Time Bound Robot Mission Planning for Priority Machine Using Linear Temporal Logic for Multi Goals

Venkata Beri(✉), Rahul Kala, and Gora Chand Nandi

Robotics and Machine Intelligence Laboratory,
Indian Institute of Information Technology, Allahabad, Allahabad, India
venkat.beri@gmail.com, rkala001@gmail.com,
gcnandi@gmail.com

Abstract. In this paper, we implement a Linear Temporal Logic-based motion planning algorithm for a prioritized mission scenario. The classic robot motion planning solves the problem of moving a robot from a source to a goal configuration while avoiding obstacles. This problem of motion planning gets complicated when the robot is asked to solve a complex goal specification incorporating boolean and temporal constraints between the atomic goals. This problem is referred to as the mission planning. The paper assumes that the mission to be solved is a collection of smaller tasks, wherein each task constituting the mission must be finished within a given amount of time. We assign the priorities for the tasks such that, the higher priority tasks should be completed beforehand. The planner solves the missions in multiple groups, instead of the classic approach of solving all the tasks at once. The group is dynamic and is a function of how many tasks can be incorporated such that no time deadline is lost. The grouping based prioritized and time-based planning saves a significant amount of time as compared to the inclusion of time information in the verification engine that complicates the search logic. NuSMV tool is used to verify the logic. Comparisons are made by solving all tasks at once and solving the tasks one-by-one. Experimental results reveal that the proposed solver is able to meet the deadlines of nearly all tasks while taking a small computation time.

Keywords: Linear temporal logic · Mission planning · Robot mission planning · Model checking · NuSMV

1 Introduction

The problem of robot motion planning is to empower a robot to explore and advance out in a confused hindrance or an obstacle-prone environment. A number of technology applications for *mission planning* and self-governing frameworks [1], (e.g., autonomous vehicle, unmanned air vehicles) require effective procedures that can produce the desired sequence of operations to be done to achieve the user defined mission, correctly and effectively. Over the period of time, the idea of correctness and efficiency has become more and more sophisticated. This results in providing tasks consisting of many other sub tasks with Boolean and temporal constraints, rather than a single goal

N. Kumar and R. Venkatesha Prasad (Eds.): UBICNET 2019, LNICST 276, pp. 250–263, 2019.
https://doi.org/10.1007/978-3-030-20615-4_19

problem constituting the *classical motion planning* problem. The complex goals and several environmental constraints can make a robot wander infinitely in an environment for a given problem statement.

This paper consequently solves the high-level mission planning problem, where the mission is given as linear temporal logic (LTL) formula and develops controllers for the same. It generates the path for the robot which satisfies the given LTL formula. One of the qualities of this system is that it considers assignments in which the conduct of the robot relies upon the data it assembled at runtime. If the task is feasible, then the LTL will be generating a sequence so as to give a path to the robot that accomplishes the mission. There are bisimulation methods which provide a finite transition system to provide an optimal run [2]. The model checking tools NuSMV [3, 4] and LTLMoP [5], provide us the guarantee that the specifications can be guaranteed.

Temporal logic has been used for many complex specifications [6–8]. We construct the problem into a map by partitioning the workspace of the robot [9] and formulating an LTL for its desired behavior. This paper describes the problem of solving a mission planning with the help of LTL specified goals. Metric Temporal logic is another such instance used for obtaining the results in the desired way [10]. Temporal logic has been used for sampling-based motion planning where a multi layered approach has been presented [11]. A mechanism to deal with the large computation time of Temporal Logic is also to use restricted languages and evolutionary techniques to generate iterative solutions [12, 13].

Solving the problem of mission planning hence requires representing the mission as a LTL formula and solving the same using any LTL based verification system. There are two problems that occur in this methodology. First, that the verification of a solution in LTL has an exponential complexity and therefore there is a limited size that the formula can have. Second, the real-life missions can be very complex, requiring time for the robot to compute a solution, however the real-life missions are also real time in nature and beyond a time the solution is needless as the facts requiring the solution may have changed (like the need to have a coffee in a meeting) or a person may volunteer to himself/herself carry the job without the robot.

