Dynamic Programming Accelerated Evolutionary Planning for Constrained Robotic Missions

Rahul Kala, ***Member, IEEE***

Citation: R. Kala, "Dynamic programming accelerated evolutionary planning for constrained robotic missions," 2018 IEEE International Conference on Simulation, Modeling, and Programming for Autonomous Robots (SIMPAR), Brisbane, QLD, 2018, pp. 81-86.
Full Text Available At: R. Kala, "Dynamic programming accelerated evolutionary planning for constrained robotic missions," 2018 IEEE International Conference on Simulation, Modeling, and Programming for Autonomous Robots (SIMPAR), Brisbane, QLD, 2018, pp. 81-86.

***Abstract*— Attributed to the increased automation, the day is not far wherein the robots will be seen doing a lot of sophisticated tasks, after which it is imperative that the offices and homes will have robots to replace the secretaries to be of common use for a large number of office-mates or house-mates. A mission comprises of a collection of high order tasks that a robot is asked to do with some logical and temporal constraints. The current approaches using model verification techniques have exponential complexity in terms of the number of variables, and are therefore not scalable to a very large level. The paper proposes a constrained mission specification language consisting of a sub-task as a logical relation between atomic tasks, a task as a collection of tasks to be performed one after the other, and a mission consisting of multiple tasks given by different users. An evolutionary approach is used to compute the solution to the mission that can scale to a very large number of variables. Problem specific heuristics are devised to compute a solution quickly. Particularly Dynamic Programming is used to align the solutions of multiple tasks to make a solution of a mission. Experimental results confirm that the proposed solution performs extremely well as compared to exhaustive search based approaches, model verification approaches and evolutionary approaches available in the literature. The results are demonstrated in simulations and on the Pioneer LX robot in the lab arena.**

I. INTRODUCTION

The problem of robot motion planning is classically seen as a solution to the problem "*Go from A to B avoiding obstacles*" [1,2], and the robots could be made to do a variety of tasks by repeated calls to classic motion planners. Todays, the robots are able to pick and place items, operate machines, deliver mails, get objects of interest, etc. Hence the human user no longer wishes to give instructions to the robot so as to solve the atomic problems. The intent is that the robot should be able to take high level instructions or missions and solve the same, and operate upon autonomously. Consider an example that some people may ask the robot to "*Get a letter signed by any 2 of the 3 coordinators, then the head, and get it back; simultaneously buy coffee from any of the coffee machines and check the store for stationary; also take the students' attendance, report the same to the divisional head, and then give the feedback to the students*". All this needs to be done by the robot autonomously without any human intervention. Here we do not consider the problem of manipulation, and only concentrate on scheduling the different tasks respecting the logical and temporal constraints specified.

Each of the atomic tasks hence is to visit a particular mission site σ to carry an operation. The classic motion planning problem may be used to compute for a trajectory $\tau(\sigma_i,\sigma_j):[0,1]\rightarrow C^{free}$ between any pair of mission sites σ_i and σ_j, such that $\tau(0)=\sigma_i$, $\tau(1)=\sigma_j$ and $\tau(s\in C^{free})\ \forall\ 0\leq s\leq 1$. The cost of the trajectory is given by $d(\sigma_i,\sigma_j)=\|\tau(\sigma_i,\sigma_j)\|$. Here C^{free} denotes the free configuration space that is a set of robot configurations such that the robot does not collide with any obstacle (or itself). If C is a space of all robot configurations and C^{obs} denotes the set configurations with collisions, $C^{free}=C\backslash C^{obs}$.

A typical way to solve the problem of mission planning is to specify the mission using Linear Temporal Logic and then to use model verification techniques to search for a solution [3,4]. The model verification techniques have computational complexity exponential to the number of propositional variables. An exhaustive search is enormously computationally expensive and infeasible. A sub-optimal solution may be returned reasonably early, however still the exponential computational complexity restricts the scalability of the approach. In a similar work Kress-Gazit [5] solved the problem specified as Linear Temporal Logic for single and multiple robots. The map was triangulated to get the transitions possible. Svorenova et al. [6] incorporated the use of preferences and rewards so as to keep the search directed towards the goal while maximizing the rewards. Lahijanian et al. [7] additionally incorporated probabilities into the mission specification, so that the executed mission satisfies the minimum probabilistic guarantee. The probabilities were leant using Reinforcement Learning. A robot catering to the requests of a very large number of robots will require significant number of variables, nullifying such approaches. Even if a better search techniques are utilized, the creation of a model to aid verification is still exponential in nature.

