E-Graphs: Bootstrapping Planning with Experience Graphs*

Mike Phillips
mlphilli@andrew.cmu.edu
Carnegie Mellon University

Benjamin Cohen
bcohen@seas.upenn.edu
University of Pennsylvania

Sachin Chitta
sachinc@willowgarage.com
Willow Garage

Maxim Likhachev
maxim@cs.cmu.edu
Carnegie Mellon University

Abstract

In this paper, we develop an online motion planning approach which learns from its planning episodes (experiences) a graph, an *Experience Graph*. On the theoretical side, we show that planning with Experience graphs is complete and provides bounds on suboptimality with respect to the graph that represents the original planning problem. Experimentally, we show in simulations and on a physical robot that our approach is particularly suitable for higher-dimensional motion planning tasks such as planning for two armed mobile manipulation.

Many mundane manipulation tasks such as picking and placing various objects in a kitchen are highly repetitive. It is expected that robots should be capable of learning and improving their performance with every execution of these repetitive tasks. This work focuses on learning from experience for motion planning. Our approach relies on a graph-search method for planning that builds an *Experience Graph* online to represent the high-level connectivity of the free space used for the encountered planning tasks. The planner uses the Experience graph to accelerate its planning whenever possible and gracefully degenerates to planning from scratch if no previous planning experiences can be reused. Planning with Experience graphs is complete and it provides bounds on suboptimality with respect to the graph that represents the original planning problem. Related work in (Jiang and Kallmann 2007) takes a database of motion plans and uses an RRT to draw the search towards a similar path to the new query. Our approach may use parts of many prior paths (not just one) and provides bounds on solution quality, unlike the above work. We provide results showing Experience Graphs can significantly improve the performance of a high-dimensional full-body planner for the PR2 robot. For more details refer to the full paper (Phillips et al. 2012).

Algorithm

An *Experience Graph* or E-Graph is a graph formed from the solutions found by the planner for previous planning queries. We will abbreviate this graph as $G^{\mathcal{E}}$. The graph $G^{\mathcal{E}}$ is incomparably smaller than graph G used to represent the original planning problem. At the same time, it is representative of the connectivity of the space exercised by the previously found motions.

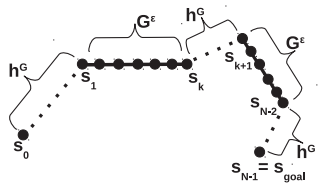


Figure 1: A visualization of $h^{\mathcal{E}}$. Solid lines are composed of edges from $G^{\mathcal{E}}$, while dashed lines are distances according to h^G. Note that h^G segments are always only two points long, while $G^{\mathcal{E}}$ segments can be an arbitrary number of points.

Our algorithm's speed-up comes from being able to reuse parts of old paths and avoid searching large portions of graph G. To accomplish this we introduce a heuristic which intelligently guides the search toward $G^{\mathcal{E}}$ when it looks like following parts of old paths will help the search get close to the goal. We define a new heuristic $h^{\mathcal{E}}$ in terms of the given heuristic h^G and edges in $G^{\mathcal{E}}$ ($c^{\mathcal{E}}$ is the cost function in $G^{\mathcal{E}}$).

$$h^{\mathcal{E}}(s_0) = \min_{\pi} \sum_{i=0}^{N-1} min\{\varepsilon^{\mathcal{E}} h^G(s_i, s_{i+1}), c^{\mathcal{E}}(s_i, s_{i+1})\}$$

where π is a path $\langle s_0 \dots s_{N-1} \rangle$ and $s_{N-1} = s_{goal}$ and $\varepsilon^{\mathcal{E}}$ is a scalar ≥ 1.

This equation is a minimization over a sequence of segments such that each segment is a pair of arbitrary states $s_a, s_b \in G$ and the cost of this segment is given by the minimum of two things: either $\varepsilon^{\mathcal{E}} h^G(s_a, s_b)$, the original heuristic inflated by $\varepsilon^{\mathcal{E}}$, or the cost of an actual least-cost path in $G^{\mathcal{E}}$, provided that $s_a, s_b \in G^{\mathcal{E}}$.

In Figure 1, the path π that minimizes the heuristic contains a sequence of alternating segments. In reality π can alternate between h^G and $G^{\mathcal{E}}$ segments as many or as few times as needed to produce the minimal π. When there is a $G^{\mathcal{E}}$ segment we can have many states s_i on the segment to connect two states, while when on a h^G segment a single pair of points suffices since no real edge between two states is needed for h^G to be defined.

The larger $\varepsilon^{\mathcal{E}}$ is, the more we avoid exploring G and focus on traveling on paths in $G^{\mathcal{E}}$. Figure 2 demonstrates how this works. As $\varepsilon^{\mathcal{E}}$ increases, it becomes more expensive to travel

*This is a short version of a paper that will appear at RSS'12

(a) $\varepsilon^{\mathcal{E}} = 1$ (b) $\varepsilon^{\mathcal{E}} = 2$ (c) $\varepsilon^{\mathcal{E}} \to \infty$

Figure 2: Shortest π according to $h^{\mathcal{E}}$ as $\varepsilon^{\mathcal{E}}$ changes. The dark solid lines are paths in $G^{\mathcal{E}}$ while the dark dashed lines are the heuristic's path π. Note as $\varepsilon^{\mathcal{E}}$ increases, the heuristic prefers to travel on $G^{\mathcal{E}}$. The light gray circles and lines show the graph G and the filled in gray circles represent the expanded states under the guidance of the heuristic.

off of $G^{\mathcal{E}}$ causing the heuristic to guide the search along parts of $G^{\mathcal{E}}$. In Figure 2a, the heuristic ignores the graph $G^{\mathcal{E}}$ because without inflating h^G at all, following edges in $G^{\mathcal{E}}$ will never be the cheaper option. On the other hand as $\varepsilon^{\mathcal{E}}$ increases, the heuristic uses more and more of $G^{\mathcal{E}}$ (Figure 2b, 2c). This figure also shows how during the search, by following old paths, we can avoid obstacles and have far fewer expansions. The expanded states are shown as filled in gray circles, which change based on how the $h^{\mathcal{E}}$ is biased by $\varepsilon^{\mathcal{E}}$.

