Integrated Task and Motion Planning for Mobile Service Robots

Shih-Yun Lo, Shiqi Zhang and Peter Stone
University of Texas at Austin
{yunl;szhang;pstone}@cs.utexas.edu

I. Introduction

"Planning", or selecting a sequence of actions to achieve a goal, has been a core focus of interest within the field of Artificial Intelligence (AI) since the field was founded in the 1950's. Initially, the focus of attention was on *task planning* which is concerned with sequencing actions within a symbolic representation of the state space [2]. When action costs are further incorporated into the planning process, existing task planners can find the *optimal* plan that minimizes the overall plan cost. A largely independent thread of research has arisen on *motion planning* that focuses on producing a continuous motion while avoiding collision with obstacles in 2D or 3D continuous space [8]. Traditionally, motion planning has been concerned with computing a path connecting a single start configuration to a single goal configuration, without any concern for sequencing of subgoals.

Task and motion planning remained generally independent in large part for a long time, because until recently, physical robots have only been able to execute very short missions that could be solved entirely with motion planning algorithms. Meanwhile, complex task plans have needed to be generated in simulation or in purely software domains. However, with the recent advances in long-term autonomy on mobile robots in large-scale indoor environments [10, 6], there is a pressing need for the ability to generate task plans that are fully aware of, and indeed dependent upon, the grounded costs of actions that can only be determined by motion planning algorithms. In principle, motion planning costs could be evaluated on all the possible action sequences in the task planning space. However, especially in cases with combinatorially many possible sequences (for example if there are many possible separate places to buy coolers, ice, milk, and hot dogs), doing so is computationally infeasible.

The aim of this research is to fully integrate task and motion planning in order to find the lowest cost, optimal plan in task planning[1] in a computationally tractable manner. We develop a novel task and motion planning algorithm that has bi-directional communication between task and motion planners: the task planner is capable of computing a symbolic plan with the lowest cost conditioned on existing evaluations of action costs; and the motion planner is capable of evaluating the true cost of these constituent actions in the "lowest-cost" plan. This interactive process is repeated until a lowest-cost task plan is achieved such that all the plan's actions have been evaluated by the motion planner, and is thus guaranteed to achieve task-level optimality. The algorithm has been evaluated using a mobile service robot working on delivery tasks in an indoor environment in simulation. We observe significant improvements in overall efficiency compared to a baseline that evaluates costs of all actions at the motion planning level.

[1]Motion-level optimality is intractable in practice.

II. Related Work

Existing research on integrating task and motion planning has been largely focused on manipulation domains [1, 4, 7, 9]. For instance, symbolic plans computed by a task planner have been used to generate constraints for pruning the geometric search space [7], and a motion planner has been used to check the feasibility of symbolic actions and to update the task planner accordingly [9]. Task and motion planners have been integrated in belief space to account for current-state and future-state uncertainty [4]. All the above work focused on manipulation tasks, presumably because (from the viewpoint of motion planning) 3D manipulation problems are more challenging than 2D navigation problems: task planning techniques can thus be useful for "guiding" motion planning.

Recent advances in long-term autonomy have enabled mobile robots to provide service to people in large-scale indoor environments [10, 6]. In such domains, task planners need to represent large numbers of rooms, humans, objects and their locations, which soon becomes computationally infeasible. Therefore, in this paper, we focus on integrating task and motion planning in large-scale indoor environments and develop a novel algorithm that, for the first time, significantly improves the overall efficiency while maintaining a guarantee of task-level optimality.

III. Problem Statement

A task and motion planning (TMP) problem requires domain descriptions at both a symbolic level for task planning and a geometric level for motion planning.

Symbolic-level Domain Description: $\mathcal{D}^{sym}$

$\mathcal{D}^{sym}$ specifies a task planning domain that includes a set of states, S, and a set of actions A. We assume a factored state space such that each state is defined by the values of a fixed set of variables; each action $a \in A$ is defined by its preconditions and effects. A cost function $Cost$ maps a state-action pair to a real number such that $Cost(\langle s,a \rangle) \rightarrow \mathbb{R}$ represents the cost of action a being executed in state s.