On the contrary, most set of instructions given to assistants have a structure that can be exploited to get

*Research supported by the Science and Engineering Research Board, Department of Science and Technology, Government of India through Research Grant ECR/2015/000406.

R. Kala is with the Robotics and Artificial Intelligence Laboratory, Indian Institute of Information Technology, Allahabad, India (phone: +91-532-292-2117; +91-7521050744; e-mail: rkala001@gmail.com).

computational betterment. The aim is hence to make a constrained language that enables the use of heuristics to verify a solution in polynomial time. Once a polynomial verification of a solution is possible, the major challenge is to use an algorithm to search for a solution in a non-exponential time, even though the space is exponential. The best option is to use Genetic Algorithms that is probabilistically complete and probabilistically optimal and can give good solutions in reasonable time even for a very large size of input. A blind search using Genetic Algorithm also faces the problem of high computational time, which can again be improved by exploiting the structure of the problem in such a constrained setting. The paper designs the use of various heuristics at different levels of the Genetic Algorithm for the same.

Specifically, Dynamic Programming has been used in the proposed approach to solve an 'alignment problem' that integrates multiple part solutions to make a complete solution. The Dynamic Programming operates inside the Genetic Algorithm to give an exact solution that the Genetic Algorithm would have taken a longer time while operating in a probabilistic nature. Using twin solvers of Genetic Algorithm and Dynamic Programming in a master slave mode, each technique solving the part of the problem for which it is most suited is a unique contribution of the work.

Evolutionary algorithms have been widely used to solve problems involving multiple goals that can be taken as examples of very specific missions. Xidias et al. [8] solved the problem of visiting a number of goals by a team of robots using centralized genetic algorithm. Kala [9] also used a centralized evolutionary framework to enable multiple robots visit a set of mission sites, with every site specifying a preference of robots. Xiao et al. [10] solved the classic problem of robot motion planning using an evolutionary framework. Similarly the use of a co-evolutionary framework for the same problem with a single or multiple robots is found in the works of Wang and Wu [11] and Kala [12]. The approaches are however for very specific missions of vising all the sites that has a small practical utility in service robotics.

The main contributions of the work are: (i) The proposed work first makes a constrained language for the mission, on the lines of the Linear Temporal Logic. The constrained language is rich enough for the specification of missions needed in daily life, while facilitates a polynomial verification without an overhead of the construction of an exponential model. (ii) An evolutionary framework is used for solving the problem that works for inputs of very large sizes. In order to make the evolution fast and directed, multiple heuristics are devised in the architecture of the genotype, initial population generation and use of genetic operators. (iii) Dynamic Programming is used along with Genetic Algorithm as a heuristic to give computational speedup. (iv) By experiments it is seen that the approach can give better performance as compared to model based approach, exhaustive search approaches and evolutionary approaches. The results are shown in simulations as well as on the Pioneer LX robot operating the combined classroom, lab and office area of the university.

II. Language for Mission Specification

This section presents the language to specify a robot mission, constrained so as to enable a polynomial time verification and the use of heuristics to search for a solution. A simple Boolean expression like "Get coffee from machine 1 or machine 2", "Get signatures from 2 of the 3 coordinators", etc. are called *sub-tasks*. An instruction to do one sub-task after the like "*Get signatures from 2 of the 3 coordinators then get the signature of either the head or the principle and then submit it to relevant office*" is called as a *task*. Multiple tasks to be performed like "*Get signatures from coordinators then from head and then deposit to office; simultaneously get coffee from either machine then buy cookies and then give to person A; simultaneously to check mails and then inform person B*" is called as a *mission*.

An atomic task that the robot may perform, requiring the robot to visit the site and perform the manipulation, is given by $\Diamond\sigma_i$, read eventually visit the site σ_i. A *sub-task* (ϕ) is a logical expression of the atomic tasks, given by the grammar (1)

$$\begin{aligned} &\phi ::= \sigma \\ &\sigma ::= \sigma \wedge \sigma \mid \sigma \vee \sigma \mid (\sigma) \\ &\sigma ::= \Diamond \sigma_1 \mid \Diamond \sigma_2 \mid \Diamond \sigma_3 \mid \Diamond \sigma_4 \mid \ldots\ldots \Diamond \sigma_n \end{aligned} \tag{1}$$

A *task* (ψ) is a collection of sub-tasks that must be performed serially one after the other. The sequence is however intrusive and anything may be performed in-between two sub-tasks or in-between a sub-task. The task definition is given by (2). A *mission* (χ) is simply a number of tasks given by a user or different users. All the tasks in a mission must be performed in any order by the robot, to complete the mission.

