HIGH-LEVEL PLANNING WITH EXPERIENCE GRAPHS

Aram Ebtekar, Maxim Likhachev

Motivation

- **AI "dream"**: oracle that solves all problems!
- Is this **reasonable**? Are **humans** oracles?
- Recent trend across AI/robotics:
  - **Learning** from **experience**, solution **reuse**
  - Next time a similar problem appears, find **better** solutions **faster**
  - Requires knowledge representation

Abstract Representation

- A **plan** is a path from start to goal
- Graph is implicitly represented in **STRIPS** form:
  - A = set of **atoms**: ON A B, ONTABLE B, HOLDING C
  - 2^A = set of nodes or **states**
  - $Op = (P, A, D)$ is an **action** with sets of atoms designated as **preconditions**, **add** and **delete effects**, corresponding to edges: STACK A B, PICKUP C
  - **Transition:** $\big(S, (P, A, D)\big) \rightarrow S \cup A \setminus D$ if $S \supseteq P$
  - We're at a **goal** iff $S \supseteq G$

Heuristic Search Planner

- Admissible relaxation: ignore delete lists
- Estimate cost to achieve **individual** atoms
  - $g_S(a) = 1 + \min_{Op} g_S(P)$ where Op **adds** a
  - For **sets** of atoms, use $g_S(P) = \max_{p \in P} g_S(p)$
- Heuristic estimate to goal: $h(S) = g_S(G)$
- Do forward weighted A*
  - When **generating** S, need to **compute** $g_S(a)$
  - Use **dynamic programming**

Experience Graphs

- Originally developed for **explicit** graphs by **SBPL**
- Store edges from previously generated paths
- **Inflate** non E-graph edges by ϵ^E to bias search
  - $h^E(S) = \min_{\pi} \sum_i \min\{\epsilon^E h(s_i, s_{i+1}),\ c^E(s_i, s_{i+1})\}$ over all **paths** $\pi = \langle S = s_0, s_1, s_2, \dots, s_N = G \rangle$
  - ϵh^E is $\epsilon\epsilon^E$**-consistent** provided h is consistent
- But how can we **compute** h^E in **STRIPS**?
  - Answer: reverse **Dijkstra** from G on the E-graph!

STRIPS E-Graphs

- **Preprocessing** phase:
  - Let N^E = all E-graph nodes, plus **minimal** goal state G
  - Run DP to compute $g_C(a)$ for every state $C \in N^E$
  - Now we have pairwise distance estimates $g_C(D)$
  - Reverse **Dijkstra** from G with E-graph and $\epsilon^E g$ edges
- When **generating** $S \notin N^E$:
  - $h^E(S) = \min_{\pi} \sum_i \min\{\epsilon^E h(s_i, s_{i+1}),\ c^E(s_i, s_{i+1})\}$
  - **Computable** by $h^E(S) = \min_{C \in N^E} (\epsilon^E g_S(C) + h^E(C))$

Analysis

- $h^E(S) = \min_{C \in N^E}(\epsilon^E g_S(C) + h^E(C))$
- ϵh^E is $\epsilon\epsilon^E$-**consistent**, so solution is $\epsilon\epsilon^E$-**optimal**
- For large ϵ^E, after **generating** a node with a direct E-path to goal, **only** E-graph nodes are **expanded**
- Experimental analysis… coming soon!

Extensions for Future Work

- **Generalize** E-graph actions by **projections**
  - Can “partially inflate” non E-graph edges according to some similarity measure against E-graph edges
  - To a limited extent, h already acts as such a measure
- What to do when E-graph gets **big**?
  - “Forget” edges which have not helped recently
- Combine with other planning methods
  - Anytime incremental planning with variable-cost actions
  - Less straightforward: GRT, abstraction heuristics, etc.

References

- *Mike Phillips, Benjamin Cohen, Sachin Chitta and Maxim Likhachev,* ***"E-Graphs: Bootstrapping Planning with Experience Graphs,"*** *Proceedings of the Robotics: Science and Systems Conference (RSS), 2012.*
- *Blai Bonet and Héctor Geffner,* ***"Planning as heuristic search,"*** *Artificial Intelligence 129.1 (2001): 5-33.*
- Our in-progress code based on the previous paper: https://github.com/EbTech/HSP2