A typical way of solving the problem is by incorporating timing information associated with the missions and its components thereon for verification. However, the robot first plans and then executes the mission, and the time incurred in the planning stage is significantly high and cannot be assumed to be zero that the current approaches assume. Further, adding time information complicates the search space for a solution in the verification engine and this severely complicates the time.

In this paper, the same problem is solved by decomposition. We assume that a mission is composed of smaller sub-missions called tasks. Each task has its own priority and hence a time within which the task must complete. The search incrementally groups tasks to make a mission, such that the total incurred computation time and expected processing time is within the threshold of time and the solution is hence useful. The grouping also ensures that the number of propositional variables in the LTL formula are small and the computation time as per the exponential complexity of LTL is hence limited.

This paper has been discussed as per the following sections. Section 2 provides a brief about Linear temporal logic. Section 3 discusses the proposed methodology. Section 4 details about the results and simulations. Finally, Sect. 5 concludes the paper.

2 Linear Temporal Logic

Linear temporal logic defines set of logic operators which are bound with a factor of time in contrast to the traditional Boolean logic. The linear temporal logic consists of the operators such as eventually ($\Diamond$), next ($\Box$), always ($\bigcirc$) and until (U). These operators are further described by the assumptions such as safety, liveliness, sequencing and reachability.

The typical LTL operators used are notably used as: Next or $\bigcirc\phi$, meaning ϕ is true in the next moment in time; Always or $\Box\phi$, meaning ϕ is true in all future moments of time; Eventually or $\Diamond\phi$, meaning ϕ is true in some future moment; and Until or ϕ U ψ, meaning ϕ is true until ψ is true. The LTL has some properties with environment assumptions these include:

Safety: Which describes the condition which must be always satisfied, for example like "Always avoid obstacles". The LTL formula negation ($\neg$) is used to describe the conditions $\neg(O_1 \vee O_2 \vee O_3 \vee \ldots \vee O_n)$ U R means eventually reach region R by avoiding all obstacles O_i, $i = 1, 2, \ldots$ n.
Liveness: This specifies that goals which always must be eventually satisfied by some actions in the future (e.g. "if region A is visited then must visit region B infinitely often").
Sequencing: This describes the sequence of goals to be followed in any order. The LTL formula $\Diamond$ specifies that we must visit all regions (i.e. R_1, R_2, R_3, R_4, R_5) as per requirement $\Diamond\ (R_1 \wedge \Diamond\ (R_2 \wedge \Diamond\ (R_3 \vee \Diamond\ R_4)))$ in given environments.
Reachability: An individual state assume is reachable from any initial present states. Moreover, this kind of goal specification is more useful when the problem is similar corresponds to a single/multi goals without any complex specifications.

3 Proposed Methodology

3.1 Overall Solution Design

The general approach for motion planning comprises of generating the graphical image or representation of the given map from the environment as based on the conditions, mission specification, navigation, control and planning shown in Fig. 1.

Mission Specification describes how we achieve the end goal. For example, it can be to reach a point B from Point A without reaching a particular point C. Mission Planner tells on an overlay how this specific goal can be achieved. Low level planning involves various strategies. The control specification lets us control the robot in the specific directions with the concerned motion.

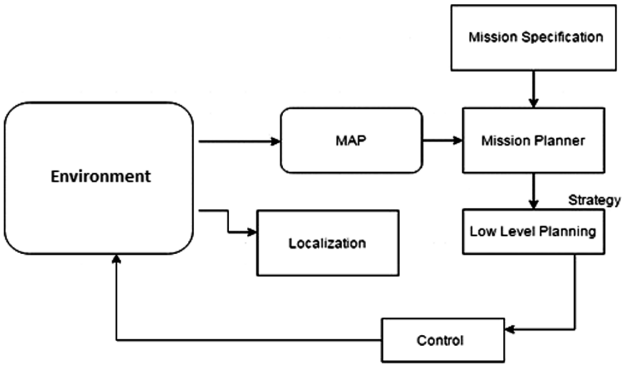

Fig. 1. Overall solution design

Mapping involves looking at the real-world as it is. It depicts the arrangement of the system in the real-world. Localization tells us where the particular object is with respect to the map. It takes into consideration the robots' exact coordinates. The entire real-world system or experimental scenario is represented using a map. Map of real-life environments may 2D or 3D. The map conversion from real-world to workspace environment can happen through camera calibrations, distance information of obstacle to regions, navigable path area, etc.