$$\psi = \Diamond(\phi_1 \wedge \Diamond(\phi_2 \wedge \Diamond(\phi_3 \wedge \ldots))). \tag{2}$$

$$\chi = \Diamond\psi_1 \wedge \Diamond\psi_2 \wedge \ldots\ldots\Diamond\psi_p. \tag{3}$$

The verification of a sub-task (φ) for a string *s* requires converting the sub-task into a postfix form, substitution all variables in φ by the Boolean constants (using hash maps) depending upon the presence in the string *s*, and then evaluating the expression, with a complexity of $O(|\varphi|+|s|)$, where $|\varphi|$ and $|s|$ denote the lengths of the sub-task and string respectively. The verification of a task (ψ) by a string *s* requires finding the smallest continuous sub-strings that satisfy a sub-task and to iterate upon the sub-tasks. Let υ_i denote the index at which the sub-task φ_i is satisfied, that is the sub-string $s(\upsilon_{i-1}+1:\upsilon_i) \models \varphi_i$, $\upsilon_0=0$. υ_i may be given by (4).

$$\upsilon_i = \begin{cases} \min_j \left(j : s(\upsilon_{i-1}+1 : j) \models \phi_i \right) & \begin{array}{l} if \upsilon_{i-1} \neq \infty, \\ s(\upsilon_{i-1}+1 : end) \models \phi_i, \\ i > 1 \end{array} \\ \infty & otherwise \end{cases}. \tag{4}$$

Computing $\upsilon_{|\psi|}$ iteratively and getting a non-infinite value means that the string $s \models \psi$. Moreover the sub-string $s(\upsilon_{|\psi|+1}:end)$ is unnecessary and may be trimmed. The complexity is hence $O(|\varphi|.|s|+|s|^2)$. Verification of a mission is simply verification of all the tasks that constitute a mission and the complexity is hence given by $O(|\chi|.|\varphi|.|s|+|\chi|.|s|^2)$

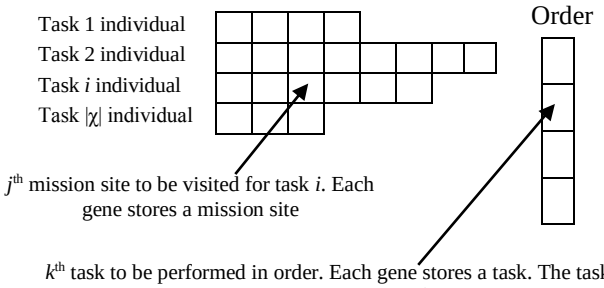
III. SOLUTION DESIGN

A. Evolutionary Algorithm

A naïve evolutionary algorithm may be used with the polynomial time verification technique of Sec. II to give a solution, however the computational requirements will still be very large due to not exploiting the structure of the problem. The evolutionary algorithm is hence customized.

The individual representation technique of the genetic algorithm stores the prospective solution of every task separately, and uses a variable sized array to store the prospective solution of a task. Even though the different tasks are not strictly solved one after the other, the individual representation specifies an order in which the tasks need to be solved as an additional array. First the algorithm sequentially places the tasks to make the solution of a mission, and then aligns the different tasks so as to get a better solution using Dynamic Programming. The details or order are given in Sec. III.C. The individual representation is shown in Fig. 1. The genes denoting a solution to a task do not necessarily denote a combinatorial problem as a task may require a robot to go to the office, collect papers, process them and submit them back to the office. Since the office is being visited twice, the problem is not combinatorial optimization. However there are partly characteristics of a combinatorial optimization problem to get a solution to a task. Accordingly, the genetic operators operating on a task are made to interpret the problem as combinatorial optimization to an extent of ω and as a standard parametric optimization problem to an extent of 1-ω. The order variable used is strictly combinatorial optimization in nature and hence its genetic operators.

Figure 1. Individual Representation



B. Initial Population Generation

The initial population is generated so as to be always feasible and respecting the logical and temporal constraints of the problem, for which the problem architecture is exploited. Let a task ψ consist of the unit sub-tasks φ. Let *var*(φ) denote the variables used in the sub-task φ. In the worst case, if the robot visits all mission sites *var*(φ), the sub-task will always be satisfied. Of course, most of these will be 'or' separated and visiting all places may be unnecessary. Hence the task component of an individual is given by a random permutation of *var*(φ). The verification scheme for task trims any excess string and any individual generated with the optimal number of mission sights followed by excess sites will be trimmed and gain an optimal structure. The order component is again a random permutation of all tasks possible. Algorithm 1 summarizes the individual representation scheme.

