problem of mission planning. In the first school of learning the problem is specified by using a Linear Temporal Logic or Computational Tree Logic [3,4]. In addition to the normal logic consisting of the AND, OR and NOT expressions, the temporal logic allows for specifying temporal relations like eventually, globally, next, until, etc. The mission specification as a temporal logic is converted into an automaton, which is further pruned to allow only valid specific transitions. A search by using model verification techniques on such an automaton returns transitions that satisfy the given expression and thus represent a solution to the problem.

The model verification techniques have been widely used for both single and multi-robot cases, but the utility is restricted because of an exponential complexity of these techniques. The solutions are therefore not scalable to the level when missions will comprise of specifications given by hundreds of users with tens of robots and hundreds of sites, representing a typical home or institution scenario. Kress-Gazit [5] solved for missions specified as a LTL specification for the case of multiple robots. The resultant controller was extended to the continuous spaces. The generic problem was extended by Lahijanian et al. [6] to account for the probabilities of transition. The probabilities were learnt by using Reinforcement Learning, while the human user could specify the probabilistic guarantee needed and the planner made sure the strategy computed as a result met the specified guarantee. Svorenova et al. [7] added the notion of preferences and rewards to the generic problem definition. The same was used to get the search focused towards exploring towards the goal and simultaneously maximizing the rewards. McMahon

The work is sponsored by the Science and Engineering Research Board, Government of India through research grant ECR/2015/000406

XXX-X-XXXX-XXXX-X/XX/$XX.00 ©20XX IEEE

and Plaku [8] also used heuristics to guide the search and generate the results in good time. While the works are interesting, the problem still cannot be solved for very large sizes in terms of the number of mission sites and robots, while heuristics will do a little only for improvement. The approaches are not optimal.

The other school of learning uses evolutionary computation to solve the problem and is hence closer to our work. The missions are typically specified as the problem to visit all of a list of mission sites. This broadly may be called as the classic Travelling Salesman Problem with the major difference that the Travelling Salesman Problem takes as input a graph with the mission sites as vertices, while this problem is in a continuous space. The approach is better because the solutions are iterative in nature and even for very complex problems, will find a reasonable solution.

A lot of research has also been done to solve for such missions. The closest work is of Xidias et al. [9] who solved for a multi-robot mission planning problem wherein all sites had to be visited. The authors used a centralized Genetic Algorithm that also simultaneously optimized the trajectory of every robot. The approach as a result is extremely computationally expensive. In another related work Hussain et al. [10] solved the problem of mission planning with multiple robots in a centralized problem. The authors optimized both the waypoints in-between mission sites and the visiting order of mission sites simultaneously. The planning for the classic motion planning problem has been extensively solved by using evolutionary computation. Lamont et al. [11] solved a related problem called as the Swarm mission planning, wherein the authors optimize the schedule of visiting mission sites and enable the robots move as a swarm using a route obtained by multi-objective optimization. Upon reaching a mission site the swarm may split to attain different targets. Eberbach et al. [12] designed a language called the Common Control Language for the control of multiple robots, which could be solved by multiple techniques including a Genetic Algorithm. The interpretation of the language is however still computationally expensive which limits the utility. In a related work Kala [13] initially planned the path for a number of mobile robots and then coordinated for potential collisions by iterative local collision avoidance by using a Genetic Algorithm. For the case of a single robot, Xiao et al. [14] optimized the complete robot trajectory using a genetic algorithm.

Asking the robot to visit all of the mentioned mission sites to accomplish a mission can solve for a large variety of problems of everyday life, however, many problem specifications require a greater expressing ability. Consider you need a coffee, however there are 5 vending machines. The robot should be able to decide which of the 5 machines to get to. Similarly assume that an equipment issue letter needs signatures from any two academic staff members of a laboratory, when the laboratory is controlled by 4 academic staffs. Again the robot should be able to decide for whom to approach for signatures. In both cases, the robot may approach a mission site closer to another mission site that the robot has to compulsorily visit.

To explain the situation clearly, let us take an example. Say, a student has to get a report signed by the chief supervisor, Professor A compulsorily and also by either of the two co-supervisors, Professor B or Professor C. Moreover being a class representative he/she ensures that the board is cleaned and the markers are in place. Hence his/her operation and the mission sites (in brackets) would be: (i) Getting the report signed by Professor A, who is in his office (A); (ii) Getting the report signed by Professor B, who is in his office (B); (iii) Getting the report signed by Professor C, who is again in his office (C); (iv) Ensuring board is cleaned (X); (v) Ensuring that markers are in place (Y).

He/She also has a helpful friend who would take care of some of the operations. Thus his whole mission can be denoted as: $(\Diamond A) \wedge (\Diamond B \vee \Diamond C) \wedge (\Diamond X) \wedge (\Diamond Y)$ which gives the idea that operations with mission sites A, X and Y will be performed for sure but anyone of B or C can be performed to complete the mission. Here the operator '$\Diamond$' is the eventually operator of Linear Temporal Logic and specifies that the particular operation may be performed at any time. The mission will be planned in such a way that both the student and his friend complete it quickly and with fewer efforts. They have to thus divide the mission amongst them. One solution can be that the student might take the sign of Professor A and then go for the sign of Professor B, whose office is nearer and his friend might clean the board and ensure the markers are in place meanwhile. This is one solution, there can be many others. If the number of mission sites increase, the number of solutions will become huge. From those solutions, we need to find the optimal solution which can be in terms of the smallest amount of time or the smallest path length. We will be assigning the mission sites to complete the mission accordingly to the robots.

