DOI 10.31256/UKRAS19.10

Dynamic, Anytime Task and Path Planning for Mobile Robots

Cuebong Wong, Erfu Yang, Xiu-Tian Yan, and Dongbing Gu

***Abstract*— The study of combined task and motion planning has mostly been concerned with feasibility planning for high-dimensional, complex manipulation problems. Instead this paper gives its attention to optimal planning for low-dimensional planning problems and introduces the dynamic, anytime task and path planner for mobile robots. The proposed approach adopts a multi-tree extension of the T-RRT* algorithm in the path planning layer and further introduces dynamic and anytime planning components to enable low-level path correction and high-level re-planning capabilities when operating in dynamic or partially-known environments. Evaluation of the planner against existing methods show cost reductions of solution plans while remaining computationally efficient, and simulated deployment of the planner validates the effectiveness of the dynamic, anytime behavior of the proposed approach.**

Keywords—robotics, autonomous systems, task planning, path planning, combined task and motion planning, dynamic planning

I. Introduction

The study of task planning and path planning for applications in robotics has largely been conducted in isolation. Task planning is carried out using a symbolic representation of the world consisting of a finite set of discrete states. In this domain, geometric relationships of objects in the world are, in general, highly abstracted to reduce the size of the state space. Thus task planners are rarely able to consider geometric constraints of the planning problem. Path planning (a purely geometric *motion planning* problem [1]) on the other hand seeks to find an admissible path in $\mathbb{R}^d$ space to transition a robot from a start configuration to a goal configuration by exploiting the geometric representation of the environment. While trivial problems may be efficiently solved by performing task planning in isolation and subsequently calling a path planning instance for each movement action, applying the same approach to more complex problems may produce sub-optimal plans, or in the worst case be unsolvable.

An emerging concept in literature called combined task and motion planning (CTMP) seeks to address this by integrating the process of path planning and task planning. The authors in [2] proposed a hierarchical approach that interleaves planning with execution. The FFRob [3] is a CTMP planner that extends FastForward heuristics used in symbolic planning to robot motion planning, while [4] performs CTMP by precomputing motion graphs and collision tables for a mobile manipulator. The Task-Motion Kit (TMKit) [5], which addresses probabilistic completeness and generality, is a general-purpose framework that interfaces the symbolically-defined task domain with the geometric relational properties of the motion domain through a domain semantics layer. These aforementioned work focus on feasibility planning for high complexity problems involving object manipulation. However, far fewer works have applied CTMP concepts to planning for lower-complexity problems consisting of mobile robots. Yet CTMP can provide improvements to the optimality of long mission plans, or adapt task plans in response to failures or perceived dynamic changes to the world. We highlight the UP2TA framework [6], which integrates task and path planning for applications to exploration mission planning. However, UP2TA does not consider general cost spaces in path optimisation, nor facilitates dynamic re-planning. Thus the contributions of this work is two-fold: (i) we compare a base planner, which integrates task and path planning to enable optimal task planning in continuous cost spaces by making use of a multi-tree T-RRT* algorithm [7], against UP2TA and a *planning-in-isolation* approach, and (ii) we extend the base planner with dynamic, anytime capabilities to enable high-level re-planning and low-level path corrections in dynamic environments. We collectively refer to the proposed planner as the Dynamic, Anytime Task and Path Planner (DA-TPP).

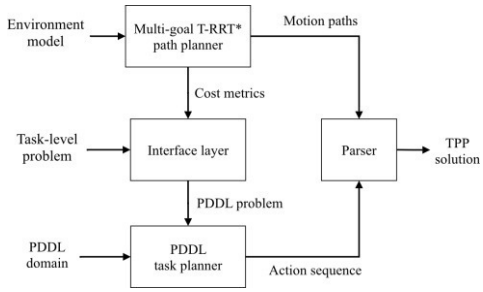


Figure1. The proposed base planner architecture

II. Problem Definition

This paper addresses task and path planning (TPP) problems for autonomous mobile robots. As a minimum, this consists of a robot in an environment containing a set of landmarks L. These landmarks represent locations in space where a robot must perform some actions. A movement action consists of traversing between any pair of landmarks $l_a, l_b \in L$. A valid planning problem within this domain consists of an initial landmark from which the robot starts from, and a set of tasks that must be performed at each landmark. These may be defined as a discrete action within the planning domain. A goal landmark for which the robot must be located at the end of the plan may also be specified. For consistency, we assume that the robot must begin and end at a root landmark l_0, referred to as the robot base, throughout this paper.