Algorithm 1: Random individual generation

$X_\chi \leftarrow$NULL
for all tasks ψ in mission χ
  $X_\psi \leftarrow$NULL
  for all φ in ψ
    $X_\varphi \leftarrow$random_permutation(var(φ))
    List append all σ in X_φ to X_ψ
  add X_ψ to X_χ
$X_O \leftarrow$random_permutation(1,2,3… |ψ|)
return < X_χ, X_O>

C. Genetic Operators

Apart from the standard operators, two operators need a special mention for the nature of the problem. The Crossover operator is modified to account for the variable sized individual and part combinatorial style of operation. While using the combinatorial variant of the crossover, the standard crossover as applied in combinatorial optimization cannot be used since a site may have multiple occurrences in reality. The principle used in crossover is that the total number of occurrences of the mission sights in both parents should match with the respective children. Hence, the number of occurrences of the mission sights are noted in both parents. After copying half the individual using one-point crossover, only those genes are copied from the other half of the individual that do not exceed the number of occurrences allowed. The remaining occurrences are randomly permutated and appended in the respective children. The crossover of the order part is a simple combinatorial optimization crossover. The crossover technique is shown as Algorithm 2. If *X* denotes an individual, X_χ refers to the mission component, X_O denotes the order component and X_ψ denotes the task component.

Algorithm 2: Crossover(X,Y)

$C_\chi \leftarrow \emptyset$, $D_\chi \leftarrow \emptyset$
for all tasks ψ in mission χ
  $C_\psi \leftarrow \emptyset$, $D_\psi \leftarrow \emptyset$
  with probability ω, combinatorial crossover
    $M_\psi(\sigma) \leftarrow$count of σ in X_ψ
    $N_\psi(\sigma) \leftarrow$count of σ in Y_ψ
    for i from 1 to len(X_ψ)/2,
      add X_ψ(i) to C_ψ, $M_\psi(X_\psi(i)) \leftarrow M_\psi(X_\psi(i))$ -1
    for i from 1 to len(Y_ψ)/2,
      add Y_ψ(i) to D_ψ, $N_\psi(Y_\psi(i)) \leftarrow N_\psi(Y_\psi(i))$ -1
    for i from len(X_ψ)/2+1 to len(X_ψ)
      if $M_\psi(X_\psi(i))>0$,
        add X_ψ(i) to D_ψ, $M_\psi(X_\psi(i)) \leftarrow M_\psi(X_\psi(i))$ -1
    for i from len(Y_ψ)/2+1 to len(Y_ψ)
      if $N_\psi(Y_\psi(i))>0$,
        add Y_ψ(i) to C_ψ, $N_\psi(Y_\psi(i)) \leftarrow N_\psi(Y_\psi(i))$ -1
    $L_1 \leftarrow \emptyset$, $L_2 \leftarrow \emptyset$
    for all i in M_ψ
      while $M_\psi(i)>0$, add i to L_1, $M_\psi(i) \leftarrow M_\psi(i)$-1
    for all i in N_ψ
      while $N_\psi(i)>0$, add i to L_2, $N_\psi(i) \leftarrow N_\psi(i)$-1
    add random permutation of L_1 to C_ψ
    add random permutation of L_2 to D_ψ

  with probability 1-ω, scattered crossover
    for i from 1 to max(len(X_ψ),len(Y_ψ))
      if rand<0.5
        if i ≤ len(X_ψ), add X_ψ(i) to C_ψ,
        if i ≤ len(Y_ψ) add Y_ψ(i) to D_ψ
      else
        if i ≤len(X_ψ) add X_ψ(i) to D_ψ
        if i ≤ len(Y_ψ) add Y_ψ(i) to C_ψ
  add C_ψ to C_χ, add D_ψ to D_χ
C_O, $D_O \leftarrow$combinatorial one point crossover of X_O,Y_O
return < C_ψ, C_O >, < D_ψ, D_O >

Similarly mutation operator is designed. Two mutations are used for changing the individual structure that is deleting an existent site and adding a random site (out of variables in var(φ) for all φ in selected ψ) at a random position. Further, two parametric mutation operators are used, a combinatorial version that swaps two mission sites, and a parametric version that replaces a gene with a random site (out of variables in var(φ) for all φ in selected ψ). The technique is shown as Algorithm 3.

A couple of other concepts used in the algorithm deserve a mention (i) The algorithm is also given a memetic framework by incorporation of a local search algorithm that behaves very similar to the mutation operator. (ii) The fitness evaluation as detailed below requires a huge computational load, and the chances of repetition of individuals is very large. Therefore a hash table is made that maps the individuals with their fitness values. If the fitness evaluation is made on an individual that was generated earlier, the hash table directly returns the value.