Assigning the mission sites so as to complete the mission optimally requires planning. In the above scenario, we can replace people by robots as now-a-days robots can easily perform operations in homes and offices. Such missions and sharing of responsibilities are very common with human beings. If the robots are to replace the humans in such clerical jobs, it is important that they be able to display the same capabilities of distribution of work, planning and executing. Thus, using robots we can automate mission planning. Moreover, using multiple robots we can more efficiently solve for the overall mission giving higher quality of service to the end users. Mission planning is thus assigning each robot some mission sites that it would be doing in order to achieve the requirements of a particular mission.

In this paper, we propose to solve the problem of mission planning with multiple robots using evolutionary computation. We cannot determine the exact optimal solution for mission planning as this is an NP hard problem. As the number of mission sites and the number of robots increase, it becomes very difficult to get the optimal solution as the combinations increase exponentially. The evolutionary approach is probabilistically optimal, meaning the probability of finding the optimal solution tends to one as time tends to infinity.

Even though evolution solves the problems of exponential complexity of search based techniques and can solve for generalized mission descriptions, the search would still take a

prolonged time to generate good solutions because the number of genes are linearly proportional to the number of mission sites and considering the number of robots, the resultant search space is exponential. The centralized solutions to multi-robot motion planning solve in such complex spaces and can take prolonged time for a large number of robots. The decentralized solutions on the contrary solve for every robot independently and coordinate between the robots for potential conflicts.

Here, we have approached the mission planning problem in a decentralized way. The whole mission is divided into a number of smaller missions according to the number of robots. Each robot is given its own list of mission sites and we find the local optimum solution for that particular robot using a genetic algorithm. Thus, each robot visits a particular list of mission sites optimally. Combining all the robot actions which are optimal locally, we get a global solution will is the final answer to the problem. To divide the mission into a list of smaller missions, we clustered the mission sites into the same number of clusters as the number of robots, which brings the nearby mission sites into one cluster. This is unlike the centralized algorithm in which we do not form clusters and optimize for mission site assignment to the robots by sending the whole genome to the genetic algorithm. Another mechanism to work is using a master algorithm to generate mission site assignment to the robots and a slave algorithm that sequences the assignments individually for every robot. The same strategy is also used as a baseline for comparisons, called as the master slave algorithm. We have in the end compared the proposed decentralized technique with the centralized and the master slave strategies and results suggest that the decentralized outperforms both the strategies.

The algorithm is tested on the Pioneer LX robot. The robot was initially made to traverse so as to make a map of the laboratory. The interesting mission sites were marked during the traversal. The robots were then given a mission that they optimized and followed their components individually.

This is an extension of our earlier paper [15]. In the earlier work the missions were restricted to Travelling Salesman Problem only, which are now generalized to any Boolean expression. The previous work solved the problem using a centralized optimization, which is now made decentralized. Further, the results are now also executed on a real robot, Pioneer LX, while the previous work was restricted to simulation. So it may be seen that the approach has been severely extended and the previous work may now be regarded as motivation only.

The major contribution of the paper are: (i) using evolutionary computation on a generalized mission specification consisting of AND and OR operators. This generalizes the current approaches of mission planning specified as a Travelling Salesman Problem for greater expressing ability, while does not face the exponential complexity problems associated with the model verification solvers. (ii) A heuristics is designed to divide the mission into a list of mission site assignment for every robot for the generalized problem description. The first heuristic notes that the input expression can be written in the OR or AND form, wherein every term consisting only AND component can be separately solved thus reducing the problem. The second heuristic uses clustering to divide the mission sites occurring in the AND expressions to assign them to robots. (iii) The proposed approach is experimentally compared with the centralized genetic algorithm and a master slave based genetic algorithm. By experiments it is seen that the proposed approach outperforms both strategies. (iv) The algorithm is also tested on the Pioneer LX robot running in the Robot Operating System (ROS) framework.

II. METHODOLOGY

A. Problem Definition

Let C be the *configuration space* of any robot. C^{free} is the set of configurations where the robot does not collide with any obstacle. C^{obs} is the set of configurations where the robot collides with some known obstacles, $C^{obs}=C\backslash C^{free}$. It is assumed the workspace and hence the configuration space is known in advance. Let $\Sigma=\{\sigma\}$ be the set of *mission sites*, $\sigma \in C^{free}$ which are also given. A *mission* (φ) is specified as a Boolean expression consisting of AND ($\wedge$) and OR ($\vee$) operators, given by the BNF (1)

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

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

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

where, $\Diamond$ is the eventually operator and n is the number of mission sites.

A set s satisfies φ ($s \vDash \varphi$) if the Boolean expression φ is true by replacing $\Diamond\sigma_i$ by true, iff $\sigma_i \in$ s, and replacing $\Diamond\sigma_i$ by false otherwise. The satisfiability can be checked with a complexity of $O(|\sigma|+|s|)$, where $|\sigma|$ is the maximum size of the mission and $|s|$ is the maximum length of the string.

There are k similar robots. The robots are stationed at source $S=\{S_i\}$, where S_i is the source of the i^{th} robot. Let $\Omega=\{\omega_i\}$ be the set of mission site assignments to the robot, where $\omega_i = [\sigma^1_i, \sigma^2_i, \sigma^3_i \ldots]$ is a linear sequence of mission sites to be traversed by the i^{th} robot.