Path planning can be done in a structured environment or in an unstructured environment. In a structured environment, the obstacles are represented using polygon like figures whereas in the unstructured environment, the size and the shape of the obstacles are not available. This will generate plans only for valid for the LTL specifications and synthesis of the LTL specifications.

3.2 Triangulation

While representing the real-world system for our navigation map in workspace configurations, we first need an efficient technique for representing the complicated world system. Thus, we need to redesign the real-world environmental information for a robotic map. The path from source to goals is fully dependent on how the real-world system is represented.

The original map is assumed to be known with polygons acting as the obstacles. The robot moves in free areas and therefore the free areas are modelled by using triangulation. Triangulation involves the decomposition of the regions into a polygonal area or rather a set of triangles [9, 14]. The area classified as the regions covered by the obstacles which is referred as holes, which is required so that no triangulation happens inside the obstacles, whereas the rest of the region lies to be under triangulation. Each of these triangles or regions serve as a well-connected graph, so that the transitions can be done easily. We consider the map shown in Fig. 2 for all the experiments, which represents a typical map of any home or office scenario. The same map will be used for discussions. The regions of interest are labeled. The given map was triangulated and the result is shown in Fig. 3.

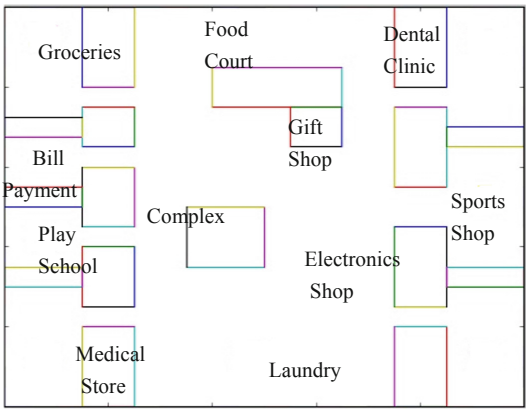

Fig. 2. Pictorial representation of the workspace

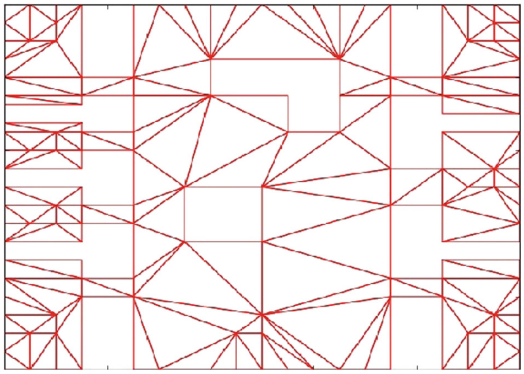

Fig. 3. Triangulated region of the workspace

4 Mission Solver

The main aspect of the problem is solving for a mission. Assume that the mission is given by ψ. In our case we assume that the mission is composed of a group of tasks of the form $\psi = \cup\{<\phi_i, \pi_i, \theta_i>\}$, where ϕ_i is the i^{th} task specified as a LTL, π_i is the priority of the i^{th} task and θ_i is the time within which the task must be completed, otherwise it will be useless to do the task. In the actual implementation the priorities and time thresholds are taken to be related in the sense that a higher priority is given to the tasks with a small time threshold, that is π_i, = $rank(\theta_i)$, however in general the two may not always be related and the algorithm is generic for the same.

The proposed algorithm operates in a greedy manner and takes up tasks in the increasing order of priority. Therefore, the higher priority tasks get solved first and have a lesser chance of losing the deadline. The assumption here is that a hard prioritization is followed and a better priority task cannot be compromised to any number

