# An Optimal Approach to Anytime Task and Path Planning for Autonomous Mobile Robots in Dynamic Environments

Cuebong Wong[1][0000−0002−6541−6125], Erfu Yang[1][0000−0003−1813−5950], Xiu-Tian Yan[1][0000−0002−3798−7414], and Dongbing Gu[2][0000−0002−0986−2921]

[1] Department of Design, Manufacture and Engineering Management
University of Strathclyde, Glasgow, United Kingdom
{cuebong.wong,erfu.yang,x.yan}@strath.ac.uk
[2] School of Computer Science and Electronic Engineering
University of Essex, Colchester, United Kingdom
dgu@essex.ac.uk

**Abstract.** The study of combined task and path planning has mainly focused on feasibility planning for high-dimensional, complex manipulation problems. Yet the integration of symbolic reasoning capabilities with geometric knowledge can address optimal planning in lower dimensional problems. This paper presents a dynamic, anytime task and path planning approach that enables mobile robots to autonomously adapt to changes in the environment. The planner consists of a path planning layer that adopts a multi-tree extension of the optimal Transition-based Rapidly-Exploring Random Tree algorithm to simultaneously find optimal paths for all movement actions. The corresponding path costs, derived from a cost space function, are incorporated into the symbolic representation of the problem to guide the task planning layer. Anytime planning provides continuous path quality improvements, which subsequently updates the high-level plan. Geometric knowledge of the environment is preserved to efficiently re-plan both at the task and path planning level. The planner is evaluated against existing methods for static planning problems, showing that it is able to find higher quality plans without compromising planning time. Simulated deployment of the planner in a partially-known environment demonstrates the effectiveness of the dynamic, anytime components.

**Keywords:** Robotics · Autonomous systems · Task planning · Path planning · Combined task and motion planning · Dynamic planning.

## 1 Introduction

Many practical applications for robots to date still rely on human-in-the-loop control [1], which is costly and inefficient for remote operation or long-duration missions. Embedding intelligence into robotic systems can provide robots with the required autonomy to plan their actions to achieve the desired goals. However, this remains an ongoing challenge due to the existence of uncertainty in the real-world. Robots must be capable of dynamically adapting to changes and recovering from failures to operate reliably and safely without human intervention. The study of dynamic task planning and path planning is therefore an important aspect in the development of autonomous robotics.

Task planning is the process of finding a sequence of high-level actions to accomplish defined objectives. Generally, the task planning domain is represented using symbolic language and solved by searching a finite set of discrete states. Geometric relationships of objects are abstracted to reduce the size of the state space. In contrast, path planning (a purely geometric motion planning problem [2]) solves for a low-level motion path in $\mathbb{R}^d$ space to move a robot from a start configuration to a goal configuration. Extensive work exist in literature for task planning and path planning, but they have mostly been conducted in isolation. While various problems such as fruit harvesting [3] and component disassembly [4] can be solved using a decoupled approach to planning, solving more complex or larger scale problems often demands a more seamlessly coupled approach. Applying a decoupled approach in these cases may produce sub-optimal plans or, in the worst case, lead to an intractable problem. Various authors have begun to address complex manipulation problems by integrating task and motion planning. Notable examples include FFRob [5], an integrated planner that extends the *FastForward* heuristics used in symbolic planning to consider geometric details in the task planner, and the TM-Kit [6], a probabilistically complete, general-purpose framework for combined task and motion planning. However, these work focused on feasibility planning due to the high dimensionality of manipulation planning problems. When applied to robotic navigation, a coupled planning approach can improve the optimality of long mission plans, or adapt task plans in response to observable failures or perceived changes in the world.