Algorithm 3: Mutation(X)
ψ←random task ψ in mission χ
r_1←random gene in X_ψ, r_2←random gene in X_ψ
r_3←random gene in X_O, r_4←random gene in X_O
φ←random sub-task in ψ, σ←random variable in var(φ)

with probability p_1, delete $X_\psi(r_1)$ from X_ψ
with probability p_2, insert σ_r to X_ψ at position r_1
with probability p_3, swap($X_\psi(r_1)$, $X_\psi(r_2)$)
with probability p_4, $X_\psi(r_1) \leftarrow \sigma_r$
with probability p_5, swap($X_O(r_4)$, $X_O(r_5)$)
return X

Overall, Stochastic Universal Selection scheme with rank based expectation is used. Crossover and mutation operators make the children population which combines with the parent population to compete for existence, to be used as for the next generation.

D. Fitness Evaluation and Dynamic Programming Acceleration

The first phase of the fitness evaluation checks for the validity of the tasks that constitute a mission. The validity may simply be computed using (5). Even though the individuals were initially inserted ensuring feasibility, it is possible that crossover and mutation operators might eventually make an individual infeasible. Penalty is added for the number of sub-tasks in a task that are not satisfied by the genetic individual. The penalties of all tasks are added to compute the total penalty of a mission. Additionally the first phase trims any extra part of the task component of the individual. Even though the strings of the largest size were initially generated, many of these strings will get trimmed to smaller lengths while being still feasible. The algorithm is given as Algorithm 4. The variable *i* in the algorithm counts the total number of sub-tasks satisfied by the individual.

The first phase produces trimmed individuals associated with any task. Finally the robot needs a linear string consisting of all mission sites in the order that they should be visited. The individual consists of multiple strings, each string that solves a task. Let $X_\psi=[a_1,b_1,c_1]$be the solution of the task ψ_x and let $Y_\psi=[d_2, e_2]$be the solution of the task ψ_y. Any combination of both the strings is a valid solution of mission consisting of ψ_x and ψ_y, as long as the sub-string X_ψ and sub-string Y_ψ in the resultant string are found in the same order. So strings $[a_1,b_1,c_1,d_2, e_2]$, $[d_2, e_2, a_1,b_1,c_1]$, $[a_1,b_1, d_2, c_1, e_2]$, $[a_1,b_1, d_2, e_2, c_1]$, etc. are all valid solutions of $\chi = \Diamond\psi_x \wedge \Diamond\psi_y$.The problem of missions incorporating two tasks is hence to align the individual solutions of X_ψ and Y_ψ such that a fused solution optimally solves the mission given optimal solutions of tasks. Dynamic Programming is used to solve the same alignment problem. Let d_{ij} denote the shortest distance in aligning the sub-strings $X_\psi(1:i)$ and $Y_\psi(1:j)$. Correspondingly let π_{ij} be the trace of path that stores 1 if X_ψ contributes the last mission site in the combined sub-solution and 2 if Y_ψ contributes the last mission site. Let δ_{ij} be the last mission site in the combined sub-solution. The metrics are given by (5)-(7). It may be seen that the formulation remains intact even if either or both of the input solutions were of part missions instead of individual tasks.

$$d_{ij} = \begin{cases} 0 & \text{if } i = 0, j = 0 \\ d_{i-1,j} + c\left(\delta_{i-1,j} + X_\psi(i)\right) & \text{if } i > 0, j = 0 \\ d_{i,j-1} + c\left(\delta_{i,j-1} + Y_\psi(j)\right) & \text{if } i = 0, j > 0 \\ \min\left(d_{i-1,j} + c\left(\delta_{i-1,j} + X_\psi(i)\right), d_{i,j-1} + c\left(\delta_{i,j-1} + Y_\psi(j)\right)\right) & \text{if } i > 0, j > 0 \end{cases} \quad (5)$$

$$\pi_{ij} = \begin{cases} 0 & \text{if } i = 0, j = 0 \\ 1 & \text{if } i > 0, j = 0 \\ 2 & \text{if } i = 0, j > 0 \\ 1 & \text{if } d_{i-1,j} + c\left(\delta_{i-1,j} + X_\psi(i)\right) \le d_{i,j-1} + c\left(\delta_{i,j-1} + Y_\psi(j)\right), i > 0, j > 0 \\ 2 & \text{if } d_{i-1,j} + c\left(\delta_{i-1,j} + X_\psi(i)\right) > d_{i,j-1} + c\left(\delta_{i,j-1} + Y_\psi(j)\right), i > 0, j > 0 \end{cases} \quad (6)$$