of smaller priority tasks. However, the time threshold is soft, meaning that even if a task cannot be completed on time, it must still be completed as early as possible. A group G is defined as an ordered sequence of tasks in the prioritized order, given by $G^k = \cup[<\phi_i, \pi_i, \theta_i> : \pi_i \leq \pi_{i+1}, \pi_j \leq \pi_0 \vee \pi_j \geq \pi_{|G|} \forall <\phi_i, \pi_i, \theta_i> \notin G^k]$. Here G^k denotes the k^{th} group. Let $T^k = t_E + t_P + T^{k-1}$ be the total travel time for the robot to cater to simultaneously solve all the tasks specified in G^k (say execution time, t_E), the time incurred in computing such a plan and division of the group (say planning time, t_P) and the time incurred in planning and executing all previous groups (say T^{k-1}). The constraint is that either all tasks in G^k must adhere to their time thresholds, which is the best possibility; however, in case a task cannot meet its time constraint, then it must be the first and only task in the group so as to specify that there is no way to solve the task within the time threshold, given by $T^k < \max_{j \in Gk} \theta_j \vee |G^k| = 1$. Obviously, each task is solved only once ($G^k \cap G^l = \varnothing \forall k \neq l$) and all tasks are solved ($\exists k: <\phi_i, \pi_i, \theta_i> \in G^k$).

The algorithm is then to iteratively build up the groups. Given the group G^{k-1}, the formulation of the group G^k involves iteratively adding in tasks $<\phi_i, \pi_i, \theta_i>$ to G^k until the summation of times $t_E + t_P + T_{k-1}$ is within threshold $\max_{j \in Gk} \theta_j$. A minimum of 1 task is compulsorily added. t_P is the continuous summation over all times in the iterative addition process. Thereafter the group is executed and the information is used for designing the next group. Algorithm 1 gives a pseudo-code of the process.

Algorithm 1: Mission Solver

1: $T^0 \leftarrow 0$, $k \leftarrow 1$
2: $G_1 \leftarrow \emptyset$
3: $t_P \leftarrow 0$
4: for all tasks $<\phi_i, \pi_i, \theta_i>$ in sorted order of priority
5: add $<\phi_i, \pi_i, \theta_i>$ to G^k
6: $S_k \leftarrow$ Plan(G_1) with computation time t
7: $t_P \leftarrow t_P + t$
8: if $t_P + t_E(S_k) + T^{k-1} > \max_{j \in Gk} \theta_j$
9: if $|G^k| > 1$
10: remove $<\phi_i, \pi_i, \theta_i>$ from G^k
11: $t^k \leftarrow t_P + t_E(S_{k-1}) + T^{k-1}$
12 else $t^k \leftarrow t_P + t_E(S_k) + T^{k-1}$
13: $k \leftarrow k+1$, $G_k \leftarrow \emptyset$
14: Return G^k

5 Results and Discussions

For experimentation, we took the tasks given in Table 1 as per the priority assigned to them in the increasing order, the top being more priority. We started off with the tasks and carried out one after the other. NuSMV was used as a model verification tool.

- Task 1: Visit the medical store and pick up medicines and a syringe
- Task 2: Visit the laundry store to collect clothes and get them washed

- Task 3: Visit the dental clinic to check whether the doctor is available or not until an appointment is done.
- Task 4: Visit the office to pay the electricity and phone bills
- Task 5: Visit the Groceries shop to pick the kitchen items and utensils
- Task 6: Visit the Food court and order Chinese and continental food item
- Task 7: Visit the child play school and pick toys
- Task 8: Visit the gifts store and pack gift items dinner set and grinder box
- Task 9: Visit the sports shop and pick up a football and Tennis racquet
- Task 10: Visit the electronics Merchandise and bring appliances AC and a geyser
- Task 11: Visit the shopping complex and bring ties and some clips.

Table 1. Region aliases

S. No	Area	Alias
1	Medical: Medicine	A_1
2	Medical: Syringe	A_2
3	Laundry: Clothes	B_1
4	Laundry: Washing	B_2
5	Play School	C_1
6	Play School: Pick Toys	C_2
7	Bills: Electricity	D_1
8	Bills: Phone	D_2
9	Groceries: Kitchen items	E_1
10	Groceries: Utensils	E_2
11	Food Court: Chinese	F_1
12	Food Court: Continental	F_2
13	Dental Clinic: Doctor available	G_1
14	Dental Clinic: Appointment	G_2
15	Gift store: Dinner Set	H_1
16	Gift Store: Grinder Set	H_2
17	Sports Shop: Football	I_1
18	Sports Shop: Tennis Racquet	I_2
19	Electronics Shop: AC set	J_1
20	Electronics Shop: immersion rod	J_2
21	Complex: Ties	K_1
22	Complex: Clips	K_2