Several authors have explored this avenue in the context of mobile robots. For example, the UP2TA framework [7] provides optimal plans for exploration mission planning. However, the authors did not consider general cost spaces or aspects of re-planning. The authors in [8] addressed multi-robot planning for partially-known environments and considered minimisation of robot resources. Their approach enables adaptation of the plan as new obstacles are detected, but possesses a number of limiting characteristics, such as being unable to consider task dependencies, requiring several hundred seconds to plan for single robot problems, and performing a needless number of re-planning instances as each update to a planned path triggers an instance of task planning. Motivated by these ongoing challenges, the contributions of this work is two-fold: (i) we present a base planner that integrates task and path planning to enable optimal task planning in continuous cost spaces using a multi-tree T-RRT* (Transition-based Rapidly-exploring Random Tree) algorithm [9] and compare it to existing methods, and (ii) we extend the base planner with dynamic, anytime capabilities to enable efficient high and low level re-planning in partially known or dynamic environments. We collectively refer to the proposed planner as the Dynamic, Anytime Task and Path Planner (DA-TPP)[3].

[3] This paper is an invited extension to the work presented in [10].

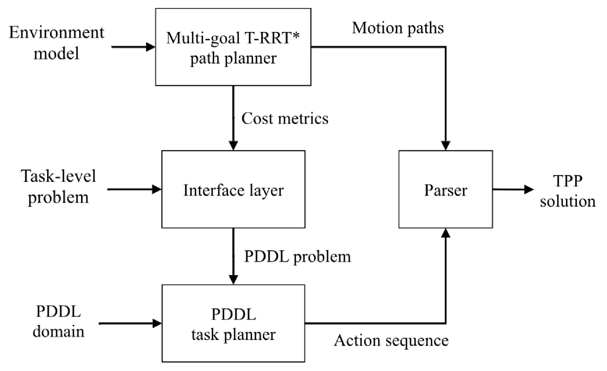

**Fig. 1.** Base planner architecture

## 2 Problem Definition

This paper addresses Task and Path Planning (TPP) problems consisting of a robot in $\mathbb{R}^2$ space and a set of landmarks $L$. $L$ represents locations in space where a robot must perform a set of domain-specific actions. The robot navigates the environment by performing movement actions corresponding to motions between any pair of landmarks $l_a, l_b \in L$. A valid planning problem contains an initial landmark $l_{init}$, where the robot starts from, and a set of goals describing the tasks that must be performed at each landmark. A goal landmark $l_{goal}$ for which the robot must be located at the end of the plan may optionally be specified. In our experiments we assume that for any initial planning problem the robot must begin and end at a root landmark $l_{init} = l_{goal} = l_0$ (the *robot base*) and there exist tasks that must be completed at every landmark (note, however, that our approach is equally applicable when these are not true). As re-planning takes place during execution, $l_{init}$ is updated to reflect the new robot location. A valid TPP solution then consists of a sequence of movement actions and corresponding motion paths that guide the robot from $l_{init}$ to $l_{goal}$ through a route that enables the completion of all tasks while accumulating the lowest cost.

## 3 The DA-TPP Approach

The base planner of the DA-TPP (Fig. 1) employs a path planning layer to find an optimal path for all valid movement actions. The corresponding path costs are then linked to the discrete movement action costs for optimal task planning through an interface layer such that true path plans are used to guide the task planner. Given a continuous cost space mapping function, $c$, from which a cost value can be derived for all robot configurations, we define the path cost function, $c_p$, of a path $\sigma$ as a weighted sum of integral cost and path length:

$$c_p(\sigma) = f(\sigma)\left(\frac{w_a}{n}\sum_{k=1}^{n} c\left(\sigma\left(\frac{k}{n}\right)\right) + w_b\right) \quad (1)$$

Where $n$ is the number of subdivisions of $\sigma$, $f()$ is the path length, $w_a$ and $w_b$ are weight factors for $c$ and $f(\sigma)$, respectively, and $k$ represents the point along $\sigma$. This formulation enables consolidation of both the cost function and path length as a weighted sum multi-objective optimisation problem.

When a termination criteria is met, the planner returns the best set of paths found for each of these movement actions ($c_p = \infty$ if no solution is found). A satisfactory TPP solution exists when all landmarks associated with the task objectives are connected in a reachability graph containing $l_{init}$, where each vertex represents a landmark and each edge is a valid motion path. That is, the robot can reach and return from any landmark by traversing through the vertices of the graph. The task planning layer employs PDDL representation [11], and is solved using Local Planning Graphs (LPG-*td*) [12]. We chose to represent the planning problem in PDDL due to its wide acceptance as a standard for representing classical symbolic planning problems. This enables interchangeable use of other heuristic planners (such as Metric-FF [13]) and provides generality to the planner to include new actions and treat task dependencies etc.