$$\delta_{ij} = \begin{cases} Source & \text{if } i = 0, j = 0 \\ X_\psi(i) & \text{if } i > 0, j = 0 \\ Y_\psi(i) & \text{if } i = 0, j > 0 \\ X_\psi(i) & \text{if } d_{i-1,j} + c\left(\delta_{i-1,j} + X_\psi(i)\right) \le d_{i,j-1} + c\left(\delta_{i,j-1} + Y_\psi(j)\right), i > 0, j > 0 \\ Y_\psi(i) & \text{if } d_{i-1,j} + c\left(\delta_{i-1,j} + X_\psi(i)\right) > d_{i,j-1} + c\left(\delta_{i,j-1} + Y_\psi(j)\right), i > 0, j > 0 \end{cases} \quad (7)$$

If $|\sigma|$ is the maximum number of mission sites in any task, the complexity to align two tasks into a fused solution is $O(|\sigma|^2)$. The complexity to align all tasks in a mission is $O(|\sigma|^{|\chi|})$, where $|\chi|$ is the number of tasks in a mission. The complexity is clearly exponential in terms of $|\chi|$ which may be very large because of a very large number of users and hence the same cannot be accepted. In order to reduce the complexity into polynomial, we take *k* tasks at a time, fuse them to make a combined solution. Having this solution of a sub-mission, we further take *k*-1 more tasks and fuse it to the solution built so far. The step is repeated until the solutions of all tasks are aligned with the common solution. In such a manner the complexity of the approach reduces to $O\left(\left\lceil |\chi|/k - 1 \right\rceil |\sigma|^k\right)$. However now the order of the presentation of tasks to make a fused solution may be important and may affect optimality. Hence the *order* terms were taken into consideration while formulating the genetic individual.

Algorithm 4: Fitness Evaluation(X)

Phase I: Validity Computation

penalty←0

for all tasks ψ in mission χ

  i←1 //*The sub-task being worked over*

  υ_i←1, $s_{\varphi i}$←∅

  for all mission site σ in X_ψ

    add σ to s_ψ,

    if s_ψ |= φ_i

      i ←i+1, υ_i← υ_{i-1}, $s_{\varphi i}$←∅

      if i=|ψ|+1, trim X_ψ till $\upsilon_{|\psi|}$, break

    υ_i←υ_i +1

  if i<|ψ|+1, penalty←penalty+(|ψ|-i+1)

Phase II: Dynamic Programming

X_ψ← first task in X as per order X_O

for all tasks Y_ψ in X as per order X_O

  calculate d, π and δ using (6-8)

  τ←∅

  <i,j>←<|X_ψ|, |Y_ψ|>

  cost←d_{ij}

  while <i,j>≠<0,0>

    if π_{ij}=1, add X_ψ(i) to τ, <i,j>←<i-1,j>

    else add Y_ψ(j) to τ, <i,j>←<i,j-1>

  reverse(τ), X_ψ← τ

return <cost,X_ψ>

The resultant approach is a fusion of Genetic Algorithm and Dynamic Programming. The Genetic Algorithm produces good solutions to tasks, while the Dynamic Programming aligns the tasks so as to produce an optimal linear sequence of mission sights that the robot can follow. The Genetic Algorithm would take indefinitely long to solve the alignment problem that can best be solved by using Dynamic Programming. Similarly the Dynamic Programming would require indefinitely long to solve the complete mission with an exponential complexity, and hence the aspect of the problem is best solved by using Genetic Algorithm.

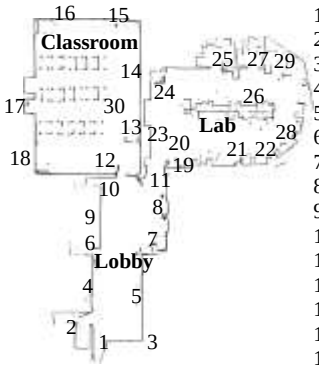
Since the primary intent is Genetic Algorithm, in order to give more share of the computation time to Genetic Algorithm *k* is kept as 2. The complexity of Dynamic Programming hence becomes $O(|\chi||\sigma|^2)$. The algorithm as a pseudocode is shown as Algorithm 4.

