Improvements in Synergistic Framework for Manipulation Tasks

Keliang He Lydia Kavraki

I. Introduction

Manipulation tasks are abundant in today's robotic applications, especially in the areas of manufacturing, warehouse management, and domestic robots. For these types of problems, the robot manipulator must interact with the objects in the environment to achieve the task. Automated motion planning for these tasks is highly desirable, as it would allow for a much wider range of tasks achievable by a single robot.

There are two key problems that must be addressed for manipulation planning. The first is the problem of task planning, where the planning algorithm must consider the discrete actions available in the domain, and find a sequence of actions (action plan) to achieve the task. The actions may include reaching for objects, carrying objects, pushing objects, etc.. There has been much work in the area of planning in the artificial intelligence community [1, 12, 13, 19, 20, 28] to solve this problem. The other problem that must be addressed is motion planning, where continuous motions must be found to implement the actions in the action plan. For manipulation problems, the robots often have more than 6 degrees of freedom. Sampling-based motion planning [14, 21, 23–25, 31] has proven to be the method of choice for these high dimensional problems, for its ability to quickly explore such high dimensional spaces.

Many frameworks have been proposed for combining task and motion planning in the context of manipulation planning [2, 4, 6–8, 10, 15–18, 22, 26, 27, 29, 30, 32]. Many of these, however, lack completeness guarantees or have strong assumptions for any guarantee [4, 6–8, 15–18, 26, 27, 30, 30, 32]. This is in many cases a result of sampling-based motion planners being probabilistic complete rather than complete. One approach we previously proposed, the Synergistic Framework for manipulation tasks [11], solves the manipulation planning problem while still guaranteeing probabilistic completeness. However, it suffered from poor scalability, and experiments were limited to less than five movable objects. In this abstract, we present our efforts in improving the scalability of the Synergistic Framework by using heuristic to guide task planning, allowing it to solve problems with up to 15 objects.

II. Background

A. Synergistic Framework

The Synergistic Framework approaches manipulation planning in the following manner: The task is specified using a co-safe linear temporal logic (co-safe LTL) formula. The atomic propositions of the formula are elements of $O \times \mathcal{L}$, representing that an object $o \in O$ is at some location whose labels contains $l \in \mathcal{L}$. Using an external tool Spot [5], the formula is converted into a deterministic finite automaton (DFA) whose transition criteria are propositional logic formula over the atomic propositions. Alternatively, the task could be specified directly using a DFA.

The manipulation domain is captured using an implicit graph called the abstraction. Each node in the abstraction maintains information about the current (discrete) locations of the objects and the end-effector, as well as the current action the robot is performing. Edges in the abstraction graph represent motions or transitions in robot execution that could be taken to change the world from one state to the next.

Task planning is performed by running Dijkstra's algorithm over the product of the DFA and the abstraction. A node in the product graph is a pair (z, v), where z is the DFA state, and v is the abstraction node. This product is constructed as needed starting from the initial state of the DFA and the abstraction state that represent the initial configuration of the world. Task planning terminates and returns a sequence of abstraction nodes, called the task plan, when a node whose DFA state is the accepting state is expanded.

The sequence of abstraction nodes is then split into motion segments, and they are planned for using a sampling-based algorithm RRT-Connect [23] from the Open Motion Planning Library (OMPL [3]). If a motion plan is found for each required motion, then the concatenation of these plans is returned as the solution. Otherwise, the weights of the abstraction graph is updated according to the successfulness of motion planning. For actions with continuous motion found, their weights are reduced to 0. For actions that failed, their weights are increased proportional to the amount of time spent in motion planning. This enables the task planner to generate new task plans which are more likely to be implemented by the motion planner. This framework was shown to be probabilistic complete [11].

B. Performance Issues

The Synergistic Framework as proposed in [11] for manipulation planning, however, suffered from poor scalability. The results showed that the framework scaled to problems with four objects in eight locations, with a task that converted to a DFA with 27 states. The main bottleneck lies within searching the product graph in task planning. The number of nodes explored grows exponentially with the depth of the search, making planning for long tasks difficult. Additionally, considering a larger number of objects and locations increase

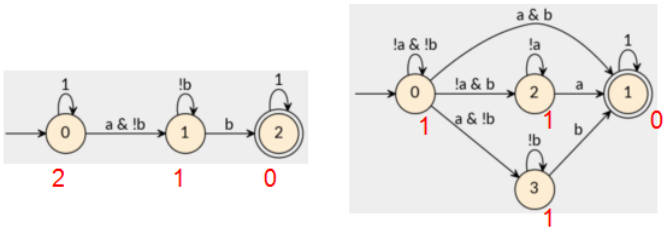

Fig. 1. DFA for two tasks and the heuristic value generated by the transition-based heuristic (in red).