### 3.1 Anytime Extension

Anytime planning supports a request for an initial solution after a fixed allotted time, which minimises idle time at the start of a task. During execution of an existing plan, the planner continues to iterate the path planning algorithm to improve the solution at subsequent requests. Suppose that the path cost for an action $a$ is $c_p$. Like the work in [14], an upper cost bound $C_s^+$ is defined as

$$C_s^+ = (1 - \eta_a) \cdot c_p(\sigma). \tag{2}$$

Where $\eta_a$ is a constant. When the path planning layer finds a new path $\sigma'$ for $a$ with $c_p(\sigma') < C_s^+$, a new instance of task planning is called. This mechanism guarantees that the task planning layer is called only when a guaranteed improvement to an action cost is found. During execution, it is also necessary to consider the goals that have been met thus far. This is addressed by updating the initial state of the planning problem at each planning instance to reflect the next state of the world after executing the current action of the latest plan.

### 3.2 Dynamic Anytime Extension

The complete DA-TPP architecture is shown in Fig. 2. The key extensions are a local path correction modeul and a global re-planner. Each time an obstruction to a currently executed path is detected, the local path correction algorithm finds a new optimal path to the goal configuration. The DA-TPP then determines whether an instance of global re-planning should be called using a heuristically-defined lower cost bound $C_s^-$ shown in (3). Letting $c_p'$ be the path cost of the remaining segments of the original path, $C_s^-$ is given by

$$C_s^- = (1 + \eta_d) \cdot c_p'(\sigma). \tag{3}$$

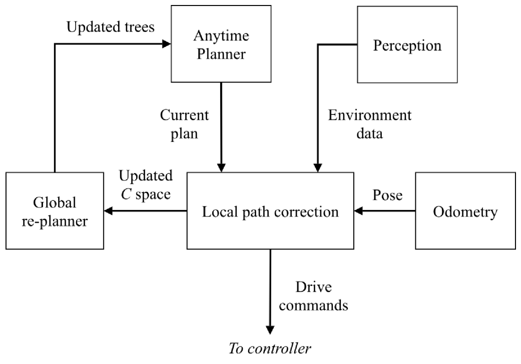

**Fig. 2.** Dynamic anytime planner architecture

Where $\eta_d$ is a constant. When $c_p(\sigma') > C_s^-$, global re-planning takes place. This permanently updates all planning trees with the detected obstruction and a new optimal sequence of movement actions is generated.

By applying the heuristic cost bound defined in (3), the DA-TPP provides the following behaviours in dynamic environments. When minor obstructions are encountered, only local adjustments are made to the currently executed path. This, in general, does not affect the optimality of the task plan from a high-level perspective. It is sufficient then to correct paths locally each time the same obstruction is encountered without initiating a global re-planning procedure, thus limiting the number of task-level re-planning instances. However, in situations where an obstruction causes significant diversion for a particular traverse (e.g. from road blockages), the likelihood of the obstruction affecting the optimality of the task plan is high. This is due to the increased cost of the current path and possible extended effects on other planned paths. In these situations the planner updates the entire plan (including all motion paths) to maintain optimality.

## 4 Path Planning

The path planning layer of the base planner is implemented following the multi-T-RRT* approach described in [9], which simultaneously searches for all optimal paths between landmarks by iteratively growing trees rooted at each landmark to explore the configuration space. Readers are directed to this initial work for a detailed description of the algorithm.