5.1 Simulation Scenario 1

The robot was asked to visit the medical shop to pick medicines and a syringe, followed by visiting a laundry shop and getting the clothes washed, followed by a visit to the doctors clinic until an appointment is being fixed, and visit to the office to pay the electricity bill and phone bill, and then visit to the groceries shop to collect the kitchen items and utensils, followed by a visit to the restaurant to order and collect Chinese and Continental food,

next to the child play school and pick some toys, followed by the visit to a gifts store to pack items dinner set and grinder box, subsequently to visit the sports shop to pick a football and a Tennis racquet, followed by a visit to the electronics merchandise shop to book an Air Conditioner and a geyser, and then finally to a shopping complex to bring some clips and ties. The LTL specification for the same was as $((\Diamond A_1 \wedge \Diamond A_2 \wedge \Diamond B_1 \wedge \Diamond B_2 \wedge ((\Diamond C_1)\, U\, (\Diamond C_2)) \wedge \Diamond D_1 \wedge \Diamond D_2 \wedge \Diamond E_1 \wedge \Diamond E_2 \wedge \Diamond F_1 \wedge \Diamond F_2 \wedge \Diamond G_1 \wedge \Diamond G_2 \wedge \Diamond H_1 \wedge \Diamond H_2 \wedge \Diamond I_1 \wedge \Diamond I_2 \wedge \Diamond J_1 \wedge \Diamond J_2 \wedge \Diamond K_1 \wedge \Diamond K_2))$.

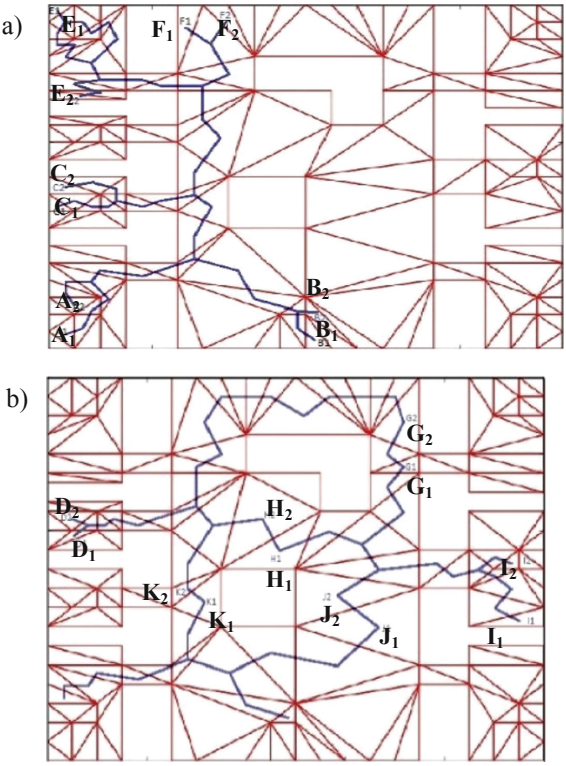

Fig. 4. Output of the algorithm (a) First group (b) Second group

The problem is solved by using the proposed algorithm. The threshold was set to 35 min. The algorithm succeeded by forming 2 groups only. The First group recorded a time of 34.19 min for the path length of 139. The path followed by the robot is shown by Fig. 4(a). Table 2 summarizes the calculations made by the algorithm in the computation of the path. Group 2 primarily incorporates all other tasks and the path is shown in Fig. 4(b) while the calculations are shown in Table 3. The total time required to complete the assigned tasks was 34.26 min. The path length for the tasks to be finished was 137 each assumed to be of a unit path length. Thus, with the proposed algorithm, the total time comes out to be 72.47 min and the path length for the same turns out to be 276 assuming unit lengths each which is less than that of the 290.

Overall with the proposed algorithm the computational time and the path length both turned out to be modestly small, while the algorithm could cater to the time thresholds of most of the tasks.

