A Flexible Sampling-Based Approach to Task and Motion Planning

Wil Thomason
Department of Computer Science
Cornell University
Email: wbthomason@cs.cornell.edu

Ross A. Knepper
Department of Computer Science
Cornell University
Email: rak@cs.cornell.edu

I. Introduction

Generalized robot autonomy requires robots to plan solutions for complex and varied tasks involving interaction with the physical world. Traditionally, the (continuous) geometric and (discrete) symbolic aspects of the problem are studied independently; motion planners solve geometric problems, whereas task planners solve symbolic problems. **Integrated Task and Motion Planning** (TAMP) seeks to unify the planning problem using information from each component to ease the solution of the other.

Most TAMP approaches focus on information flow between task and motion planners; the majority of these approaches integrate the motion planning layer into the task planning layer as a source of constraints and/or a validator for task plans [1, 2, 5, 8]. Validating task plans with a motion planner may require repeated expensive executions of both planners. Getting constraints from the motion planner is more efficient but is often limited by the manual selection of a restricted subset of motion constraints to integrate in the task layer. Both of these issues affect not only the efficacy of a planner but also its robustness — planning for or responding to action failures and uncertainty is difficult if doing so is computationally heavy or if the system does not capture all reasons for action failure.

We present a novel approach to TAMP that fully embeds a symbolic abstraction of a planning problem into the continuous representation of the problem. This fused representation allows a single sampling-based motion planner to solve both problems simultaneously and efficiently in a single invocation while implicitly capturing all geometric/kinematic constraints on actions. To make this process efficient, we contribute methods for: 1) a continuous semantics for logical formulae in configuration space, 2) a factorization of the sampling problem to mitigate a dimensional explosion, and 3) the use of standard task planning heuristics as sampling biases. Our approach lends itself to efficient replanning in the face of failure, which we discuss as a possible extension to the technique.

II. Approach

Two ideas form the core of our approach: the **composite configuration space** and the notion of **"unsatisfaction" semantics** for Boolean predicates.

A. *Composite Space*

The composite configuration space is a simple idea generalizing the concept of modes from the hybrid systems planning literature as well as the manipulation graph concept from traditional manipulation literature.

Definition 1 (Composite Space)**.** Take $\mathbf{C}_R$ to be the configuration space of a robot R, $\mathcal{O}$ to be a set of manipulable objects, and $\mathcal{B}$ to be the set of ground Boolean atoms in a symbolic planning domain. Then the *composite space* corresponding to the environment and domain is

$$\mathcal{C} = \mathbf{C}_R \bigtimes_{\mathcal{O}} \mathrm{SE}(3) \bigtimes_{\mathcal{B}} \mathbb{Z}_2$$

That is, the Cartesian product of the ordinary robot configuration space with a copy of SE(3) for each movable object and a Boolean domain for each ground atom.

The advantages of planning in composite space are that: 1) we can use an ordinary motion planning algorithm to solve both the symbolic and geometric parts of the problem simultaneously, 2) the composite space implicitly encodes all kinematic, geometric, and symbolic constraints on the problem, and 3) exploring symbolic and geometric states in tandem helps both parts of the problem guide the search of the other. The composite space can be conceptualized as a copy of continuous space for each setting (total assignment of values) of the symbolic state in the domain. Symbolic actions "move" between these copies. The planning problem is then that of finding valid paths between states where symbolic actions can be taken. This idea is similar to that of "modes" used by e.g. Hauser and Latombe [3] and Vega-Brown and Roy [9]; our composite space can represent both contact points and purely symbolic states inducing changes to the manifold of feasible configurations. Each copy is geometrically independent of each other copy; thus we can factor the configuration space by settings of the symbolic state dimensions and consider only those settings which we reach directly in the course of planning by applying symbolic actions from the initial symbolic state.

B. *Unsatisfaction Semantics*

All TAMP approaches require a "bridge" between the symbolic and geometric parts of the problem — some way to translate a symbolic state to a state in the real world and vice versa. Often this bridge takes the form of a domain-specific black box. With an eye toward generalizing this, we introduce *unsatisfaction semantics* as an extensible means of geometrically interpreting symbolic state. The basic idea is that we can define an alternative semantics for logical operators (e.g. $<, \leqslant, >, \geqslant, =, \wedge, \vee$) describing, for a given state and formula, the Euclidean distance in physical space to the closest state where the formula holds. Given implementations of primitive predicates (e.g. `on`, `at`, etc.) using these operators, we can automatically derive a representation for arbitrary logical formulae (using the operators and predicates we have defined). While this does not entirely escape the need for some domain-specificity, it shifts the level of specification toward the abstract (and reusable).

Unsatisfaction semantics lets us solve (using gradient descent or any other optimization approach) for states in continuous space which satisfy arbitrary logical formulae. This presents an efficient means of locating states satisfying the preconditions for symbolic actions.

C. A Sampler-Only Algorithm for TAMP in Composite Space

We combine these ideas to create an efficient algorithm for solving TAMP problems. Solving TAMP problems is equivalent to motion planning in $\mathcal{C}$ and unsatisfaction semantics can efficiently guide a motion planner to states where symbolic actions can be taken; all that's missing is a means of deciding which symbolic actions are important. We thus assume that we are given some black-box satisfying the interface: 1) Given a setting of symbolic values $s \in \bigtimes_{\mathcal{B}} \mathbb{Z}_2$, the black box returns a list of actions which may be viable given s and 2) (Optionally) the list of actions returned are ranked by "priority" (roughly, how important the black box thinks the actions are to the problem solution). We use the "helpful actions" heuristic from the FF planner [4] to provide this interface in our proof of concept implementation. Given such a black box, our algorithm is shown in Algorithm 1. In short: Select between uniform random sampling in composite space and sampling directly in the precondition region for an action the black box says is important. This has the effect of biasing the planner toward interesting areas of composite space.