C. Wong, E. Yang and X. T. Yan are with the Department of Design, Manufacture and Engineering Management, University of Strathclyde, Glasgow, G1 1XJ, UK. (e-mails: {cuebong.wong, erfu.yang, x.yan}@strath.ac.uk).

D. Gu is with the School of Computer Science and Electronic Engineering, University of Essex, Wivenhoe Park, Colchester C04 3SQ, UK (e-mail: dgu@essex.ac.uk).

III. DA-TPP APPROACH

In DA-TPP, the base planner (Fig. 1) pre-plans all possible paths before the execution of a task planner. The resulting path costs are then configured into the task planning problem as discrete movement action costs. This enables optimal task planning as true path plans are considered in the task planning phase. Indeed it can be inefficient to run an individual planning instance for each possible movement action. This is addressed through our previously developed multi-goal path planning algorithm based on T-RRT* trees [7], which simultaneously and efficiently finds feasible paths between all landmark pairs.

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 σ as a weighted sum of integral cost and path length:

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

Where n is the number of subdivisions of σ, $l()$ is the path length and w_a and w_b are weight factors for c and $l(\sigma)$, respectively. 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 and corresponding costs found for each of these actions ($c_p = \infty$ if no solution is found). In the interest of anytime planning, we note that a TPP solution can be found if all landmarks form a fully-connected graph such that any landmark $l_i \in L$ can be reached from every other landmark $l_j \in L | i \neq j$ by traversing the graph.

The task planning layer employs PDDL representation [8], and is solved using the openly available planner Local Planning Graphs (LPG-*td*) [9]. 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 [10]) developed for PDDL so long as they are compatible with *fluents* as a minimum requirement.

A. 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. This is supported by the base planner under the condition that sufficient paths are found to form a connected graph across all landmarks. Once an initial TPP solution is obtained, the path planning layer continues to iterate the path planning algorithm while the robot executes the initial solution. Suppose that the initial path cost for an action a is c_p. Following the work in [11], an upper cost bound C_s^+ is defined as:

$$C_s^+ = (1 - \eta_a) \cdot c_p. \quad (2)$$

Where η_a is a constant. When the path planning layer finds a new path for a with $c_p < C_s^+$, a new instance of task planning is called and c_p is updated to the new path cost. This mechanism guarantees that the task planning layer is called only when a guaranteed improvement to an action cost is found.

When running this anytime planner alongside robot 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 instance of task planning to reflect the next state of the world after executing the current action of the latest plan. In this way the robot always commits to the first action of each plan.

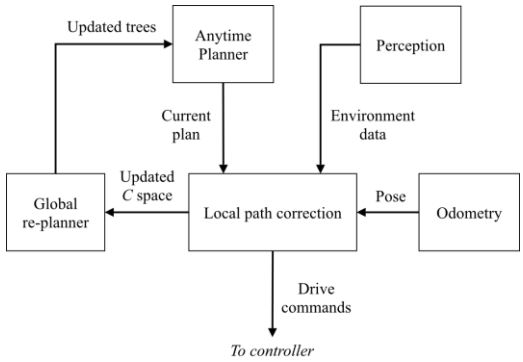


Figure 2. DA-TPP architecture

B. Dynamic anytime extension

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

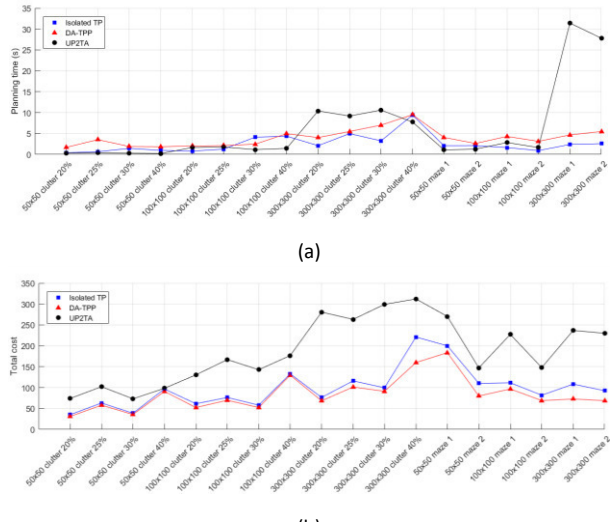
$$C_s^- = (1 + \eta_d) \cdot c_p'. \quad (3)$$

Where η_d is a constant. When the cost of the corrected path exceeds C_s^-, global re-planning takes place. This permanently updates all planning trees with the detected obstruction, and lazily finds a path to the robot's current location, if it exists, from every landmark. A new TPP solution is then produced from the updated path plans to generate a new optimal sequence of movement actions. The benefits of applying (3) as a re-planning condition is briefly discussed in section V. B.