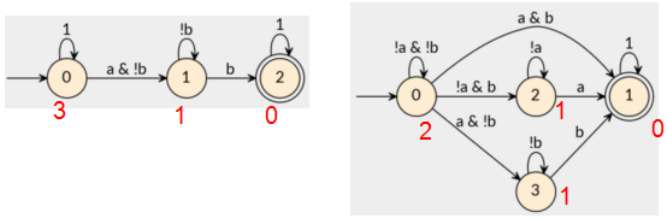

Fig. 2. DFA for two tasks and the heuristic value generated by the literal-based heuristic (in red).

the branching factor of the search process, also slowing down the search.

III. Heuristic Search in Synergistic Framework

To improve the scalability of the Synergistic Framework, we introduce two heuristics to guide the search of the product graph. Even though the number of nodes in the product graph is large, the size of the DFA is relatively small, generally no more than a few hundred nodes. Therefore we generate heuristics that focus on the DFA component z of a product graph state (z, v). If it is easy to reach an accepting state from z, then it should have a smaller cost-to-go heuristic than a state from which it is difficult to reach an accepting state.

The first heuristic is transition based. We count the minimum number of transitions in the DFA needed to reach an accepting state from the current state z. Figure 1 shows the heuristic values generated using such a heuristic. For the DFA on the left, the transition-based heuristic seems reasonable. It suggests that it is easier to reach the accepting state 2 from state 1 compared to state 0. On the other hand, for the DFA on the right, which expresses the task of achieving two propositions in any order, the transition-based heuristic is not very informative. It ranks all of the states the same, as a transition from the initial state to the accepting state is possible if all the propositions become true at the same time.

In practice, this is rarely the case. The manipulator often could only move one object at a time, so the truth value of propositions also change one at a time. Therefore, we introduce a second heuristic based on the minimum number of literals we need to check in order to reach an accepting state. Figure 2 shows the values from such a heuristic. We can see that using this new heuristic, we can separate the initial state out from the rest in the DFA on the right.

Due to the fact that in the Synergistic Framework, actions that have been found would have their costs reduced to 0, both the transition-based and literal-based heuristics are inadmissible. This causes the task plan to potentially be non-optimal. The non-optimality could cause the planner to not take full advantage of using actions whose continuous motions are already found. We did not observe this to happen in practice, as continuous motion already found are results of being parts of previously-optimal task plans, preceding the point of failure for these plans. Thus in future iterations when the heuristics are inadmissible, these actions are still considered before other choices.

IV. Results

We test our heuristics using the A* [9] planner and compare the runtime of a single task planning run against the original planner, which is running Dijkstra's algorithm. Note that Dijkstra's algorithm is equivalent to running A* with the heuristic $\forall x, H(x) = 0$. The experiments were performed on several benchmark problems. The first is a task where the goal is to move a single object to a new location. The location is occupied, so removal of another object is needed. The second is the Baxter example from [11]. And finally, we test on a set of benchmarks where $n - 1$ objects are in n locations, and their places need to be swapped. The average runtime over 10 runs are shown in Figure 3.

We see that in all cases, using heuristics help with speeding up the task planner. The literal-based heuristic performs well across the board, achieving order of magnitude speed-up over the previous framework. This allows us to solve difficult problems such as the swapping task that involves 128 DFA states.

V. Conclusion

We have presented our efforts in improving the scalability of the Synergistic Framework for manipulation problems by using heuristics generated from the planning task to guide the search in task planning. We have found that the minimum number of propositions required for the task to be a very good heuristic for manipulation tasks. Using these heuristics, we were able to achieve order of magnitude speed-up over the previous planner. We are also actively exploring ways to improve motion planning performance, as well as enhancing the synergy between task and motion planning to achieve better performance.

References

[1] Avrim L. Blum and Merrick L. Furst. Fast Planning Through Planning Graph Analysis. *Artificial Intelligence*, pages 90:281–300, 1997.

[2] Stephane Cambon, Rachid Alami, and Fabien Gravot. A hybrid approach to intricate motion, manipulation and task planning. *The International Journal of Robotics Research*, 28(1):104–126, 2009.