Table 2. First group with the relative path length

S. No	No. of propositions	Computation time (secs)	Path length	Robot driving time (secs)	Total time (mins)
1	2	151	43	645	10.75
2	4	313	53	795	13.25
3	6	537	83	1245	20.75
4	8	867	93	1395	23.26
5	10	1046	117	1755	29.26
6	12	2434	139	2085	34.19
7	14	3945	151	2265	37.81

Table 3. Second group with the relative path length

S. No	No. of propositions	Computational time (secs)	Path length	Robot driving time (secs)	Total time (mins)
1	2	147	65	975	16.25
2	4	425	86	1290	21.51
3	6	526	104	1560	26.01
4	8	794	122	1830	30.51
5	10	945	137	2055	34.26

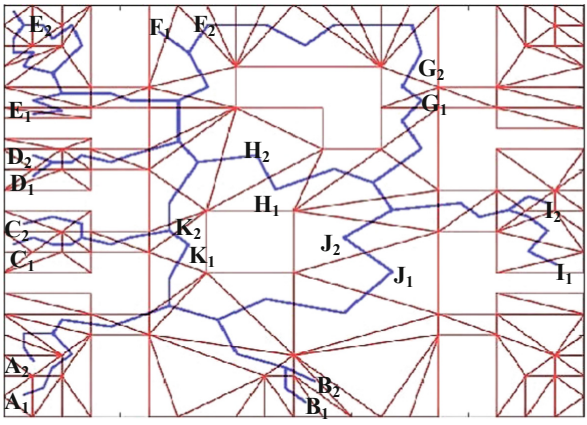

Fig. 5. The Final Path depicting the tasks

In order to compare the approach, two baselines are proposed. The first baseline is a typical implementation of model verification using LTL, in which the entire mission is given as one group to the mission solver, and is called as *all at once* for comparisons. The second baseline is based upon the heuristics that the tasks are already available in some prioritized manner. The baseline takes the missions in a greedy manner, in the increasing order of priority and solves them one by one, called as *one by one* for comparisons.

The results of the first baseline are shown in Fig. 5. The total path length taken to traverse all the satisfying regions avoiding the obstacles was observed as 290. These tasks were performed without having any time constraint. The total time taken for the above specified specification was around 92.51 min for solving 22 proposition variables with 1200 s as the computation time and 4350 s as the robot driving time.

To compare the results, we use 3 metrics path length, total time (planning time and execution time) and number of real time tasks whose time threshold was met. It can be seen that the proposed algorithm performed better in all metrics as compared to the baselines. The number of regions covered by our algorithm within the required threshold was 8 as compared to the others which were 2. The results of the three algorithms are summarized in Table 4. A more detailed view showing every task and the performance on a per task basis can be seen from Table 5.

Table 4. Overall comparisons for scenario 1

	Path length	Total time (mins)	No. of pass
Proposed algorithm	**276**	**72.47**	**8**
One by one	*290*	92.51	*2*
All at once	360	*82.42*	*2*

Table 5. Task level comparisons between algorithms for scenario 1

S. No	Region	Proposed algorithm	One by one	All at once
1	Medical	110, Pass	360, Fail	269, Fail
2	Laundry:	112, Pass	317, Fail	278, Fail
3	Play School	124, Pass	287, Fail	245, Fail
4	Bills:	68, Pass	261, Fail	233, Fail
5	Groceries	26, Pass	237, Fail	188, Fail
6	Food Court:	60, Pass	209, Fail	159, Fail
7	Dental Clinic	38, Pass	184, Fail	139, Fail
8	Gift store:	50, Pass	148, Fail	129, Fail
9	Sports Shop:	98, Fail	110, Fail	87, Pass
10	Electronics Shop	53, Fail	62, Fail	49, Pass
11	Complex:	16, Fail	14, Pass	15, Pass

5.2 Simulation Scenario 2

The robot was asked to dental clinic to check whether the doctor was present or not, followed by a visit to the groceries shop to pick some kitchen items, and then visit the sports shop to find the new cricket bat and new hockey stick has arrived or not, succeeded by a visit to the laundry shop until the visit to the electronics shop to get a washing machine, and visit to the play school to pick the toys for the children, followed by the visit to the complex to get some new pins and ties, and then visit to the gift shop to get some gift items packed. The temporal logic specification for the same is as $(((\Diamond G_1)\ U\ (\Diamond G_2)) \wedge \Diamond E_1 \wedge \Diamond E_2 \wedge \Diamond I_1 \wedge \Diamond I_2 \wedge \Diamond B_1 \wedge \Diamond B_2 \wedge \Diamond J_1 \wedge \Diamond J_2 \wedge \Diamond C_1 \wedge \Diamond C_2 \wedge \Diamond K_1 \wedge \Diamond K_2 \wedge \Diamond H_1 \wedge \Diamond H_2)$. The total path length was observed to be 187.