Given domain $\mathcal{D}^{sym}$, a task planning problem is defined by an initial state $s^{init} \in S$ and a specification of the goal that corresponds to a set of goal states $S^G \subseteq S$. Solving a task planning problem produces plan p^* that has the lowest cost among the plans that can lead state transitions from s^{init} to $s^{goal} \in S^G$. p^* is called the *optimal* plan. A plan $p \in P$ that includes a sequence of actions and states before and after each action can be represented as: $p = \langle s_0, a_0, \cdots, s_{N-1}, a_{N-1}, s_N \rangle$. where P is the set of satisfactory plans. We compute p^* as:

$$p^* = \text{argmin}_{p \in P} \Sigma_{\langle s,a \rangle \in p} Cost(\langle s, a \rangle)$$

Geometric-level Domain Description: $\mathcal{D}^{geo}$

$\mathcal{D}^{geo}$ specifies a motion planning domain, where we directly search in the 2D workspace (instead of higher-dimensional configuration space), because in this work we focus on only 2D navigation problems for motion planning. Given $\mathcal{D}^{geo}$, a motion planning problem can be specified by an initial position x^{init} and a goal position x^{goal}. The 2D space is represented as a region in Cartesian space such that the position and orientation of the robot can be uniquely represented as a *pose* $(\mathbf{x}, \theta)$. Some parts of the space are designated as free space, and the rest is designated as obstacle.

The motion planning problem is solved by algorithm $\mathcal{A}$ to compute a collision-free trajectory ξ^* (connecting $\mathbf{x}^{init}$ and $\mathbf{x}^{goal}$ taking into account any motion constraints on the part of the robot) with minimal trajectory length $Len(\xi) = L$. We use Ξ to represent the trajectory set that includes all satisfactory trajectories. The *optimal* trajectory is

$$\xi^* = \text{argmin}_{\xi \in \Xi} Len(\xi), \text{ where } \xi(0) = \mathbf{x}_{init} \text{ and } \xi(L) = \mathbf{x}_{goal}$$

Connecting $\mathcal{D}^{sym}$ and $\mathcal{D}^{geo}$ in TMP problems

A symbolic state s in $\mathcal{D}^{sym}$ corresponds to a geometric constraint in $\mathcal{D}^{geo}$ that can be represented as a set of positions X in 2D space. For instance, the symbol of "beside a table" corresponds to a (infinite) set of positions within a range of the table. This mapping from s to X is represented as function f: $X = f(s)$. Given function f, each state transition, $\langle s, a, s' \rangle$, at the symbolic level can be realized as a motion planning problem $\langle x, \xi, x' \rangle$ at geometric level: we first use f to map states s and s' to position sets X and X', and then arbitrarily select $x \in X$ and $x' \in X'$.

Therefore, the input of a TMP problem is a five-tuple

$$\langle \mathcal{D}^{sym}, \mathcal{D}^{geo}, s^{init}, S^G, \mathbf{x}^{init}, f \rangle$$

where $\mathbf{x}^{init} \in f(s^{init})$ meaning that the geometric initial position must be consistent with the symbolic initial state.

A satisfactory output of a TMP problem is a two-tuple,

$$\langle p, [\xi_0, \xi_1, \cdots, \xi_{N-1}] \rangle$$

that includes a symbolic plan and a set of trajectories, where $p(0) = s^{init}$, $p(N) \in S^G$, $|p| = N$, $\xi_0(0) = \mathbf{x}^{init}$, $\xi_i(0) \in f(s_i)$, and $\xi_i(T_i) \in f(s_{i+1})$ for $0 \leq i \leq N-1$.

Finally, we define the task-level optimal plan to be the lowest-cost plan p^*, conditioned on the motion planning algorithm $\mathcal{A}$ and constraint function f:

$$p^* = \text{argmin}_{p \in P} \Big(\sum_{0 \leq i < |p|} Len(\xi_i) | \mathcal{A}, f \Big)$$

where $\xi_i = \mathcal{A}(\langle s_{i-1}, a_{i-1}, s_i \rangle)$ is the trajectory returned by $\mathcal{A}$ given state transition $\langle s_{i-1}, a_{i-1}, s_i \rangle \in p$.

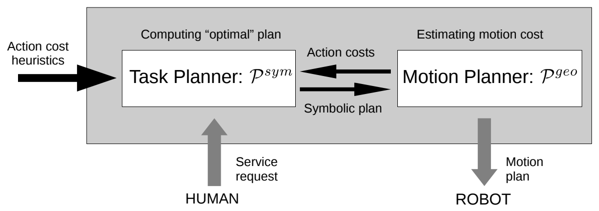

Fig. 1. Overview of our algorithm for efficiently solving TMP problems, with guarantee of task-level optimality.