Let c:$[\Sigma \cup S]^2 \rightarrow \{0 \cup R^+\}$ be the cost function with $c(\sigma_i,\sigma_j)$ as the cost between the mission sites σ_i and σ_j (any of them may also be the source for any robot). Computation of $c(\sigma_i,\sigma_j)$ is a classic robot motion planning problem.

The cost of a mission site assignment to a robot is given by (2)

$$C(\Omega) = \sum_i \left(c\left(S_i, \sigma_i^1\right) + \sum_j c\left(\sigma_i^{j-1}, \sigma_i^j\right) + c\left(\sigma_i^{end}, S_i\right) \right) \tag{2}$$

The problem is to find a sequence Ω that minimizes (2) with the constraint that the sequences together satisfy the mission φ. It is not necessary that the mission be satisfied considering the sequence of any one robot ω_i, however the robots collectively should satisfy φ. An abuse of notation is used here. Even though ω_i represents an ordered list, since there are no temporal constraint in the mission, we neglect the order and interpret as a set to take a union and create a

combined list of mission sites visited $\{\cup_i \omega_i\}$. The problem is hence given by (3)

$$\Omega^* = \arg\min C(\Omega), \text{ such that } \{\cup_i \omega_i\} \vDash \varphi \quad (3)$$

The solution of (3), that is Ω^*, gives a sequence of mission sites $\omega_i=[\sigma^1_i, \sigma^2_i, \sigma^3_i \ldots]$ to be visited by the robot from the source (S_i). The process of computing a solution to the cost function c also gives the trajectory between any two mission sites (or sources). Let $\tau'(\sigma_i, \sigma_j)$ denote the trajectory between σ_i and σ_j where any of the two may even be a robot source. While it may appear that the sequence $[\tau'(S_i, \sigma^1_i), \Pi_j[\tau'(\sigma^{j-1}_i, \sigma^j_i), \tau'(\sigma^{end}_i, S_i)]$ is a valid solution. Here [.] denotes an ordered list and Π_j denotes the list append operator, which is appending the trajectories one after the other. However the appended trajectory may not be realizable due to kinematic constraints and dynamic obstacles. Further every stop at the mission site requires a Human Robot Interaction and/or a manipulation operation. The actual trajectory τ is followed by a reactive technique guided by the deliberative trajectory τ'.

B. Solution Design

The overall approach for the work is given by Fig. 1. First the cost matrix c needs to be computed, which as a bi-product also gives the corresponding trajectory τ' between every pair of mission site and robot sources. The mission φ may be given in any form, however a heuristic used decomposes the mission into a Travelling Salesman Problem hence requiring the conversion of the mission into a OR of AND form. The processed mission is then used for decentralized mission planning. The solution generated Ω^* is an ordered list of mission sites to be visited by every robot. The same is executed individually by the robots using a fusion of deliberative and reactive mechanisms.

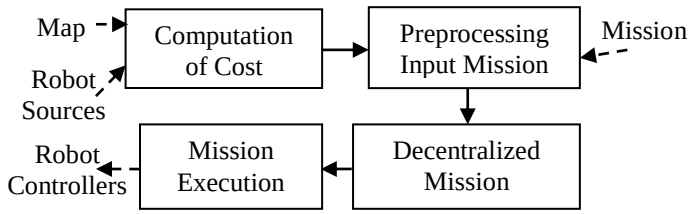

Fig. 1. Block diagram of the proposed methodology

C. Computation of Cost Matrix

The problem is to compute $c(\sigma_i, \sigma_j)$ between every pair of mission sites and sources. For the same the Probabilistic Roadmap [16] technique is used. Random vertices $V=\{v \in C^{free}\}$ are sampled from C^{free} to make the vertices of the roadmap. The technique to build the roadmap is used from [17] and is therefore not described in detail here. Sampling of vertices is done by using a hybrid of narrow corridor sampling, obstacle-based sampling and uniform sampling. Initially every vertex $v \in V$ is checked for connection with the neighboring vertices $u \in \delta(v)$ and the edge $<u,v>$ is added to the list of edges E if the straight line joining u and v is collision free, that is $\lambda u+(1-\lambda)v \in C^{free}$, $\forall\ 0 \leq \lambda \leq 1$. Here δ is the neighborhood function taken as the k nearest neighbors. If a connection attempt fails, a new sample is added near the obstacle where collision is recorded.

Subsequently four strategies are used for edge connection: connecting connected components in order to attempt Rapidly-exploring Random Tree (RRT) based connection of two disconnected sub-graphs; Expansive Spaces Tree (EST) based expansion of a leaf node identified as the one with lesser connectivity; hash based expansion into areas with lesser samples using RRT; and random RRT based expansion. The robot sources are similarly connected to the roadmap by adding them as vertices and connecting with the neighboring vertices. The roadmap so produced is passed to a Floyd Warshall algorithm to give the cost between all pair of vertices in the roadmap, which includes all sources and mission sites. The corresponding trajectory is also computed.