The path length for the tasks to be finished was 137 with our proposed algorithm. With the proposed algorithm, the total time comes out to be 52.34 min with respect to the original time of 69.21 min and the path length turns out to be 137 assuming unit lengths each which is less than that of the 187. Overall, with the proposed algorithm, the computational time and the path length both turned out to be less. The number of regions covered by our algorithm within the required threshold was 8 as compared to the others which were 7. Table 6 shows the comparison of the path lengths when the tasks were carried by proposed algorithm, all tasks taken in go and when a single task at a time for scenario 2. Table 7 gives the calculations on a per task basis.

Table 6. Overall comparisons for scenario 2

Scenario 2	Path length	Total time (mins)	No. of pass
Proposed algorithm	**137**	**52.34**	**8**
One by one	*149*	*62.42*	*7*
All at once	187	69.21	1

Table 7. Task level comparisons between algorithms for scenario 2

Region	Proposed algorithm	One by one	All at once
Dental Clinic:	48, Pass	187, Fail	61, Pass
Groceries	72, Pass	170, Fail	171, Fail
Sports Shop:	52, Pass	145, Fail	138, Pass
Laundry:	85, Pass	110, Fail	85, Pass
Electronics Shop:	91, Pass	95, Fail	91, Pass
Play School	72, Pass	81, Fail	72, Pass
Complex:	45, Pass	48, Fail	45, Pass
Gift store:	16, Pass	16, Pass	16, Pass

5.3 Simulation Scenario 3

The robot was asked to dental clinic to check whether the doctor was present or not, followed by a visit to the groceries shop to pick some kitchen items, and then visit the sports shop to find the new cricket bat and new hockey stick has arrived or not, succeeded by a visit to the laundry shop until the visit to the electronics shop to get a washing machine, and visit to the play school to pick the toys for the children followed by the visit to the complex to get some new pins and ties and then visit to the gift shop to get some gift items packed. The temporal logic specification for the same is as $(((\Diamond C_1 \wedge \Diamond C_2)\ U\ (\Diamond F_1 \wedge \Diamond F_2)) \wedge \Diamond D_1 \wedge \Diamond D_2 \wedge \Diamond J_1 \wedge \Diamond J_2 \wedge \Diamond I_1 \wedge \Diamond I_2 \wedge \Diamond K_1 \wedge \Diamond K_2)$. The total path length was observed to be 125.

The total time required to complete the assigned tasks was 46.25 min. The path length for the tasks to be finished was 137. With the proposed algorithm, the total time comes out to be 46.25 min and the path length for the same turns out to be 125, which is less than that of the 156. Table 8 shows the comparison of the chosen metrics with the baselines for Scenario 3. Overall with the proposed algorithm the computational time and the path length both turned out to be less. The number of regions covered by our algorithm within the required threshold was 6 as compared to the others which were 4 and 3 respectively. A detailed per-task view can be seen in Table 9.

Table 8. Overall comparisons for scenario 3

Scenario 3	Path length	Total time (mins)	No. of pass
Proposed algorithm	**125**	**46.25**	**6**
One by one	*149*	*56.42*	*4*
All at once	156	62.45	3

Table 9. Task level comparisons between algorithms for scenario 3

Region	Proposed algorithm	Single task	All at once
Play School	48, Pass	156, Fail	48, Pass
Food Court:	41, Pass	119, Fail	118, Fail
Bills:	54, Pass	103, Fail	104,Fail
Electronics Shop:	87, Pass	82, Pass	88, Fail
Sports Shop:	35, Pass	50, Pass	53, Fail
Complex:	15, Pass	15, Pass	15, Pass

6 Conclusions

In this paper, we have proposed a solution to the robotics mission planning problem expressed with temporal logic with complex specifications wherein the mission is composed of a prioritized set of tasks, each task has a time threshold within which it must preferably be solved. We have demonstrated that the approach works much faster as compared to the two baselines, solving the complete mission in a go and solving the

tasks one by one. This is one of the only papers that defines optimality as a mixture of computation time and execution time of the robot which is a more realistic modelling.