IV. Algorithm

Our TMP algorithm is summarized in Figure 1. A task planner, $\mathcal{P}^{sym}$, and a motion planner, $\mathcal{P}^{geo}$, serve as the two main components of our algorithm. The task planner interacts with humans by taking their service requests and generating human-understandable symbolic plans. We apply answer set programming (ASP) [3] for its strong capability of knowledge representation to formalize our task domain and query human inputs (our algorithm is not restricted to specific task planners). We initialize the cost function of $\mathcal{P}^{sym}$ using a loose lower-bounded heuristic. Our motion planner is used for generating motion plans for robots and for evaluating more realistic motion costs of each action in symbolic plans.

The task planner uses a lower-bounded heuristic action cost function, h, for initializing its cost function. h maps each motion planning problem (specified by a symbolic state-state pair) to a cost value that is less than or equal to the real cost:

$$h(\langle x, x' \rangle) \leq C(\langle x, x' \rangle).$$

where C reports the real cost of a 2D motion planning problem.

In our framework, h computes the Euclidean distance between x and x'. It is a good lower bound because motion cost cannot be smaller than the Euclidean distance and it satisfies our inexpensive-computation expectation.

$$h(\langle x, x' \rangle) = ||x - x'||_2,$$

Algorithm 1 shows our algorithm for TMP problems. It requires a task planner $\mathcal{P}^{sym}$ and a motion planner $\mathcal{P}^{geo}$ for planning at the symbolic and geometric levels respectively. A TMP problem is specified by its initial state and goal specification at symbolic level: s^{init} and S^G. The output is a symbolic plan that with guarantee of task-level optimality, which is conditioned on the function that maps each task-level state to a motion-level point and the motion planner.

V. Experiments

We have conducted experiments in an indoor office environment using a map that is generated using a real robot. We focus on evaluating the overall efficiency (both task and motion planning) by comparing our algorithm to a baseline algorithm that evaluates all possible navigation actions. Here,

Algorithm 1 Our algorithm for TMP problems

```
Input: Task planner $\mathcal{P}^{sym}:(s,s',C)\rightarrow p$, where $C$ is a cost function
Input: Motion planner $\mathcal{P}^{geo}:\langle x,x'\rangle\rightarrow\xi$
Input: Initial state $s^{init}$, and goal specification $S^G$
Input: Lower-bounded heuristic function $h:\langle x,x'\rangle\rightarrow\mathbb{R}$
Output: Symbolic plan $p$ that is optimal at task level
 1: Initialize motion-cost evaluation function $C$ with $h$: $C\leftarrow h$
 2: while true do
 3:     $p\leftarrow\mathcal{P}^{sym}(s^{init},S^G,C)$
 4:     Initialize termination flag: $term =$ true
 5:     for each $\langle s,a,s'\rangle\in p$ do
 6:         Map symbolic states to 2D points: $x\leftarrow f(s)$; $x'\leftarrow f(s')$
 7:         if $C(x,x') == h(x,x')$ then
 8:             Evaluate motion cost: $C(x,x')\leftarrow Len(\mathcal{P}^{geo}(x,x'))$
 9:             $term =$ false
10:         end if
11:     end for
12:     break if $term$ is true
13: end while
14: return $p$
```

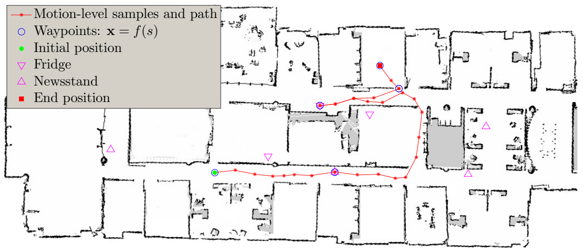

Fig. 2. An illustration of applying our algorithm to a TMP problem. The robot needs to collect newspaper from one of the newsstands (upward triangles) and juice from one of the fridges (downward triangles), and then deliver to a person whose position is marked with a solid red dot.

it is assumed that non-navigation actions (e.g., *load* and *deliver* actions) have no time consumption. Our hypothesis is that our algorithm requires significantly less time than the baseline and scales well in domains that include more objects.

Figure 2 shows the occupancy-grid map generated by running a SLAM algorithm on a real robot. At task level, we formalize a set of areas such as rooms and corridors and a set of objects that include doors, persons, and containers (fridges and newsstands). At the motion level, we apply the probabilistic roadmap method [5] along with A^* to search in a space that includes random samples. The two levels are connected through waypoints at motion level (a subset is shown as blue hollow circles on the map) that each corresponds to an object at task level. The red-star path in Figure 2 shows an illustrative result generated by our algorithm, where the robot needs to collect a newspaper and a bottle of juice and then deliver them to a target person (solid red dot).