[3] Ioan A. Şucan, Mark Moll, and Lydia E. Kavraki. The Open Motion Planning Library. *IEEE Robotics & Automation Magazine*, 19(4):72–82, December 2012.

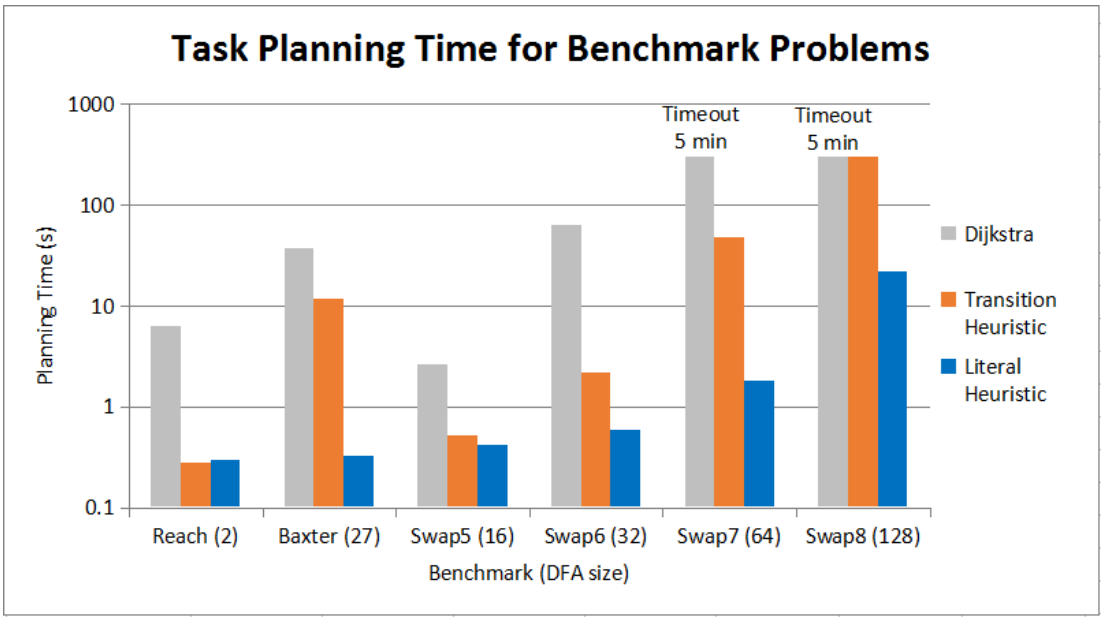

Fig. 3. Average runtime for a single iteration of task planning. Results shown in log scale. The size of the DFA is shown in parenthesis next to benchmark. A five min timeout was used in the experiments.

[4] Christian Dornhege, Patrick Eyerich, Thomas Keller, Sebastian Trüg, Michael Brenner, and Bernhard Nebel. Semantic attachments for domain-independent planning systems. *Springer Tracts in Advanced Robotics*, 76 (STAR):99–115, 2012.

[5] Alexandre Duret-Lutz and Denis Poitrenaud. Spot: An extensible model checking library using transition-based generalized buchi automata. In *Int. Symp. on Modeling, Analysis, and Simulation of Computer and Telecommunications Systems*, pages 76–83. IEEE, 2004.

[6] 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. *Proceedings - IEEE International Conference on Robotics and Automation*, pages 4575–4581, 2011.

[7] Caelan Reed Garrett, Leslie Pack Kaelbling, and Tomas Lozano-Pérez. FFRob : An efficient heuristic for task and motion planning. *WAFR*, pages 179–195, 2014.

[8] Fabien Gravot, Stephane Cambon, and Rachid Alami. asymov a planner that deals with intricate symbolic and geometric problems. In *Robotics Research. The Eleventh International Symposium*, pages 100–110. Springer, 2005.

[9] Peter E Hart, Nils J Nilsson, and Bertram Raphael. A formal basis for the heuristic determination of minimum cost paths. *Systems Science and Cybernetics, IEEE Transactions on*, 4(2):100–107, 1968.

[10] Kris Hauser and JeanClaude Latombe. Integrating task and prm motion planning: Dealing with many infeasible motion planning queries. *International Conference on Automated Planning and Scheduling*, (1):34–41, 2009.

[11] Keliang He, Morteza Lahijanian, Lydia E. Kavraki, and Moshe Y. Vardi. Towards manipulation planning with temporal logic specifications. In *IEEE Conference on Robotics and Automation*, pages 346–352, May 2015.