D. Pre-processing of the Input Expression

In the pre-processing step, conversion of the input mission specification φ to OR of AND form is done. For Example, let a_1, a_2, a_3, b_1, b_2, c_1, c_2 and c_3 be the various mission sites. Consider the scenario where any one of a_1, a_2 and a_3 has to be visited; and any one of b_1 and b_2 has to be visited; and any one of c_1, c_2 and c_3 has to be visited. This scenario can be expressed as: $(\Diamond a_1 \vee \Diamond a_2 \vee \Diamond a_3) \wedge (\Diamond b_1 \vee \Diamond b_2) \wedge (\Diamond c_1 \vee \Diamond c_2 \vee \Diamond c_3)$. This is an AND of OR units expression. To ease the further computation, it is converted to an OR of AND form which in this case would be: $(\Diamond a_1 \wedge \Diamond b_1 \wedge \Diamond c_1) \vee (\Diamond a_1 \wedge \Diamond b_1 \wedge \Diamond c_2) \vee (\Diamond a_1 \wedge \Diamond b_1 \wedge \Diamond c_3) \vee (\Diamond a_1 \wedge \Diamond b_2 \wedge \Diamond c_1) \vee (\Diamond a_1 \wedge \Diamond b_2 \wedge \Diamond c_2) \vee (\Diamond a_1 \wedge \Diamond b_2 \wedge \Diamond c_3) \vee (\Diamond a_2 \wedge \Diamond b_1 \wedge \Diamond c_1) \vee (\Diamond a_2 \wedge \Diamond b_1 \wedge \Diamond c_2) \vee (\Diamond a_2 \wedge \Diamond b_1 \wedge \Diamond c_3) \vee (\Diamond a_2 \wedge \Diamond b_2 \wedge \Diamond c_1) \vee (\Diamond a_2 \wedge \Diamond b_2 \wedge \Diamond c_2) \vee (\Diamond a_2 \wedge \Diamond b_2 \wedge \Diamond c_3) \vee (\Diamond a_3 \wedge \Diamond b_1 \wedge \Diamond c_1) \vee (\Diamond a_3 \wedge \Diamond b_1 \wedge \Diamond c_2) \vee (\Diamond a_3 \wedge \Diamond b_1 \wedge \Diamond c_3) \vee (\Diamond a_3 \wedge \Diamond b_2 \wedge \Diamond c_1) \vee (\Diamond a_3 \wedge \Diamond b_2 \wedge \Diamond c_2) \vee (\Diamond a_3 \wedge \Diamond b_2 \wedge \Diamond c_3)$.

The conversion can be simply done by using backtracking to generate all the possible combinations of AND units and consequently opening up of the brackets. The output is the same mission in the form $\varphi=\{\vee_i \varphi_i\}=\{\vee_i (\wedge_j \Diamond \sigma)\}$.

The mission φ is said to be consisting of *tasks* φ_i separated by OR units, wherein each task consists of AND separated mission sites. The problem can hence be reduced to (4-5).

$$\Omega^* \text{ is such that } C(\Omega^*) \leq C(\Omega^*_i)\ \forall i \quad (4)$$

$$C(\Omega^*_i) = \min_{\Omega_i} C(\Omega_i), \text{ such that } \{\cup_j \omega_j\} \vDash \varphi_i \quad (5)$$

Simply said, all the tasks are solved in parallel and the solution corresponding to the least cost task is accepted. This is one of the key heuristics used in the paper. By decomposing a general mission into a OR of AND units, or a collection of tasks, the problem is now converted into a form wherein robot assignment heuristics can be designed. Although there is an associated problem. Instead of solving one problem, now the number of problems to be solved in parallel are a^m, where a is the number of mission sites per OR and m is the number of OR units. So the number of problems is exponential in the number of OR units.

E. Decentralized Mission Planning

The problem is already reduced to a task planning, wherein a task φ_i contains a number of mission sites, all of which must be visited. Each of the AND units denoting the tasks is processed individually and the minimum of all the AND units denoting the mission is the required path length. The problem can be solved by using a centralized technique, however the resultant search space will have a size of $k^n.n!$, where k is the number of robots and n is the number of mission sites. The evolutionary searches also face problem when exposed to such high search spaces. The problem is therefore solved by using a decentralized approach. In order to solve the problem using a decentralized technique, the first problem is to divide the mission sites between the robots and the second problem is to sequence the mission sites assigned to every robot.

The first problem of distributing the mission sites to robots is solved by using clustering. It is clear from intuition that the mission sites nearby must be catered to by the same robot and the mission sites more distant apart can be distributed among robots. Using the same notion the mission sites are clustered into the same number of clusters as the number of robots, k.

While clustering may seem an obvious solution, it needs to be noted that clustering of mission sites cannot be done in the workspace. Consider two mission sites σ_i and σ_j such that both mission sites are in adjoining rooms with just a wall separating between them. Clustering in the workspace would assign both sites to the same robot, which however will have to travel all the way out of the room and inside the other to cover both the mission sites. We already have the true distance between every pair of mission sites $c(\sigma_i, \sigma_j)$ computed from Sec. II-B. The same distance function is used for clustering. Unfortunately that means k-means and similar clustering mechanisms cannot be used wherein the cluster representative is not a real point for which the cost is unknown. Hence the k-medoids algorithm is used for the clustering. For each of the tasks (AND units), firstly, a check is performed to find if the size of this unit is lesser than the number of robots, the problem can be solved naïvely by a greedy or an exhaustive search. If that is not the case, a k-medoids clustering algorithm is performed to form clusters of number equal to the number of robots.