### 4.1 Local Path Correction

The local path re-planner corrects any single path according to procedures based on elements of the RRT$^X$ algorithm [15]. At the start of any movement action, a new tree $T_{new}$ is generated from two trees $T_0$ and $T_g$ corresponding to the start and goal landmarks $l_0$ and $l_g$, respectively (Fig. 3). $T_{new}$ is rooted at the goal

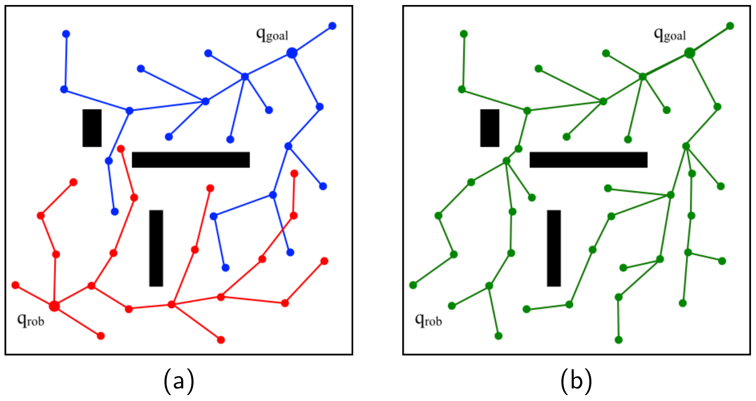

**Fig. 3.** (a) Trees $T_0$ (red), rooted at $q_{rob}$, and $T_g$ (blue), rooted at $q_{goal}$, grown across the configuration space. (b) The resulting tree $T_{new}$ (green) resulting from the merge function applied on $T_0$ and $T_g$.

**Algorithm 1** localPathCorrection

```
Input: Merged tree $T_{new}$, current path $\sigma$ and robot configuration $q_{rob}$
Output: Updated path $\sigma'$
1: $O_{new} \leftarrow getObstacles()$
2: if $collision(\sigma, O_{new})$ then
3:     $invalidNodes(T_{new}, O_{new})$
4:     $updateOrphans(T_{new})$
5:     $rewireTree(T_{new})$
6:     $\sigma' \leftarrow updatePath(T_{new}, q_{rob})$
7: end if
```

configuration $q_g$ and consists of all the vertices of $T_0$ and $T_g$ rewired to minimise path cost according to the new tree root. This speeds up dynamic re-planning later tree root does not need to be updated as the robot advances along the path. As new obstacles are detected during execution, the remaining segments of a traversed path are checked for collision. If these obstacles invalidate any part of this path, Algorithm 1 is called to update the path.

The algorithm invalidates vertices that lie in the collision region of new obstacles. All valid descendant vertices are then updated as orphans. This closely resembles the *propagateDescendants* function in [15]. The algorithm then updates the branches of the tree by iterating through a queue of vertices consisting initially of the neighbours of orphans (see *reduceInconsistency* function in [15]). For each of these vertices, the algorithm updates its parent, and then runs a rewiring procedure on its neighbouring vertices. Any vertices that are rewired at this step are then added to the queue. This continues until no further improvements can be made. Finally, a new path from the tree root to the robot configuration $q_{rob}$ is found by attempting to connect $q_{rob}$ to neighbouring vertices.

**Algorithm 2** globalReplanning

```
Input: Set of all planning trees $T$, new obstacles $O_{new}$ and robot configuration $q_{rob}$
Output: Set of updated planning trees $T$ and set of path solutions $\Sigma_{best}$
1: for all $T_k \in T$ do
2:     $invalidNodes(T_k, O_{new})$
3:     $updateOrphans(T_k)$
4:     $rewireTree(T_k)$
5: end for
6: $\Sigma_{best} \leftarrow updatePaths(T)$
7: $\Sigma_{best} \leftarrow pathsFromRobotToLandmarks(\Sigma_{best}, T, q_{rob})$
```

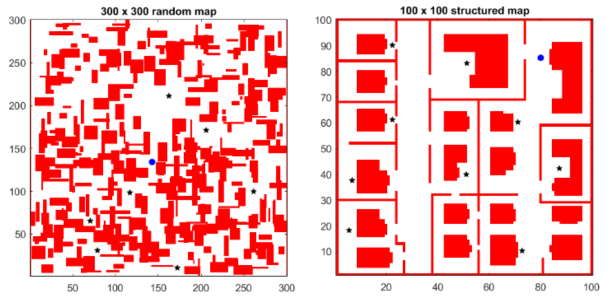

**Fig. 4.** Example planning problems used to compare planners. Blue markers represent the robot base and black stars represent all other landmarks.