Note that our algorithm only defines a sampler and is agnostic to (a) the planning algorithm used, (b) the specifics of action suggestion, and (c) the semantics of the predicates. This allows us a great degree of flexibility in exploring options for different planning problems. We have implemented Algorithm 1 in C++, choosing RRT [7] as our sampling-based planner and the helpful actions heuristic from FF [4] as our black-box heuristic. We find that this implementation achieves competitive performance on benchmarks from the set proposed in Lagriffoul et al. [6].

Algorithm 1: Composite Space Sampling

```
Output : Sampled state in C
if BiasedCoin() is True then
    return NormalSample()
else
    return HeuristicSample()
end
function NormalSample()
    // Symbolic state
    u ← UniformRandom(U);
    // Valid pose
    p ← UniformRandom(ValidPoses(u));
    // Robot config
    r ← UniformRandom(C_Robot);
    return MakeConfiguration(r, p, u);
end
function HeuristicSample()
    repeat
        c′ ← NormalSample();
        h ← PrioritizedActions(c′);
        a ← PrioritySample(h);
        f ← Precondition(a);
        c ← Solve(c′, f);
    until Satisfies(c, Precondition(a));
    UpdateInfo(c.u, GetPoses(c));
    UpdateLog(c, a);
    return c;
end
```

III. Extensions for Robustness

Though our TAMP technique assumes deterministic actions and ignores dynamics (we make the common quasi-static assumption), it is easily extended to efficiently respond to action failure. Responding to action failure has two main challenges: Determining the resulting state of the world and planning to a solution from this new state without starting over from scratch. The predicate semantics implementations we use can also be used to find the new logical state for a system after an action fails by determining the continuous state and testing it for each implemented predicate. Further, once we have the correct post-failure state, we can continue sampling just as before, reusing the previous sampling effort (in the form of the motion planning tree/roadmap and information about important actions and action precondition regions) to find a new solution. This resumption of planning is easy because our technique does not commit to a single task plan for a given problem, but explores many options in parallel, making it easier to find a plan that still works after an action has failed. This approach to handling action failure is desirable as it does not require any up-front cost (as would e.g. making a branching tree of plans based on possible action failures) and can still efficiently recover from failures if they happen.

References

[1] Neil T. Dantam et al. "Incremental Task and Motion Planning: A Constraint-Based Approach". In: *Robotics: Science and Systems XII*. 00052. 2016. ISBN: 978-0-9923747-2-3. DOI: 10.15607/RSS.2016.XII.002.

[2] Kris Hauser and Jean-Claude Latombe. "Integrating Task and PRM Motion Planning: Dealing with Many Infeasible Motion Planning Queries". In: (). 00048, p. 8.

[3] Kris Hauser and Jean-Claude Latombe. "Multi-Modal Motion Planning in Non-Expansive Spaces". In: *The International Journal of Robotics Research* 29.7 (June 2010), pp. 897–915. ISSN: 0278-3649, 1741-3176. DOI: 10 . 1177 / 0278364909352098. URL: http : / / journals . sagepub . com / doi / 10 . 1177 / 0278364909352098 (visited on 06/11/2019).

[4] Jörg Hoffmann and Bernhard Nebel. "The FF Planning System: Fast Plan Generation through Heuristic Search". In: *Journal of Artificial Intelligence Research* 14 (2001). 00000, pp. 263–312. ISSN: 10769757. DOI: 10/gcp3qj. arXiv: 1106.0675. URL: https://arxiv.org/abs/1106.0675.

[5] Fabien Lagriffoul et al. "Constraint Propagation on Interval Bounds for Dealing with Geometric Backtracking". In: *IEEE International Conference on Intelligent Robots and Systems*. 00058. 2012, pp. 957–964. ISBN: 978-1-4673-1737-5. DOI: 10/f3nsfp.

[6] Fabien Lagriffoul et al. "Platform-Independent Benchmarks for Task and Motion Planning". In: *IEEE Robotics and Automation Letters* 3.4 (Oct. 2018). 00000, pp. 3765–3772. ISSN: 2377-3766, 2377-3774. DOI: 10.1109/LRA.2018.2856701. URL: https://ieeexplore.ieee.org/document/8411475/ (visited on 09/05/2018).

[7] Steven M. Lavalle. "Rapidly-Exploring Random Trees: A New Tool for Path Planning". In: (May 1999). URL: http://msl.cs.illinois.edu/~lavalle/papers/Lav98c.pdf.

[8] S. Srivastava et al. "Combined Task and Motion Planning through an Extensible Planner-Independent Interface Layer". In: *2014 IEEE International Conference on Robotics and Automation (ICRA)*. 2014 IEEE International Conference on Robotics and Automation (ICRA). 00153. May 2014, pp. 639–646. DOI: 10.1109/ICRA.2014.6906922.

[9] William Vega-Brown and Nicholas Roy. "Asymptotically Optimal Planning under Piecewise-Analytic Constraints". In: WAFR. 00019. 2016, p. 16.