Let $\kappa^{\varphi i}$ be the clusters such that $\kappa^{\varphi i}_i$ denotes the mission sites allocated to the i^{th} cluster. As per clustering property, $\kappa^{\varphi i}_i \cap \kappa^{\varphi i}_j = \emptyset$ and $\cup_i \kappa^{\varphi i}_i = \{\sigma_i \in \varphi_i\}$, which gives the required property that every mission site should only be visited by one robot and all robots should collectively visit all the mission sites.

The next step is to assign the robots to clusters or formulate the function M: $\kappa^{\varphi i} \rightarrow K$, where $\kappa^{\varphi i}$ is the set of clusters and K is the set of robots. $M(\kappa^{\varphi i}_r)$ denotes the robot assigned to cluster κ_r. The same may be done by a greedy or an exhaustive search.

The last step is to solve the problem of sequencing the mission sites for every robot, which is a simple combinatorial optimization problem. In other words, the robot $M(\kappa^{\varphi i}_r)$ needs to visit all mission sites in $\kappa^{\varphi i}_r$. Let $\omega_i = [\sigma^1_i, \sigma^2_i, \sigma^3_i \ldots]$ be the ordered list of mission sites visited by the robot $i=M(\kappa_r)$. The problem is hence given by (6-7).

$$C(\omega_i) = c\left(S_i, \sigma_i^1\right) + \sum_j c\left(\sigma_i^{j-1}, \sigma_i^j\right) + c\left(\sigma_i^{end}, S_i\right) \quad (6)$$

$$\omega^*_i = \arg\min C(\omega_i) \text{ such that } \omega_i = \text{perm}(\kappa^{\varphi i}_r),\ i = M(\kappa^{\varphi i}_r) \quad (7)$$

Here (7) is finding the minimum across all permutations of ω_i denoted by the *perm*() function implying the permutation constraint.

Genetic algorithm is used to find the optimal sequence ω^*_i. The individual representation consists of a random permutation of all mission sites κ_r. Roulette Wheel Selection with Rank based scaling is used. Mutation swaps two genes and one point crossover is used with a check to ensure that all the genes are represented once.

Instead of solving for all tasks in a sequential manner, the search happens in parallel. Therefore the algorithm maintains the anytime nature of the algorithm wherein a good solution will always be available, by taking the best current solution from the individual tasks. Let $P^{\varphi i}_i$ denote the population of the genetic algorithm for solving the j^{th} cluster of the task φ_i

The pseudocode of the overall algorithm is given as Algorithm 1.

Algorithm 1: Decentralized Mission Planning
Input: Mission φ, cost matrix c
Preprocess φ in OR of AND format, $\varphi = \{\vee_i \varphi_i\} = \{\vee_i (\wedge_j \Diamond \sigma)\}$.
$C_{min} \leftarrow \infty$, $\Omega_{min} \leftarrow$ NULL
for each task φ_i in φ,
    $\kappa^{\varphi i}$ = k-medoid clustering(φ_i,c)
    calculate robot assignment to cluster M
for each task φ_i in φ, for each cluster $\kappa^{\varphi i}_j$ in $\kappa^{\varphi i}$
    $P^{\varphi i}_i$ = Initial population as random permutation of ($\kappa^{\varphi i}_r$)
while stopping criterion is not met
    for each task φ_i in φ
        $C \leftarrow 0$, Ω_i = empty
        for each cluster $\kappa^{\varphi i}_j$ in $\kappa^{\varphi i}$
            ω^*_i = GAOptimize($\kappa^{\varphi i}_j$, C) for 1 generation
            $C \leftarrow C + C(\omega^*_i)$
            add ω^*_i to Ω_i
        if $C < C_{min}$, $C_{min} \leftarrow C$, $\Omega_{min} \leftarrow \Omega_i$
return Ω_{min}

F. Discussions

The pre-processing of the input has a complexity $O(a^m)$, equal to the number of tasks generated. Here a is the number of mission sites per OR and m is the number of OR units. Each task undergoes a k-medoid clustering whose complexity is $O(nk)$. In the worst case a cluster gets all n mission sites, while in the best case each of the cluster gets n/k sites. The search space is hence $n!$ for the worst case and $(n/k)!$ for the best case. The resultant complexity is hence $O(a^m + a^m.nk + a^m.n!) = O(a^m.n!)$ in the worst case and $O(a^m + a^m.nk + a^m.(n/k)!) = O(a^m.(n/k)!)$ in the best case. Note that the complexity here is an abuse of notation calculated assuming the Genetic Algorithm searches the entire search space, which is not true. However the computation time of the Genetic Algorithm can be guessed by the volume of the search space.

A centralized technique would require searching a space of $O(k^n.n!)$ and verification of the solution will take $O(n^2)$ time (a string of maximum length n needs to be verified for a Boolean formula of maximum length n), totaling a complexity of $O(k^n.n!n)$. As per assumption a and m are constants while k is a variable so and $n>>m$, so the complexity of the former turns out to be better in all cases.

Hence the heuristic of preprocessing of mission into tasks and then using the clustering heuristic on tasks changes the exponential from the number of mission sites into the number of OR units, which is more desirable. Further, usually one has a limited choices specified by the OR operation, while the number of robots for an organization may be reasonably large, which proves a better base of the term.