### 4.2 Global Re-Planning

When the condition in (3) is met, a new action-motion sequence is determined by updating the costs of all movement actions. This is achieved by updating the solutions of the path planning layer using Algorithm 2. The algorithm first updates every tree by invalidating infeasible vertices, updating orphaned vertices and propagating a rewiring cascade, as in Section 4.1. New optimal paths between landmarks are obtained by finding new connecting vertices between corresponding pairs of trees. The set of best paths $\Sigma_{best}$ are updated accordingly. A temporary landmark $l_{temp}$ is then inserted into the TPP problem at $q_{rob}$. An attempt to find an optimal path from each original landmark to $l_{temp}$ is made by testing connections from neighbouring vertices of each tree to the root of $l_{temp}$. $\Sigma_{best}$ is then expanded to include these paths.

## 5 Experimental Evaluation

### 5.1 Base Planner Comparison

The base planner is benchmarked across a number of randomly generated cluttered and structured environments varying between 50 $\times$ 50, 100 $\times$ 100 and 300 $\times$ 300 in dimensions (examples shown in Fig. 4). The cost function $c$ in (1) for

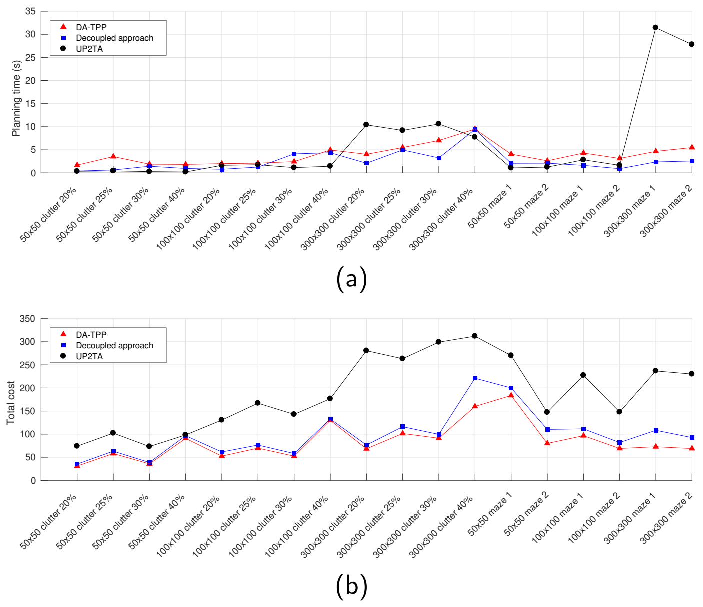

**Fig. 5.** Performance comparison between basic task planning, DA-TPP and UP2TA methods over 100 trials. (a) mean planning time for initial solution, (b) mean plan cost for initial solution. Simulations were conducted on an Intel® Xeon® CPU E3-1270 v3.

any configuration $q$ is given by $c = 1/\delta^2$ , where $\delta$ is the distance to the nearest obstacle. Thus $c$ describes the 'closeness' of $q$ to the nearest obstacle. $w_a$ and $w_b$ were set to 0.97 and 0.03, respectively.

We compared the performance of the DA-TPP base planner with a decoupled planning approach and UP2TA [7]. The decoupled approach solves for an optimal action sequence by using the Euclidean distances between landmarks as movement action costs. A single instance of path planning was subsequently called for each movement action to obtain the TPP solution. For consistency across the comparison, we used a bi-directional equivalent of the multi-T-RRT* algorithm for path planning. Our implementation of the UP2TA framework consisted of a greedy search algorithm to approximate the cost metrics for each possible movement action. The LPG-*td* planner was then used to solve the task planning problem (the original authors used the *FastForward* planner, but for consistency we applied the same PDDL planner as in our work). Finally, a second path planning layer that employs the graph search-based *Theta** algorithm [16] was used to obtain the true paths for each movement action.