Accordingly the DA-TPP provides the following behaviours in dynamic environments. When minor obstructions are encountered, only small adjustments to the planned path is required. 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. However, in situations where an obstruction causes significant diversion for a particular traverse (e.g. from road blockages), the likelihood of the obstruction affecting other paths are high and thus re-routing may produce new optimal task plans. In these situations it is necessary to update the entire plan to maintain optimality.

IV. PATH PLANNING

The path planning layer of the base planner is implemented following the approach described in [7]. Readers are directed to this initial work for a detailed description of the multi-tree T-RRT* algorithm. This section briefly discusses the dynamic algorithms for local path correction and global re-planning.



(a)

Figure 3. DA-TPP benchmarking results. (a) planning time, (b) overall plan cost

A. *Dynamic path correction (local)*

The local path re-planner corrects any single path according to procedures based on elements of the RRT^X algorithm [12]. 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. T_{new} is rooted at the goal 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 step eases the dynamic re-planning procedure as the root of the tree does not need to be updated at each time step as the robot advances along the path. As the robot traverses along this path, new environmental information is used to temporarily update the configuration space C. A feasibility check is used to determine if the updated configuration space invalidates the current path plan. If it does, the algorithm proceeds to update the path. Otherwise, the changes to C are discarded.

The algorithm invalidates all vertices that lie in the collision region of new obstacles. All descendant vertices that remain valid are then updated as *orphans*. This closely resembles the *propagateDescendants* function in [12]. Following this, the algorithm updates the connecting edges of the tree by iterating through a queue of vertices consisting initially of the neighbours of orphaned vertices (see *reduceInconsistency* function in [12]). For each of these vertices, the algorithm updates the parent of the current vertex, 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.

B. *Dynamic path correction (global)*

The global re-planner provides an update to all solutions of the path planning layer. When the condition in (3) is met, the optimality criteria for the current action-motion sequence may no longer hold and a new action-motion sequence must be determined by updating the costs of all movement actions. The algorithm first updates every tree by invalidating infeasible vertices, updating orphaned vertices and cascading a series of rewiring, as in section IV-A. New optimal paths between landmarks are obtained by finding new connecting vertices between corresponding pairs of trees. The set of best paths Σ_{best} are then 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}. Σ_{best} is then expanded to include these paths. Indeed the planner may not initially find a feasible path for each of these movement actions. However, this does not prevent the planner from obtaining a solution by making use of the anytime attributes of DA-TPP to find action-motion sequences despite certain actions possessing infinite costs.

V. Experimental Evaluation

A. *Base planner evaluation*

The base planner is benchmarked across a number of randomly generated cluttered and structured environments. We assess the performance of these approaches in 50 × 50, 100 × 100 and 300 × 300 environments. Note that in these experiments the cost function c in (1) for a given configuration q is given by $c = 1/\delta^2$, where δ is the distance to the nearest obstacle. Thus c describes the 'closeness' of q to an obstacle.

We compare the performance of the DA-TPP base planner with a simple planner consisting of task and path planning in isolation (isolated TP) and UP2TA. The isolated TP consists of a task planner that solves for an optimal action sequence by using the Euclidean distances between landmarks as movement action costs. For consistency in comparison, a single instance of a bi-directional T-RRT* algorithm is called for each movement action to obtain the final action-motion sequence. Our implementation of the UP2TA framework employs the greedy search algorithm [6] to obtain approximate cost metrics for each possible movement action. For consistency, LPG-*td* is then used to solve the task planning problem. An additional path planning layer based on Theta* [13] is required to obtain true paths for each movement action like in isolated TP.

Based on the results in Fig. 3, one may observe that the UP2TA fails to consider cost spaces and consequently performs notably worse than other planners when considering total path cost for the same planning instances. In terms of scalability, planning time highlights a key deficit of grid-based approaches: as the size of the problem increases, its performance generally decreases rapidly, as observed for environments of size 300x300. In particular, grid-based algorithms may become 'trapped' in enclosed regions that forces the algorithm to search through a large number of useless nodes needlessly. Isolated TP scales far 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. 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 (or even infeasible) movement actions are unknown to the symbolic planner.

In contrast, the DA-TPP consistently finds the lowest cost plans in all test cases. Although this sacrifices computational

efficiency, 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 × 300 environments. Finally, the quality of DA-TPP solutions may be further improved over time as a result of anytime planning, as discussed below.