A little catch here is that the exponential on the number of OR units was solved by generating all possible tasks which is an exhaustive search technique operating in parallel, however the exponential of the centralized planner occurred in a term which was solved by using Genetic Algorithm, and as Genetic Algorithm searches only a proportion of the entire space, the exponential is not fully active. However another problem that centralized planners face is being stuck at a local minima. Even though theoretically it is possible to get out of all local minima until the global minima is found, given enough time. However, practically the probability of getting out of local minima is significantly small. Using multiple parallel Genetic Algorithms with some coordination can better use the computation, so that even if one run gets stuck in a minima the other may get the global minima. But this strategy spoils all beauty of the Genetic Algorithm as compared to a random search. Overall it may be better to have a large number of simpler problems, than a very difficult complex problems which is practically hard to solve irrespective of time.

III. Results

A. Applying Decentralized Approach

In order to test the algorithm we use the Pioneer LX robot from Adept Mobile Robotics. The robot was initially made to travel the Robotics and Artificial Intelligence Laboratory of the institute when there were no moving people acting as dynamic obstacles. The interesting points in the arena were marked as goals. The lidar data was converted into a map using the Mapper3 software. The map was loaded on the robot. The coordinates of the goal were retrieved from the map. The Probabilistic Roadmap was used to compute the cost matrix that was fed. The map along with all the mission sites is shown as Fig. 2.

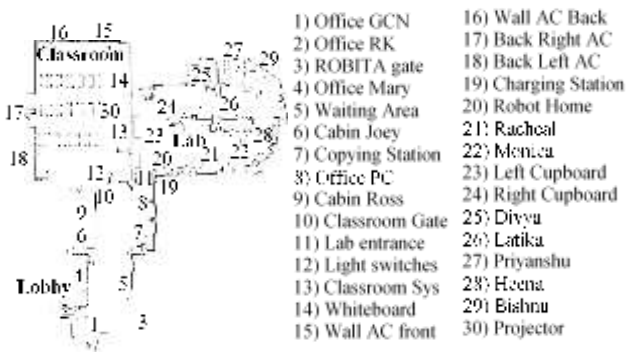

Fig. 2. Map of the robotics arena used for experimentation

The robot was given the mission: Eventually visit Office GCN (1) or Office RK (2); ROBITA Entrance Gate (3) or Office Mary (4); Waiting Area (5) or Copying Station (7); ROBITA Lab Entrance (6) or Whiteboard (14); Office PC (8) or Robot Home (20); Cabin ROSS (9) or Wall AC Front (15); ROBITA Lab entrance (11) or Right Cupboard (24); Light Switches (12) or Projector (30); Wall AC Front (15) or Back Right AC (17); Charging Station (19) or Left Cupboard (23); and Latika (26) or Heena (28). The mission expression is given by: $(\Diamond\sigma_1 \vee \Diamond\sigma_2) \wedge (\Diamond\sigma_3 \vee \Diamond\sigma_4) \wedge (\Diamond\sigma_5 \vee \Diamond\sigma_7) \wedge (\Diamond\sigma_6 \vee \Diamond\sigma_{14}) \wedge (\Diamond\sigma_8 \vee \Diamond\sigma_{20}) \wedge (\Diamond\sigma_9 \vee \Diamond\sigma_{15}) \wedge (\Diamond\sigma_{11} \vee \Diamond\sigma_{24}) \wedge (\Diamond\sigma_{12} \vee \Diamond\sigma_{30}) \wedge (\Diamond\sigma_{15} \vee \Diamond\sigma_{17}) \wedge (\Diamond\sigma_{19} \vee \Diamond\sigma_{23}) \wedge (\Diamond\sigma_{26} \vee \Diamond\sigma_{28})$.

This dataset was then run for obtaining results via decentralized algorithm. For 3 Robots, the path cost came out to be 5632. While the expression was be broken into a large number of tasks and the clusters will be different for every such task, the clusters formed for one of the tasks were: Cluster (i) σ_{26}, σ_{23}; Cluster (ii) σ_5, σ_4, σ_2; Cluster (iii) σ_6, σ_{11}, σ_8, σ_{12}, σ_{17}, σ_9. The units are arbitrary, specific to the tool and relate to the real distance units by constants. The clusters are shown graphically in Fig. 3(a). Each cluster will be taken by a particular robot. For 4 Robots, the cost came out to be 4722. For the task giving the least cost, the following clusters were formed: Cluster (i) σ_{12}, σ_{17}; Cluster (ii) σ_5, σ_4, σ_2. Cluster (iii) σ_8, σ_{11}, σ_6, σ_9; Cluster (iv) σ_{26}, σ_{23}. Each cluster will be taken by a particular robot. The clusters are shown in Fig. 3(b).

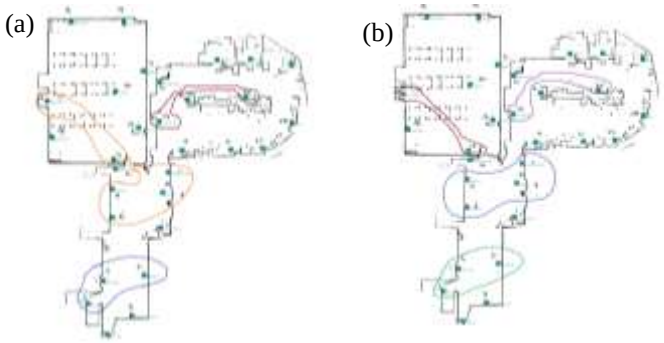

Fig. 3. Clustering of mission sites in a task for distribution between the robots (a) 3 robots (b) 4 robots

The results are discussed for the case of 4 robots. The order of mission sites traced by the robots is shown in Fig. 4. The lab

did not have 4 real robots and therefore the physical results were recorded separately by 4 runs of one robot imitating four simultaneous results. Note that there is no issue of inter-robot collision avoidance since each robot works in an independent cluster. Due to space only 3 robots are shown in the video. The results on Pioneer LX robot are attached as a supplementary video [18]. The audio is muted for simultaneous play.