The results generated from a PC with an Intel® Xeon® CPU E3-1270 v3 (3.50 GHz) are provided in Fig. 5. We observe that the UP2TA fails to consider cost spaces and consequently performs notably worse than other planners in terms of path cost. Planning times also highlight a key deficit of grid-based approaches: as the size of the problem increases, their performance decreases

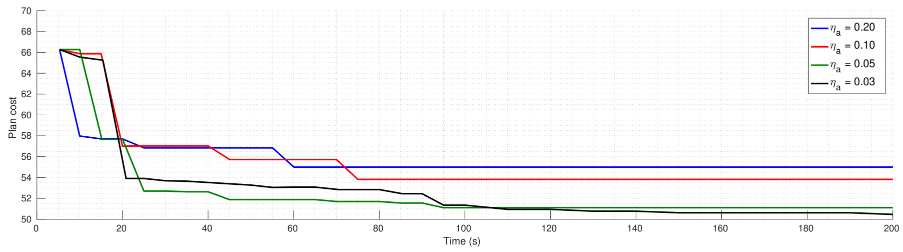

**Fig. 6.** Anytime planning - plan cost steadily decreases as more computation time is allowed. Step-like cost reductions indicate task-level improvements to the plan.

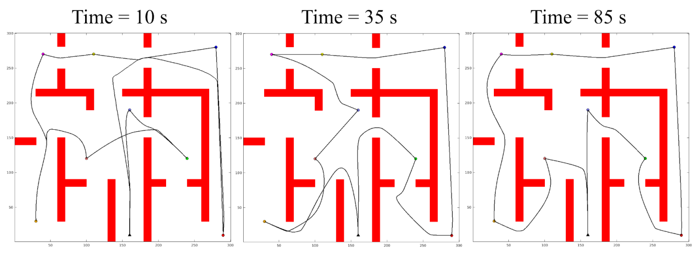

**Fig. 7.** Example of anytime plan evolution at selected time instances.

rapidly, as observed for environments of size 300 $\times$ 300. The decoupled approach scales better with the size of the problem and maintains a low computational cost across all trials. However, this approach finds solutions with overall costs that are generally greater than the DA-TPP approach, particularly for structured, maze-like environments. This is an expected observation as the task planning layer is ill-informed by misleading action costs. Without knowing the geometric relationships of objects in the world, costly movement actions are unknown to the symbolic planner. Thus DA-TPP consistently finds the lowest cost plans in all test cases. Although this sacrifices computational efficiency slightly, the proposed planner scales well with the size of the problem and indeed finds a solution faster than UP2TA in almost all cases for 300 $\times$ 300 environments. Finally, the quality of DA-TPP solutions may be further improved over time as a result of anytime planning, as discussed below.

## 5.2 Anytime Evaluation

The behavior of the anytime component of DA-TPP was assessed in the following way. An initial solution was first obtained using the base planner. Starting from this same initial solution each time, the planner described in Section 3.1 was run four times with $\eta_a$ set to 0.2, 0.1, 0.05 and 0.03, respectively. For each run, a request for the current solution was made at the defined time instances shown in Fig. 6. Sample plans obtained over the trial durations are provided in Fig. 7.

In general, the quality of a plan improves at the path level and task level as further computation time is allowed. These correspond to small progressive cost decreases and larger *step* changes observed in Fig. 6, respectively. Task level improvements take place only when sufficient improvements to a local path changes the optimality of the global task plan. These occurred only 3-4 times over the duration of 200 seconds for all runs. Hence it is often unnecessary to re-plan an entire action-motion sequence for small local path changes. This provides motivation for the use of the update criteron defined in (3) for global re-planning. Finally, we observe that the value of $\eta_a$ controls the rate of convergence and optimality of the planner. For larger values of $\eta_a$, the planner spends less time re-planning the task sequence as it is triggered only when more significant improvements to local paths are found, thus converging faster (60 seconds for $\eta_a = 0.20$ vs 95 seconds for $\eta_a = 0.05$). However, this in turn dismisses small, steady cost reductions that lead to a better quality final solution. On the other hand, with $\eta_a = 0.03$, the solution does not reach convergence after 200 seconds, yet the planner is able to find the lowest cost solution across the four runs. In the subsequent experiment we set $\eta_a = 0.05$ based on its balanced convergence and optimality characteristics.