We compare our algorithm to a baseline that pre-computes costs of all navigation actions. Both algorithms can guarantee task-level optimality. Our algorithm aims to improve overall computational efficiency. Figure 3 shows the quantitative results. For instance, when only two newsstands and two fridges are in the domain, the baseline requires more than 50s while our algorithm needs less than 15s. We attribute this efficiency improvement to the fact that our algorithm enables the motion planner to evaluate costs of only a small number of navigation actions. For example, in the last trial, out of 3081 navigation actions from one location to another, only 15 (0.52%) were

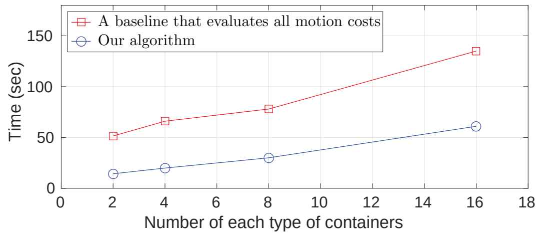

Fig. 3. Results of comparing our algorithm to a baseline that computes costs of all navigation actions. The x-axis represents the number of containers in the environment: the numbers of newsstands and fridges are the same. The y-axis corresponds to the overall time (sec) needed by task and motion planners.

evaluated towards finding the optimal task plan.

VI. Conclusions

In this paper, we propose a novel algorithm that fully integrates task and motion planning for navigational tasks, with the guarantee of task-level optimality. We introduce a lower-bounded heuristic for initializing the cost function of our task planner. The task planner computes an "optimal" plan using the current cost function and the motion planner keeps improving the cost function by evaluating the real cost of navigation actions. This process is iterated until all action costs in a plan have been evaluated when we report the optimal symbolic plan. We compare this algorithm with a baseline that pre-computes all action costs, and experimental results suggest that our algorithm performs better in overall efficiency.

Acknowledgment

This work has taken place in the Learning Agents Research Group (LARG) at UT Austin. LARG research is supported in part by NSF (CNS-1330072, CNS-1305287), ONR (21C184-01), and AFOSR (FA9550-14-1-0087).

References

[1] Esra Erdem, Kadir Haspalamutgil, Can Palaz, Volkan Patoglu, and Tansel Uras. Combining high-level causal reasoning with low-level geometric reasoning and motion planning for robotic manipulation. In *International Conference on Robotics and Automation (ICRA)*, 2011.

[2] Richard E Fikes and Nils J Nilsson. Strips: A new approach to the application of theorem proving to problem solving. *Artificial intelligence*, 2(3-4):189–208, 1971.

[3] Michael Gelfond and Yulia Kahl. *Knowledge representation, reasoning, and the design of intelligent agents: The answer-set programming approach*. Cambridge University Press, 2014.

[4] Leslie Pack Kaelbling and Tomás Lozano-Pérez. Integrated task and motion planning in belief space. *International Journal of Robotics Research*, 32(9-10):1194–1227, 2013.

[5] Lydia E Kavraki, Petr Švestka, Jean-Claude Latombe, and Mark H Overmars. Probabilistic roadmaps for path planning in high-dimensional configuration spaces. *Robotics and Automation, IEEE Transactions on*, 12(4):566–580, 1996.

[6] Piyush Khandelwal, Samuel Barrett, and Peter Stone. Leading the way: An efficient multi-robot guidance system. In *International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, 2015.

[7] Fabien Lagriffoul, Dimitar Dimitrov, Julien Bidot, Alessandro Saffiotti, and Lars Karlsson. Efficiently combining task and motion planning using geometric constraints. *The International Journal of Robotics Research*, 33(14):1726–1747, 2014.

[8] Jean-Claude Latombe. *Robot motion planning*. Springer Science & Business Media, 2012.

[9] Sanjeev Srivastava, Eugene Fang, Lorenzo Riano, Rohan Chitnis, Stephen Russell, and Pieter Abbeel. Combined task and motion planning through an extensible planner-independent interface layer. In *International Conference on Robotics and Automation (ICRA)*, 2014.

[10] Manuela Veloso, Joydeep Biswas, Brian Coltin, and Stephanie Rosenthal. Cobots: robust symbiotic autonomous mobile service robots. In *International Joint Conference on Artificial Intelligence (IJCAI)*, 2015.