B. Comparisons between Different Approaches

To test the performance, the algorithm is also tested with other approaches. The search based and model verification techniques have exponential complexity and failed to give results even for very small problem sizes, and hence those are not formally presented for comparisons. Further, the dataset used in Sec. III-A had only 30 sites, which is a very small problem. Hence to be able to compare the results with other algorithms and observe the trends while varying the inputs, datasets of higher orders had to be generated.

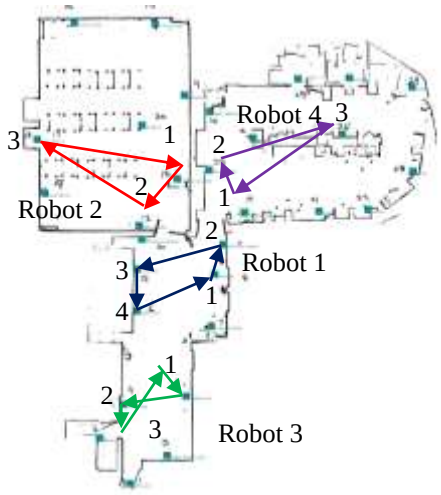

Fig. 4. Order of mission sites visited by the different robots

First, given n, the number of mission sites, an $n \times n$ cost matrix has to be generated. In order for the matrix to represent a distance function, it must follow the properties: (1) Matrix should be symmetric, or $c(\sigma_i,\sigma_j)=c(\sigma_j,\sigma_i)$. (2) For any two mission sites σ_i and σ_j, there should not exist another mission site σ_k such that the shortest distance between σ_i and σ_j through σ_k is shorter that the distance between σ_i and σ_j, or $c(\sigma_i,\sigma_k)+c(\sigma_k,\sigma_j) \geqslant c(\sigma_i,\sigma_j)$. To generate the cost matrix, firstly a random symmetric matrix $c_{n\times n}$ was constructed. Then Floyd-Warshall algorithm is applied to find the shortest distance between any two mission sites. The obtained matrix storing shortest path is the required cost matrix. The mission was generated as a random string. To generate the most complex problems, the mission was generated as a AND of OR expression, to be later opened as a long OR of AND processed form. The OR operator was generated with a probability of 0.4 and AND with a probability of 0.6.

The decentralized approach was compared with two other approaches which are Centralized and Master Slave on various datasets. In the centralized approach the robot assignment to mission sites and the order of visit of mission sites is present in the same generic individual. Genetic algorithm is used for computing the cost. In each iteration, crossover and mutation operators are performed to modify the population for the next generation.

A hierarchical Genetic Algorithm takes an excessively long time and is hence not a good candidate for comparisons. As a result a master slave approach was used in which the robots were assigned to the mission sites randomly as the master algorithm. After that, the mission sites are segregated among the robots as per the random allotment. Then the genetic algorithm was applied to calculate the cost.

Comparisons are made according to path cost vs the number of robots. A typicality here is that the proposed approach is not fully evolutionary since breaking the problem into tasks is like a parallel grid search and therefore the benchmarks become of different domains whose results will severely depend upon the parameters set. The comparison philosophy is to first run the different algorithms for prolonged durations to get an estimate of the best cost they may compute, and then to re-execute the algorithm terminating the search when the algorithm gets 5% of the optimal cost. So the costs computed are a fair indicator of the optimal cost while it also gives an indication of a practical computational expense. Further, to make the comparisons realistic, all algorithms work by breaking of the mission into tasks and searching the tasks in parallel. The readings taken from multiple iterations are averaged out to obtain the final reading to be plotted. The results were obtained using 31, 500, 750 and 1000 mission sites, representing very small, small moderate and large problem sizes noted in accordance with the number of mission sites possible in different home/lab/personal office, corporate office and large scale office scenarios. The results are given in Fig. 5.

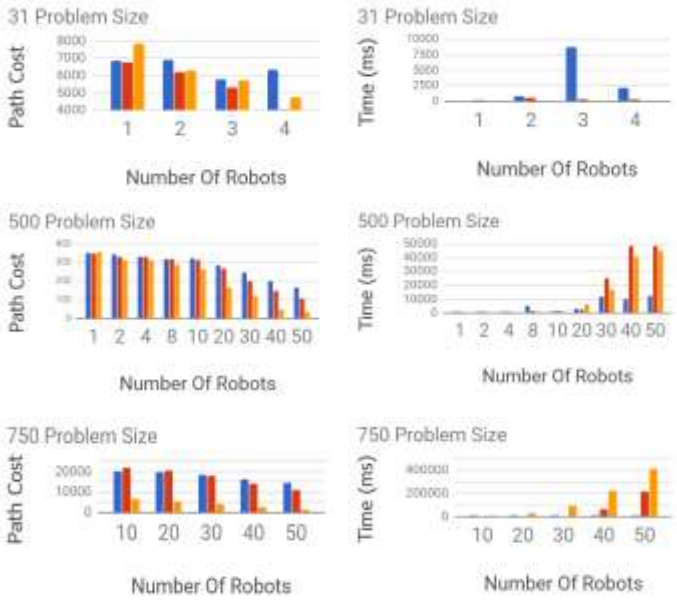

Fig. 5. Comparative analysis of the proposed appraoch with a master slave and a centralized approach.