### 5.3 Dynamic Anytime Evaluation

Finally, the complete DA-TPP approach was assessed through simulations in a $30 \times 30$ meters partially-known environment shown in Fig. 8. The robot begins executing a plan after an initial solution is obtained. We simulate real-time execution on the Gazebo simulator using a Clearpath Husky. Perception of the environment is achieved using a laser scanner with a range of 30 meters, while $\eta_d$ is set to 0.05 (chosen based on an analysis similar to the selection for $\eta_a$).

Fig. 8 shows global re-planning instances where actions belonging to a previously optimal plan (e.g. traversing from landmark 5 to landmark 6) are avoided on detection of significant blockage, while only local path corrections take place for smaller obstructions. Furthermore, we draw particular attention to the observation that the robot visits landmark 7 twice in the executed set of paths, which showcases the behaviour of the planner when a direct path between two landmarks do not exist. After running the global re-planning procedures triggered by the detection of obstacles C and D, a feasible path between landmarks 5 and 6 was no longer found. Nevertheless, a multi-segment path consisting of two movement actions (paths 7 and 8) enabled the robot to reach landmark 6. In this way the planner is able to identify infeasible actions through the inference from the path planning layer.

The solutions of DA-TPP may be subject to local minima according to the limitations of the heuristic planner used in task planning. For example, LPG-*td* may provide locally-optimal task plans but is always able to return solutions quickly. Other planners such as Metric-FF can provide globally-optimal solutions at the expense of lower efficiency. Conversely, the path planning layer maintains the asymptotic optimality property of RRT* and thus always converges towards globally-optimal solutions if sufficient time is allowed.

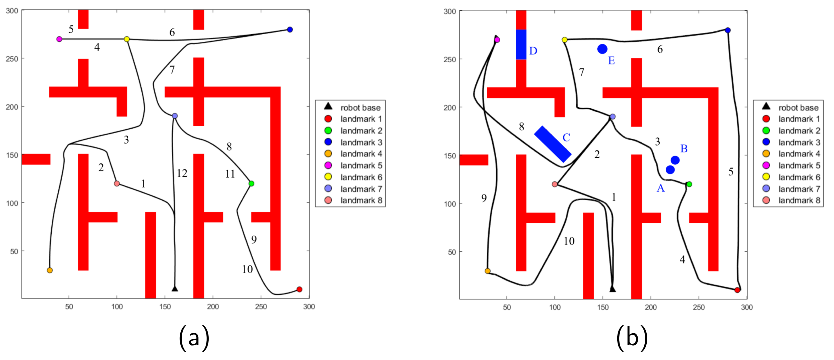

**Fig. 8.** (a) Initial plan for a mobile robot located at *robot base*. Motion sequence is indicated by numerical sequence. (b) True executed paths at runtime (detected obstacles shown in blue).

## 6 Conclusion

The DA-TPP is an anytime task and path planner for autonomous mobile robots with re-planning capabilities. The base planner uses the Multi-T-RRT* algorithm to find optimal paths for all movement actions in relation to the continuous cost space. The corresponding path costs are linked to discrete movement actions in the task planning layer, which is solved using off-the-shelf planners. In our results, we show that the planner is competitive in terms of scalability and the quality of solutions obtained. The anytime extension enables a sub-optimal solution to be found quickly, and any further computation time given to planning continues to improve the quality of the task plan by a bounded degree of improvement. A local path correction algorithm updates individual paths during execution, while the global re-planner updates the structures of all trees in the path planning layer to maintain global optimality when large changes are observed. In this way the DA-TPP possesses the flexibility to adapt both individual paths and entire plans accordingly depending on the significance of observed changes in the environment. One of the limiting factors on the computational performance of the DA-TPP in large planning problems is the population size of nodes that make up each planning tree in the path planning layer. One improvement for future work includes the removals of useless nodes and branches that provide zero contribution to the search for more optimal paths over existing ones. This is similar to the *branch-and-bound* technique applied in [17], but an admissible heuristic to assess each node against all goals is necessary.