The future work is to actually use the algorithm on a real robot and to ask it to perform the operations. Furthermore, the algorithm may also be extended to the case of multiple robots and task division among the robots is another interesting problem to be looked into the future. The generic temporal logic does not give much scope to design heuristic measures while the problem can best be solved if there are powerful heuristics to attack. This makes the problem challenging.

Another typicality that can be considered is that the robot will interact with the humans and the time for the same cannot be ascertained. This becomes more important considering that the humans may not necessarily be at the workplace but somewhere around as well as their presence or absence is stochastic in nature. So, the robot must be able to locate the human in the vicinity, approach the human and get the needed interaction done along with the human. These are all complex problems.

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

References

1. Choset, H., et al.: Principles of Robot Motion: Theory, Algorithms, and Implementations. MIT Press, Cambridge (2005)
2. Holzmann, G.: The model checker SPIN. IEEE Trans. Softw. Eng. **25**(5), 279–295 (1997)
3. Cimatti, A., et al.: NuSMV 2: an OpenSource tool for symbolic model checking. In: Brinksma, E., Larsen, K.G. (eds.) CAV 2002. LNCS, vol. 2404, pp. 359–364. Springer, Heidelberg (2002). https://doi.org/10.1007/3-540-45657-0_29
4. Vardi, M.Y., Wolper, P.: An automata-theoretic approach to automatic program verification. In: Logic in Computer Science, pp. 322–331 (1986)
5. Finucane, C., Jing, G., Kress-Gazit, H.: LTLMoP: experimenting with language, temporal Logic and robot control. In: 2010 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 18–22 October 2010, pp. 1988–1993 (2010)
6. Fainekos, G.E., Kress-Gazit, H., Pappas, G.J.: Temporal logic motion planning for mobile robots. In: Proceedings of the 2005 IEEE International Conference on Robotics and Automation, Barcelona, Spain, pp. 2020–2025 (2005). https://doi.org/10.1109/robot.2005.1570410
7. Antoniotti, M., Mishra, B.: Discrete event models+temporal logic=supervisory controller: automatic synthesis of locomotion controllers. In: Proceedings of 1995 IEEE International Conference on Robotics and Automation, Nagoya, Japan, vol. 2, pp. 1441–1446 (1995). https://doi.org/10.1109/robot.1995.525480
8. Kress-Gazit, H., Fainekos, G.E., Pappas, G.J.: Temporal-logic-based reactive mission and motion planning. IEEE Trans. Robot. **25**(6), 1370–1381 (2009). https://doi.org/10.1109/TRO.2009.2030225
9. Shewchuk, J.R.: Triangle: engineering a 2D quality mesh generator and Delaunay triangulator. In: Lin, M.C., Manocha, D. (eds.) WACG 1996. LNCS, vol. 1148, pp. 203–222. Springer, Heidelberg (1996). https://doi.org/10.1007/BFb0014497

10. Karaman, S., Frazzoli, E.: Vehicle routing problem with metric temporal logic specifications. In: IEEE DCC, December 2008
11. Bhatia, A., Kavraki, L.E., Vardi, M.Y.: Sampling-based motion planning with temporal goals. In: 2010 IEEE International Conference on Robotics and Automation (ICRA), 3–7 May 2010, pp. 2689–2696 (2010). https://doi.org/10.1109/robot.2010.5509503
12. Kala, R.: Sampling based mission planning for multiple robots. In: Proceedings of the 2016 IEEE Congress on Evolutionary Computation, Vancouver, BC, Canada, pp. 662–669 (2016)
13. Kala, R.: Dynamic programming accelerated evolutionary planning for constrained robotic missions. In: Proceedings of the IEEE Conference on Simulation, Modelling and Programming for Autonomous Robots, Brisbane, Australia, pp. 81–86 (2018)
14. Ruppert, J.: A delaunay refinement algorithm for quality 2-dimensional mesh generation. J. Algorithms **18**(3), 548–585 (1995)