As can be seen from Fig. 5, for less number of sites (eg. 31) the centralized algorithm has the minimum path cost. Hence, centralized works best for lesser number of mission sites. As the number of sites increase to 500, the following two prime trends can be observed from Fig. 5 (a) For number of robots less than 10, centralized and decentralized algorithm perform equally good with respect to path cost. (b) However as the number of robots increase beyond 10, decentralized outperforms the other two with respect to path cost. The rate of increase of difference of path cost between centralized and decentralized increases rapidly as number of robots increase.

For a problem size of 750 and 1000, a similar trend is observed. As the number of robots increase, decentralized algorithm outperforms the other two with respect to path cost. Also, the rate of increase of difference of path cost between centralized and decentralized increases rapidly as number of robots increase. The trends indicate the fact that as the problem gets very complex in terms of the volume of the search space, the heuristic allocation of mission sites is preferred over the random allocation of sites used in the centralized and master slave strategy. The master slave approach being the most randomized one, gives random results with respect to the path cost and time metrics and does not show regular trend. In general, its path cost is the worst among the three.

The results are also statistically compared for significance. The aim of the paper is to study scenarios with a very large problem size and therefore the statistical testing is done for the largest problem size of 500 mission sites. However, futuristic scenarios can have variable numbers of robots, and hence the testing is done for too few to too many robots. The number of robots are varied from 1 to 50. By statistical comparisons using F-test with 95% significance, it is asserted that the decentralized method surpassed the centralized and master-slave counterparts using the path cost metric for any number of robots.

The decentralized algorithm takes the largest time to compute for large problem sizes because it has to spend time in forming clusters using k-medoids for every task in the mission while the other algorithms don't have to. But this tradeoff may be ignored taking into account the fact that it provides a path cost which is much less that the other two algorithms. The decentralized algorithm breaks the complex problem into multiple easier problems, each of which is less likely to get trapped in a local minima, which overcomes the limitations of the loss of optimality due to the clustering heuristic.

IV. Conclusions

Having a fleet of sophisticated robots marks the future of warehouses, factories, industries, offices and homes, wherein each robot of the fleet will be sophisticated enough to carry a large variety of mobile manipulation tasks. In such a context it is eminent that the robots will be collectively given high end instructions from the collective large number of users of the environment. Such complex missions cannot be solved by model verification and heuristic search based techniques whose complexity is exponential to the number of variables used. The same hence motivates the use of evolutionary computation for the problem. Further, since the number of robots and mission sites are both large, decentralized techniques are needed.

In this paper the problem of mission planning was solved by using decentralized evolutionary computation. In terms of path cost metric, the proposed approach clearly outperformed both the centralized and a master-slave variant for problems of very large problem size. Further stopping the approaches on achieving a significant fraction of the convergent cost when computed for prolonged run gave a fair indication of the computation time. The computation time for the decentralized approach was a little higher due to the overhead of clustering. The approach can be extended to generalize the mission specification to take the LTL operators of until and next. Currently the assumption is that there is no contact constraint like getting a paper signed by multiple people cannot be distributed between multiple robots because that will require a transfer of the physical paper. The human robot interaction modules for specifying missions, interaction while catering to mission site, etc. can be further improved.

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

[5] 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.

[6] 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.

[7] M. Svorenova, J. Tumova, J. Barnat, I. Cerna, "Attractiion-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.

[8] 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.

[9] 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.

[10] T. Hussain, D. Montana, G. Vidaver, "Evolution-Based Deliberative Planning for Cooperating Unmanned Ground Vehicles in a Dynamic Environment", In: *Proceedings of the 2004 Genetic and Evolutionary Computation Conference, Lecture Notes in Computer Science*, vol 3103. Springer, Berlin, Heidelberg, pp. 1017-1029, 2004.

[11] G. B. Lamont, J. N. Slear, K. Melendez, "UAV Swarm Mission Planning and Routing using Multi-Objective Evolutionary Algorithms," In: *Proceedings of the 2007 IEEE Symposium on Computational Intelligence in Multi-Criteria Decision-Making*, Honolulu, HI, 2007, pp. 10-20.

[12] E. Eberbach, C. Duarte, C. Buzzell, G. Martel, "A portable language for control of multiple autonomous vehicles and distributed problem solving", In: *Proceedings of the 2nd International Conference on Computational Intelligence, Robotics and Autonomous Systems CIRAS*, pp. 15-18, 2003.

[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, "Sampling based Mission Planning for Multiple Robots," In: *Proceedings of the IEEE Congress on Evolutionary Computation, IEEE*, Vancouver, BC, pp. 662-669, 2016.

[16] 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, 1996

[17] R. Kala, "Homotopy conscious roadmap construction by fast sampling of narrow corridors," *Applied Intelligence*, vol. 45, no. 4, pp. 1089-1102, 2016.

[18] R. Kala, Supplementary video, Available at: https://youtu.be/JQG582wgeSw, Accessed: 1st May, 2018.