## Acknowledgements

Funded by the Engineering and Physical Sciences Research Council (EPSRC) under its Doctoral Training Partnership Programme (DTP 2016-2020 University of Strathclyde, Glasgow, UK).

## References

1. C. Wong, E. Yang, X.-T. Yan, and D. Gu, Autonomous robots for harsh environments: a holistic overview of current solutions and ongoing challenges, *Syst. Sci. Control Eng.*, vol. 6, no. 1, pp. 213-219, Jan. 2018.
2. A. Gasparetto, P. Boscariol, A. Lanzutti, and R. Vidoni, Path planning and trajectory planning algorithms: A general overview, *Motion and Operation Planning of Robotic Systems*, Springer, Cham, 2015, pp. 3-27.
3. T. T. Nguyen, E. Kayacan, J. De Baedemaeker, and W. Saeys, Task and Motion Planning for Apple Harvesting Robot, *IFAC Proc. Vol.*, vol. 46, no. 18, pp. 247-252, Aug. 2013.
4. C. Friedrich, A. Csiszar, A. Lechler, and A. Verl, Efficient Task and Path Planning for Maintenance Automation Using a Robot System, *IEEE Trans. Autom. Sci. Eng.*, vol. 15, no. 3, pp. 1205-1215, Jul. 2018.
5. C. R. Garrett, T. Lozano-Pérez, and L. P. Kaelbling, FFRob: An efficient heuristic for task and motion planning, *Algorithmic Foundations of Robotics XI*, Springer, Cham, 2015, pp. 179-195.
6. N. T. Dantam, S. Chaudhuri, and L. E. Kavraki, The Task- Motion Kit: An open source, general-purpose task and motion- planning framework, *IEEE Robot. Autom. Mag.*, vol. 25, no. 3, pp. 61-70, Sep. 2018.
7. P. Muñoz, M. D. R-Moreno, and D. F. Barrero, Unified framework for path-planning and task-planning for autonomous robots, *Rob. Auton. Syst.*, vol. 82, pp. 1-14, Aug. 2016.
8. B. Woosley and P. Dasgupta, Integrated real-time task and motion planning for multiple robots under path and communication uncertainties, *Robotica*, vol. 36, no. 3, pp. 353-373, Mar. 2018.
9. C. Wong, E. Yang, X.-T. Yan, and D. Gu, Optimal path planning based on a multi-tree T-RRT* approach for robotic task planning in continuous cost spaces, *12th France-Japan and 10th Europe-Asia Congress on Mechatronics*, 2018, pp. 242-247.
10. C. Wong, E. Yang, X.-T. Yan, and D. Gu, Dynamic anytime task and path planning for mobile robots, *UK-RAS19 Conference on Embedded Intelligence: Enabling & Supporting RAS Technologies*, 2019, pp. 36-39.
11. D. McDermott, The PDDL planning domain definition language, *AIPS-98 Plan. Compet. Comm.*, 1998.
12. A. Gerevini, A. Gerevini, A. Saetti, I. Serina, and P. Toninelli, LPG-TD: A fully automated planner for PDDL2.2 domains, *14th Int. Conf. Autom. Plan. Sched. Int. Plan. Compet.*, 2004.
13. J. Org Hoomann, The Metric-FF planning system: Translating 'ignoring delete lists' to numeric state variables, *J. Artiicial Intell. Res.*, vol. 20, pp. 291-341, 2003.
14. D. Ferguson and A. Stentz, Anytime RRTs, *Proceedings of the 2006 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2006, pp. 5369-5375.
15. M. Otte and E. Frazzoli, RRT$^X$ : Asymptotically optimal single- query sampling-based motion planning with quick replanning, *Int. J. Rob. Res.*, vol. 35, no. 7, pp. 797-822, Jun. 2016.
16. S. K. and A. F. K. Daniel, A. Nash, Theta*: Any-angle path planning on grids, *J. Artif. Intell. Res.*, vol. 39, 2010.
17. S. Karaman, M. R. Walter, A. Perez, E. Frazzoli, and S. Teller, Anytime Motion Planning using the RRT*, *2011 IEEE International Conference on Robotics and Automation*, 2011, pp. 1478-1483.