IV. RESULTS

A. Experimental Results on Pioneer LX Robot

The algorithm is tested using the Pioneer LX robot. The experiments are done in the arena of the Robotics and Artificial Intelligence Laboratory of the institute consisting of a lab area, classroom area, offices and a lobby. The robot was initially made to go around on a tour guided by a human using a joystick. The robot makes a map of the complete arena using SLAM on the laser data, also using IMU and wheel encoders. As the robot moved, it was pointed the different mission sites σ which were recorded by the robot. The map along with the different mission sites are shown in Fig. 2.

Figure 2. Map along with Mission Sites

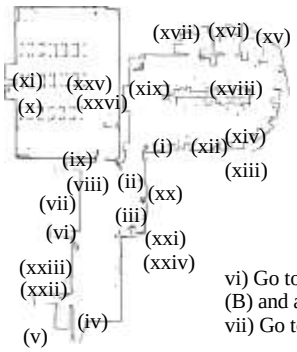


1) Office GCN
2) Office RK
3) Robita entrance gate
4) Office Mary
5) Waiting Area
6) Cabin Joey
7) Copying Station
8) Office PC
9) Cabin Ross
10) Classroom Entrance
11) Robita lab entrance
12) Light switches
13) Classroom System
14) Whiteboard
15) Wall AC front
16) Wall AC Back
17) Back Right AC
18) Back Left AC
19) Charging Station
20) Robot Home
21) Racheal
22) Monica
23) Left Cupboard
24) Right Cupboard
25) Divya
26) Latika
27) Priyanshu
28) Heena
29) Bishnu
30) Projector

The map so produced was exported as a bitmap. In order to compute the cost matrix $d(\sigma_i,\sigma_j)$, Probabilistic Roadmap [13] technique was used. The specific technique used [14, 15] uses a hybrid of obstacle-based, narrow corridor and uniform sampling strategies, and connecting them to nearest neighbors. Additional vertices are made near obstacles when a connection attempt fails. Edge connection strategies include a hybrid of connecting connected components, expansion of a leaf node outward using Expansive Search Trees, RRT style expansion, and maximizing area coverage by a hash based density equalization.

As the first mission, the robot is instructed by numerous people. A to E denote the tasks, the Arabic numbers denote the mission sites, and the small roman numbers are the order of visit of the mission sites. **(A)** The supervisor RK instructs the robot to get the request of an equipment verified by any two of his students, namely, Bishnu (29), Heena (28) and Racheal (21); and to further take the recommendation from any of the two postdocs namely, Ross (9) and Priyanshu (27). The robot should come with the signed document to Office RK (2). **(B)** Meanwhile, the person GCN has been reported a serious discipline issue by his students and calls the robot to his office (1) to get a document to be first noted by Joey (6) and thereafter delivered and signed receiving by all his students Monica (22), Divya (25) and Latika (26). The robot must finally take a photocopy of the receiving at the copying station (7), submit the photocopy to the office RK (2) and then give the original copy to the secretary's office Mary (4). **(C)** The user Priyanshu (27) also calls the robot to collect his stipend sanction paper, get his approved leave printout at the office system (8), to be finally signed by the guide RK (2). The robot must also make a copy at the copying station (7).

Figure 3. Map along with Mission Sites



i) Start from source (12)
ii) Goto ROBITA lab entrance (11) for inspection (D)
iii) Goto waiting area (5) for inspection (E)
iv) Goto ROBITA entrance gate (3) for inspection (E)
v) Go to GCN office (1) to get the disciplinary paper (B) and for attendance (F)
vi) Go to Joey (6) to get disciplinary paper noted (B) and attendance (F)
vii) Go to Ross (9) for attendance (F)
viii) Go to classroom entrance (10) for inspection (E)
ix) Switch on the lights (12) for seminar preparation (D)
x) Turn on Back Left AC (18) for seminar preparation (D)
xi) Turn on Back Right AC (17) for seminar preparation (D)
xii) Go to Rachael (21) for equipment verification (A) and attendance (F)
xiii) Go to Monica (22) to get the disciplinary document signed (B) and attendance (F)
xiv) Go to Heena (28) to get the equipment verification (A) and attendance (F)
xv) Go to Bishnu (29) for equipment verification (A) and attendance (F)
xi) Go to Priyanshu (27) for post-doc verification of equipment (A), collect Priyanshu's stipend paper (C) and attendance (F)
xvii) Go to Divya (25) for disciplinary document signing (B) and attendance (F)
xviii) Go to Latika (26) for disciplinary document signing (B) and attendance (F)
xix) Go to right cupboard (24) to take the marker (D)
xx) Go to the office system (8) to take Priyanshu's leave printout (C)
xxi) Go to copying station (7) to get a copy of the stipend document (B)
xxii) Go to RK (2) to give the equipment letter (A), give the disciplinary letter photocopy (B), get the stipend document signed (C) and for attendance (F).
xxiii) Go to Mary's office (4) to give original disciplinary papers (B), give security inspection report (E) and attendance (F)
xxiv) Go to copying station (7) to get Priyanshu's stipend copied (C)
xxv) Switch on the projectior (30) for classroom preparation (D)
xxvi) Wipe off board (14) and put the marker (D)