[12] Malte Helmert. The fast downward planning system. *Journal of Artificial Intelligence Research*, 26:191–246, 2006.

[13] Jörg Hoffmann. FF: The fast-forward planning system. *AI magazine*, 22:57–62, 2001.

[14] D. Hsu, J.-C. Latombe, and R. Motwani. Path planning in expansive configuration spaces. *Intl. J. of Computational Geometry and Applications*, 9(4-5):495–512, 1999.

[15] William N N Hung, Xiaoyu Song, Jindong Tan, Xiaojuan Li, Jie Zhang, Rui Wang, and Peng Gao. Motion Planning with Satisfiability Modulo Theories. *Proceedings - IEEE International Conference on Robotics and Automation*, pages 113–118, 2014.

[16] Leslie Pack Kaelbling and Tomas Lozano-Pérez. Hierarchical task and motion planning in the now. *Proceedings - IEEE International Conference on Robotics and Automation*, pages 1470–1477, 2011.

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

[18] Lars Karlsson, Julien Bidot, Fabien Lagriffoul, Alessandro Saffiotti, Ulrich Hillenbrand, and Florian Schmidt. Combining Task and Path Planning for a Humanoid Two-arm Robotic System. *Proceedings of TAMPRA: Combining Task and Motion Planning for Real-World Applications (ICAPS workshop)*, pages 13–20, 2012.

[19] Henry Kautz, David McAllester, and Bart Selman. En-

coding plans in propositional logic. *KR*, 96:374–384, 1996.

[20] Henry A Kautz, Bart Selman, et al. Planning as satisfiability. In *ECAI*, volume 92, pages 359–363, 1992.

[21] L. E. Kavraki, P. Švestka, J.-C. Latombe, and M.H. Overmars. Probabilistic roadmaps for path planning in high-dimensional configuration spaces. *IEEE Transactions on Robotics and Automation*, 12(4):566–580, Aug 1996.

[22] George Konidaris, Leslie Pack Kaelbling, and Tomas Lozano-Pérez. Constructing Symbolic Representations for High-Level Planning. *AAAI*, pages 1932–1940.

[23] James J Kuffner and Steven M LaValle. Rrt-connect: An efficient approach to single-query path planning. In *Int. Conf. on Robotics and Automation*, volume 2, pages 995–1001. IEEE, 2000.

[24] Steven M. Lavalle. Rapidly-exploring random trees: A new tool for path planning. Technical Report 98-11, Department of Computer Science, Iowa State University, Ames, IA, 1998.

[25] Steven M. LaValle and James J. Kuffner. Randomized kinodynamic planning. *International Journal of Robotics and Research*, 20(5):378–400, 2001.

[26] Tomás Lozano-Pérez and Leslie Pack Kaelbling. A constraint-based method for solving sequential manipulation planning problems. In *Int. Conf. on Intelligent Robots and Systems*, pages 3684–3691, 2014.

[27] Srinivas Nedunuri, Sailesh Prabhu, Mark Moll, Swarat Chaudhuri, and Lydia E Kavraki. SMT-Based Synthesis of Integrated Task and Motion Plans from Plan Outlines. *International Conference on Robotics & Automation (ICRA)*, pages 655–662, 2014.

[28] Jussi Rintanen, Keijo Heljanko, and Ilkka Niemelä. Planning as satisfiability: parallel plans and algorithms for plan search. *Artificial Intelligence*, 170(12):1031–1080, 2006.

[29] Siddharth Srivastava, Lorenzo Riano, Stuart Russell, and Pieter Abbeel. Using Classical Planners for Tasks with Continuous Operators in Robotics. *ICAPS Workshop on Planning and Robotics*, 2013.

[30] Siddharth Srivastava, Eugene Fang, Lorenzo Riano, Rohan Chitnis, Stuart Russell, and Pieter Abbeel. Combined Task and Motion Planning Through an Extensible Planner-Independent Interface Layer. *Proceedings - IEEE International Conference on Robotics and Automation*, (M):0–7, 2014.

[31] Ioan Alexandru Şucan and Lydia E. Kavraki. A sampling-based tree planner for systems with complex dynamics. *IEEE Transactions on Robotics*, 28(1):116–131, 2012.

[32] Jason Wolfe, B. Marthi, and Stuart Russell. Combined task and motion planning for mobile manipulation. *ICAPS*, pages 254–258, 2010.