E-GRAPHS: BOOTSTRAPPING PLANNING WITH EXPERIENCE GRAPH

Phillips M., Cohen B., Chitta S., and Likhachev M.

Seminar in Robotics - 236824

Nitzan Madar

29/4/2020

Table of Content

1 Summary

Many motion planning problems contain highly repetitive tasks, for example, pick and place tasks. It is expected that robots should be capable of learning and improving their performance with every execution of these repetitive tasks.

Most of these problems are represented by graph search problems. In this paper, the author shows that learning from experience is possible by building an *Experience Graph* (E-Graph) online and learn which edges are known from the older paths. This method is complete and provides a bounded suboptimal solution but accelerates the runtime significantly.

The algorithm forms an E-Graph (contained in the original graph, $G^\varepsilon \in G$) from previous solutions, and uses a heuristic that intelligently guides the search toward G^ε when it looks like following parts of old paths (see figure 1), and that will help the search get close to the goal:

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

Where π is a path $\langle s_0, \dots, s_{N-1}\rangle$, h^G is given heuristic, c^ε is the cost function in G^ε and ϵ^ε is a scalar ≥ 1 (which controls the use of G^ε, see figure 2). This equation is a minimization over a sequence of pairs segments, between the original heuristic inflated by ϵ^ε and the cost of this pair in G^ε.

What is the ONE key insight that enabled the work of the paper?

The main insight that enabled the work of the paper is that when solving a complex large graph-based problem with repetitive tasks, and experience (knowledge from old paths planning) can be used to simplify the search over the whole graph. In multi-agent context, when planning in large warehouses for a long time (lifelong MAPD problems for example) the warehouses have constant structure and the old paths segments can be used for planning new paths (each path of each agent can be used when planning next tasks solution).

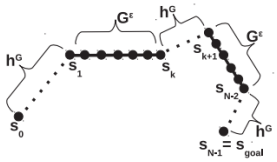

Figure 1: A visualization of h^ε. Solid lines are composed of edges from G^ε, while dashed lines are distances according to h^G. Note that h^G segments are always only two points long, while G^ε segments can be an arbitrary number of points

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

Figure 2: Shortest π according to h^{ϵ} as ϵ^{ε} changes. The dark solid lines are paths in G^{ε} while the dark dashed lines are the heuristic's path π. Note as ϵ^{ε} increases, the heuristic prefers to travel on G^{ε}. 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.

2 Future Work

Improvements under the paper context / what is missing

- **Pruning G^{ϵ} as it gets large**: edges are added to E-Graphs but in some cases, pruning edges can be done according to score function of the edges to improve the runtime or total cost.
- **Use it in anytime search:** combine this method on lifelong problems with task assigner.
- **Support user-defined E-Graphs:** for some reasons, sometimes the user wants to guide the robot to use specific path segments. Using user-provided E-Graph and chosen ϵ^{ε} can sometime have a better solution (cost or runtime). Also, this E-Graph will be constant and will not be too large.
- **Updating the E-Graph if G is changing:** sometimes the graph changes (for example, obstacle added or removed). Therefore, the E-Graph needs to be updated - some paths are forbidden if an obstacle is added or some can be improved if any obstacle is removed.
- **Cost function:** details and more example of cost functions are missing, which is an important parameter and affect the success rate of the algorithm.
- **Reduce the suboptimality:** if the problem lets the user more runtime (or cost) than we can use different parameters to reduce the cost (or runtime) and take it closer to optimal cost (or reduced runtime).

Extension: use it in MAPF problems

In MAPF problems, the map (the G graph) did not change, and it possible to build the E-Graph online when finding paths for each agent. Furthermore, in MAPD the tasks list is added, and it can be formalized as a lifelong problem. Hence, the E-Graph can be used and improve the runtime (under the assumption that bounded suboptimal solution is enough).

In addition, there is an option to combine highways methods on MAPF [2] [3] and E-graph: convert the highways to E-graph, and update that graph from the previous path founded and score given to each path.

3 Project Design [Bonus]

Required tools

This project requires understanding the MAPF problem, the CBS and ECBS algorithm, and applying the highway method these algorithms (ECBS+HWY and iECBS). I will assume that we can use an existing iECBS implementation (or ECBS+HWY), which already supports highways.

Main idea

Input: binary map (obstacle = 1, free coordinate = 0), start and goals agent's position list and initial highways graph.

Tasks:

1. **E-Graph Edges' score:** Build a function that takes as an input the low-level path founded (for each high-level node), the map, and the highway graph and return a data structure that contains the costs $c^{\epsilon}(s_i, s_{i+1})$ for each edge. This cost needs to evaluate how good this edge as a highway (for example, count how many time we use an edge and compute a transformation that normalized the most populate edge to 1 and the other to bigger cost)
2. **Heuristic:** Build a heuristic function that implements the equation in section 1, and a function that calculates a single agent path using the heuristic function.
3. **Dynamic E-Graph:** Build a function that removes and add highways to the E-graph according to thresholds or some other logic you think fit the idea (that make the E-Graph dynamic)
4. **Analysis:** Compare the optimal cost to the results for different ϵ^{ε}, and build ϵ^{ε} versus cost solution to show the optimal ϵ^{ε} for the specific problem. It might be interesting if the function is convex, or if it has more than one minimum value. In addition to that, compare the traditional highways methods to the dynamic E-Graph method you implemented.

Outcomes

As a result of this project implementation, we will get a solution that combines a dynamic E-Graph theory into the MAPF problem.

Then, comparing two methods of using highways on the MAPF problem is possible – one method is a traditional highways method [2] [3], and the other is use the dynamic E-Graph.

In addition to that, we can understand how to choose the E-Graph parameter effect on the solution, comparing to change the edges' weights in the parallel algorithms.

Expected results

Comparing to ECBS and CBS, which the runtime grew as the number collision, the highways methods can have better results as shown in [2] and [3], especially in maps with narrow corridors. That because marking a direction can let the algorithm avoiding collision and then searching over

the constraints tree is smaller. According to the same principle, we can assume that using dynamic highways (dynamic E-graph) can also improve the runtime, for correct parameter choosing.

4 References

[1] Michael Phillips, Benjamin J. Cohen, Sachin Chitta, Maxim Likhachev: E-Graphs: Bootstrapping Planning with Experience Graphs. Robotics: Science and Systems 2012

[2] Liron Cohen, Tansel Uras, Sven Koenig: Feasibility Study: Using Highways for Bounded-Suboptimal Multi-Agent Path Finding. SOCS 2015: 2-8

[3] Liron Cohen, *Tansel Uras, T. K. Satish Kumar, Hong Xu, Nora Ayanian, Sven Koenig:* Improved Solvers for Bounded-Suboptimal Multi-Agent Path Finding. *IJCAI 2016: 3067-3074*