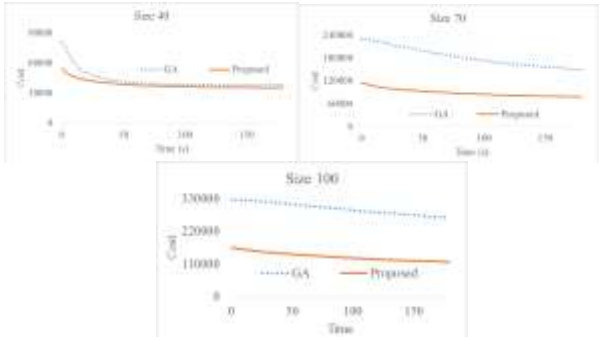
(D) The robot is also simultaneously asked to prepare the classroom for a seminar, so the robot must switch on the lights (12), then switch on any two ACs out of the wall AC front (15), wall AC back (16), back right AC (17), back left AC (18). Then the robot must go to the left cupboard (23) or the right cupboard (24) to get a marker, and must return to the whiteboard (14) to wipe off the board and place the marker and to also switch on the projector (30). **(E)** The robot is also to ensure the security and make inspections to the Classroom Entrance (10), ROBITA entrance gate (3), waiting area (5) and Robita lab entrance (11) and then to report all findings in person to office of the secretary, Mary (4). **(F)** Finally, the robot must take attendance of all people, including RK (2), GCN (1), Mary (4), Joey (6), Heena (28), Bishnu (29), Racheal (21), Monica (22), Divya (25), Latika (26), Priyanshu (27) and Ross (9).

The output sequence of the robot is shown in Fig. 3. The steps taken by the robot are as follows. The results are also given as a video available online.

B. Comparisons

The proposal was to use evolutionary computation so as to overcome the limitations of the current search based algorithms due to exponential complexity. The first proposal towards the same was to use optimal search algorithm, Uniform Cost Search. The search gave optimal answers till a problem size of 15, after which it was not possible to continue due to memory constraints. The Iterative Lengthening Algorithm is a linear memory variant of Uniform Cost Search that was tried. The algorithm again could go upto the problem size of 14 due to exponential complexity. Thereafter model verification techniques were adopted using NuSMV library, sacrificing the optimality parameter. The technique gave results of very poor cost very quickly till a problem size of 40, after which the performance depended significantly on the problem size and the time could vary from a few seconds to 23 minutes. After a problem size of 59, the algorithm consistently failed to generate solutions even after an hour of computation. The experimental comparisons are done using Genetic Algorithm. A naïve Genetic Algorithm would have performed very poorly, and hence all problem specific heuristics besides Dynamic Programming are added in the solution. This includes the part combinatorial nature of the problem, heuristic initial population generation and variable length individual representation. The genotype was kept linear. The comparisons for different problem sizes is shown in Fig. 4. It can be clearly seen that the algorithm performs better than Genetic Algorithm, showing faster convergence and better convergent cost attributed to the Dynamic Programming accelerations.

Figure 4. Comparions with Genetic Algorihtm.



V. CONCLUSIONS

The need and utility of robots is continuously increasing which showcases a promising future wherein a robot will be taking care of a very large number of inmates in home and office environments, giving rise to very complex missions that the robot must be able to solve. The paper proposed the use of Evolutionary Computation for the same that is scalable for a very large number of robots. A constrained language was designed so as to enable design of heuristics to be injected into the Genetic Algorithm. Specifically Dynamic Programming was used to align the solutions of different sub-tasks to produce a solution of a task. The results are demonstrated on very complex problems involving upto 100 variables. It was seen that the algorithm could easily solve the problems. Results were also shown on the Pioneer LX robot. Modelling of preferences between tasks and users, interaction with the user, use of multiple robots, designing time constrained missions and solving the problem incrementally will be the goal for future research.

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

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

[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. 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.

[7] 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.

[8] 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.

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

[10] 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.

[11] M. Wang & T. Wu, "Cooperative co-evolution based distributed path planning of multiple mobile robots", *Journal of Zhejiang University Science*, vol. 6A, no. 7, pp. 697-706, 2005.

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

[13] 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.

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

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