B. *Anytime evaluation*

To analyse the anytime component of DA-TPP, the base planner was first run until an initial solution was obtained. Following the planner description in section III, the planner continues to run with η_a set to 0.03. Then, at defined time instances a new solution is requested from the planner. The corresponding overall cost of the task sequence at each of these time instances are shown in Fig. 4.

These planning results show improvements made to the overall solution at two levels. The observable decreasing 'step' behaviour corresponds to changes at the action sequence level. Here the planner identifies a new optimal task sequence that provides larger quality improvements as a result of higher quality paths being found for previously expensive actions. On the other hand, more minute improvements to the solution cost is attributed to local path quality improvements that do not alter the high-level task sequence. This observation provides support for the behaviour of the dynamic components of DA-TPP: small local changes to a path do not, in most cases, change the optimality of the task-level action sequence. Hence it is unnecessary to re-plan an entire action-motion sequence for small local path changes.

C. *Dynamic anytime evaluation*

Finally, the complete DA-TPP approach is assessed through simulations in the environment shown in Fig. 5, with obstacles not known a priori shown in blue. Unlike the experiment conducted for anytime evaluation, the robot begins executing a plan after an initial solution is obtained. Hence as actions are performed, new task plans become shorter and shorter until the robot finally executes all actions and return to base. We simulate real-time execution on the Gazebo simulator using a Husky mobile manipulator. Perception of the environment is achieved using a laser scanner with a range of 30 meters, while η_d is set to 0.05.

From Fig. 5, we note the following observations. Normally, the anytime planner without the dynamic component would force the robot to commit to the first action of a plan under all circumstances. However, the global re-planning unit allows the robot to follow a new optimal task sequence even part-way through a movement action if changes are detected. Secondly, the robot efficiently navigates past new obstacles and maintains global optimality by preserving previous knowledge in the path planning layer. This is not possible with the UP2TA, which would require planning from scratch in each instance.

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 toward globally-optimal solutions if sufficient time is allowed.

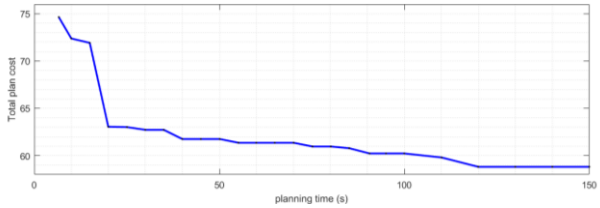
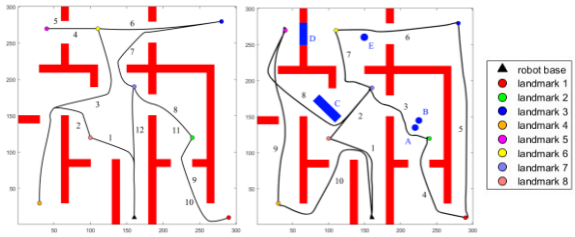


Figure 4. Anytime planning cost reductions



(a) (b)

Figure 5. DA-TPP dynamic replanning. (a) initial plan, (b) actual paths executed (unknown obstacles shown in blue)

ACKNOWLEDGMENT

This research is funded by the Engineering and Physical Sciences Research Council (EPSRC) under its Doctoral Training Partnership Programme (DTP 2016-2017 University of Strathclyde, Glasgow, UK).

REFERENCES

[1] A. Gasparetto, P. Boscariol, A. Lanzutti, and R. Vidoni, "Path planning and trajectory planning algorithms: A general overview," Springer, Cham, 2015, pp. 3–27.

[2] L. Pack Kaelbling and T. Lozano-Pérez, "Hierarchical task and motion planning in the now," in *2011 IEEE International Conference on Robotics and Automation*, 2011, pp. 1470–1477.

[3] C. R. Garrett, T. Lozano-Pérez, and L. P. Kaelbling, "FFRob: An efficient heuristic for task and motion planning," Springer, Cham, 2015, pp. 179–195.

[4] J. Ferrer-Mestres, G. Francès, and H. Geffner, "Combined task and motion planning as classical AI planning," Jun. 2017.

[5] 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.

[6] 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.

[7] 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," in *12th France-Japan and 10th Europe-Asia Congress on Mechatronics*, 2018, pp. 242–247.

[8] D. McDermott, "The PDDL planning domain definition language," *AIPS-98 Plan. Compet. Comm.*, 1998.

[9] 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.

[10] 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.

[11] D. Ferguson and A. Stentz, "Anytime RRTs," in *Proceedings of the 2006 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2006, pp. 5369–5375.

[12] 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.

[13] S. K. and A. F. K. Daniel, A. Nash, "Theta*: Any-angle path planning on grids," *J. Artif. Intell. Res.*, vol. 39, 2010.