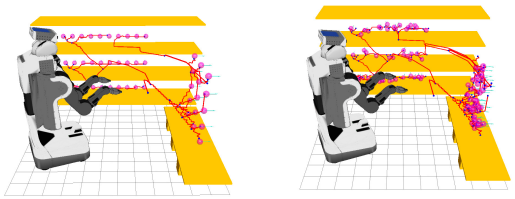
We run weighted A* without re-expansions (Likhachev, Gordon, and Thrun 2003). In addition to using the edges in G, we add two additional types of successors: *shortcuts* and *snap motions*. Also, instead of using the heuristic h^G, we use our new heuristic $h^{\mathcal{E}}$, inflated by $\varepsilon^w \geq 1$. Shortcut successors are generated when expanding a state $s \in G^{\mathcal{E}}$. They use $G^{\mathcal{E}}$ to jump to a state that is closest to s_{goal} according to the heuristic. The shortcuts allow the planner to quickly get near the goal without having to re-generate paths in $G^{\mathcal{E}}$. Finally, for environments that can support it, we introduce *snap motions*, which are dynamically generated successors for states that are very close to $G^{\mathcal{E}}$. These motions attempt to connect these close states directly to a nearby state on $G^{\mathcal{E}}$ so a shortcut on it may be used.

Since no edges are removed from the graph (we only add) and we are searching the graph with Weighted A* (a complete planner), if a solution exists on the original graph, our algorithm will find it. Furthermore, our planner provides a bound on the suboptimality of the solution cost with respect to the original graph.

Theorem 1. *For a finite graph G, the planner terminates, and the solution it returns is guaranteed to be no worse than $\varepsilon^w \cdot \varepsilon^{\mathcal{E}}$ times the optimal solution cost in graph G.*

Experimental Results

One of our experiments is modeled on an industrial warehouse where the robot must move upright objects from a pallet to a shelving unit. The 10 dimensional planning problem includes moving the base and the two arms which are rigidly attached to the object. Since our planner improves in performance with repeated attempts in a particular spatial region, we first bootstrap it with 45 uniformly distributed goals (split between the pallet and shelves). The bootstrap goals and the resultant $G^{\mathcal{E}}$ after processing them are shown in Figure 3a. We use a 3D Dijkstra search to implement the $h^{\mathcal{E}}$ for a sphere inscribed in the carried object starting from its goal position. In all experiments, $\varepsilon^w = 2$ and $\varepsilon^{\mathcal{E}} = 10$ resulting in a suboptimality bound of 20. The results were compared against regular Weighted A* with $\varepsilon^w = 20$ so both approaches have the same bound.



(a) Bootstrap goals (b) One of the test sets

Figure 3: Full-body planning in a warehouse

Table 1: Weighted A* to E-Graph Ratios on Warehouse Environment (100 goals per set)

Set	mean time(s)	std dev time(s)	mean expands	mean cost
1	18.40	39.94	206	0.63
2	17.74	51.24	231	0.60
3	224.83	1674.12	1562	0.63
4	24.30	155.42	142	0.64
5	16.69	72.41	193	0.62

Table 2: 10% Most difficult Cases on Warehouse Environment (10 goals per set)

Set	1	2	3	4	5
Mean Time Ratio	122.68	128.12	2157.14	200.80	112.17

A set of 100 random goals (with varying positions and yaws of the object) alternating between the pallet and shelves were then specified to the planner. This process was repeated 5 times with $G^{\mathcal{E}}$ cleared before running each set of 100 goals. Table 1 compares the two approaches by examining the ratio between planning times for Weighted A* and the E-Graph approach. The data indicates that planning with E-Graphs is almost 20 times faster. On average, 94% of the edges on a path produced by our planner were recycled from $G^{\mathcal{E}}$. Table 2 compares the 10% hardest goals (the ones that took the Weighted A* the longest to solve) showing that our approach is over 100 times faster on such difficult goals.

We thank Willow Garage for their support of this work. This research was also sponsored by ARL, under the Robotics CTA program grant W911NF-10-2-0016.

References

Jiang, X., and Kallmann, M. 2007. Learning humanoid reaching tasks in dynamic environments. In *IEEE International Conference on Intelligent Robots and Systems*.

Likhachev, M.; Gordon, G.; and Thrun, S. 2003. ARA*: Anytime A* with provable bounds on sub-optimality. In *Advances in Neural Information Processing Systems (NIPS)*.

Phillips, M.; Cohen, B.; Chitta, S.; and Likhachev, M. 2012. E-graphs: Bootstrapping planning with experience graphs. In *Proceedings of the Robotics: Science and Systems (RSS)*.