E-Graphs : Bootstrapping Planning with Experience Graphs

Mike Phillips, Benjamin Cohen, Sachin Chitta, Maxim Likhachev
What is an Experience Graph

- Online motion planning approach.
- Robot learns from its experiences.
- E-Graph represents the underlying connectivity of the space.
- Significant speed-up over planning from scratch.
Why using learning?

- Mundane manipulation works which are highly repetative.
- Most part of the environment stays same everyday.
- Learn from previous experiences.
- Approach is most useful when tasks are somewhat repeatable, e.g. In moving a set of dishes off a particular counter into a dishwasher.
Fig. 1. Motion planning is often used to compute motions for repetitive tasks such as dual-arm mobile manipulation in a kitchen.
Definitions

- $G(V^G, E^G)$ is a graph modeling the original motion planning problem, where $V^G$ is the set of vertices and $E^G$ is the set of edges connecting pairs of vertices in $V^G$.
- $G^E(V^E, E^E)$ is the E-Graph that our algorithm builds over time ($G^E \subseteq G$).
- $c(u, v)$ is the cost of the edge from vertex $u$ to vertex $v$.
- $c^E(u, v)$ is the cost of the edge from vertex $u$ to vertex $v$ in graph $G^E$. 
Heuristic Search

\[ h^G(s, s_{goal}) \leq c(s, s') + h^G(s', s_{goal}). \]

- Admissible: Never over-estimates the min cost from any state to the goal.
- Consistent: Satisfies the above triangle equality.
Algorithms

\[
\text{findPath}(s_{\text{start}}, s_{\text{goal}})
\]

1: \text{updateEGraph}(s_{\text{goal}})
2: \pi = \text{computePath}(s_{\text{start}}, s_{\text{goal}})
3: G^\mathcal{E} = G^\mathcal{E} \cup \pi

\[
\text{updateEGraph}(s_{\text{goal}})
\]

1: \text{updateChangedCosts}()
2: disable edges that are now invalid
3: re-enable disabled edges that are now valid
4: \text{precomputeShortcuts}()
5: compute heuristic \(h^\mathcal{E}\) according to Equation 1
Modified Heuristic

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

where \(\pi\) is a path \(\langle s_0 \ldots s_{N-1} \rangle\) and \(s_{N-1} = s_{goal}\) and \(\varepsilon^\varepsilon\) is a scalar \(\geq 1\).
Contd..

Fig. 2. A visualization of Equation 1. Solid lines are composed of edges from $G^E$, while dashed lines are distances according to $h^G$. Note that $h^G$ segments are always only two points long, while $G^E$ segments can be an arbitrary number of points.

Fig. 3. Shortest $\pi$ according to $h^E$ as $\varepsilon^E$ changes. The dark solid lines are paths in $G^E$ while the dark dashed lines are the heuristic’s path $\pi$. Note as $\varepsilon^E$ increases, the heuristic prefers to travel on $G^E$. 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.
**computePath Algorithm**

- If the scalar is larger, experience graph is used more.
- *computePath* runs weighted A* without re-expansions.
- Weighted A* uses a parameter $e > 1$ to inflate the heuristic used by A*.
- Solution is e-optimal.
- Runs dramatically faster than A*.
Pseudo-code of `computePath`

```
computePath(s_{start}, s_{goal})
1: OPEN = ∅
2: CLOSED = ∅
3: g(s_{start}) = 0
4: f(s_{start}) = ε^w_h^ε(s_{start})
5: insert s_{start} into OPEN with f(s_{start})
6: while s_{goal} is not expanded do
7:   remove s with the smallest f-value from OPEN
8:   insert s in CLOSED
9:   S = getSuccessors(s) ∪ shortcuts(s) ∪ snap(s)
10:  for all s' ∈ S do
11:    if s' was not visited before then
12:      f(s') = g(s') = ∞
13:    end if
14:    if g(s') > g(s) + c(s, s') and s' ∉ CLOSED then
15:      g(s') = g(s) + c(s, s')
16:      f(s') = g(s') + ε^w_h^ε(s')
17:      insert s' into OPEN with f(s')
18:    end if
19:  end for
20:  end while
```
Theoretical Analysis

**Theorem 1.** For a finite graph $G$, our planner terminates and finds a path in $G$ that connects the $s_{\text{start}}$ and $s_{\text{goal}}$ if one exists.

**Lemma 1.** If the original heuristic function $h^G$ is admissible, then the heuristic function $h^\varepsilon$ is $\varepsilon$-consistent.

From Equation [1]

$$h^\varepsilon(s) \leq \min \{ \varepsilon^\varepsilon h^G(s, s'), c^\varepsilon(s, s') \} + h^\varepsilon(s')$$

$$h^\varepsilon(s) \leq \varepsilon^\varepsilon h^G(s, s') + h^\varepsilon(s')$$

$$h^\varepsilon(s) \leq \varepsilon c(s, s') + h^\varepsilon(s')$$

The last step follows from $h^G$ being admissible. Therefore, $h^\varepsilon$ is $\varepsilon^\varepsilon$-consistent.
Theorem 2. For a finite graph $G$, the planner terminates, and the solution it returns is guaranteed to be no worse than $\varepsilon^w \cdot \varepsilon^E$ times the optimal solution cost in graph $G$.

Consider $h'(s) = \frac{h^E(s)}{\varepsilon^E}$. $h'(s)$ is clearly consistent. Then, $\varepsilon^w h^E(s) = \varepsilon^w \cdot \varepsilon^E h'(s)$. The proof that $\varepsilon^w \cdot \varepsilon^E h'(s)$ leads to Weighted A* (without re-expansions) returning paths bounded by $\varepsilon^w \cdot \varepsilon^E$ times the optimal solution cost follows from [10].
Implementation Details

- **Heuristic** can be computed in $O(1)$ time upon request.

- **Shortcuts** are pre-computed edges that connect all states in $G$ to a very small set of states in $G$.

- Pre-computations are done to determine the costs of paths between a pair of states in $G$. 
Experimental Results: Warehouse

**E-Graphs on Warehouse Environment (100 goals per set)**

<table>
<thead>
<tr>
<th>Set</th>
<th>mean time(s)</th>
<th>std dev time(s)</th>
<th>mean expands</th>
<th>mean cost</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1.08</td>
<td>1.77</td>
<td>103</td>
<td>7933</td>
</tr>
<tr>
<td>2</td>
<td>1.26</td>
<td>3.18</td>
<td>150</td>
<td>7806</td>
</tr>
<tr>
<td>3</td>
<td>1.53</td>
<td>4.58</td>
<td>178</td>
<td>7804</td>
</tr>
<tr>
<td>4</td>
<td>1.80</td>
<td>4.70</td>
<td>221</td>
<td>7775</td>
</tr>
<tr>
<td>5</td>
<td>1.16</td>
<td>1.97</td>
<td>142</td>
<td>7351</td>
</tr>
</tbody>
</table>

**TABLE II**

**Weighted A* on Warehouse Environment (100 goals per set)**

<table>
<thead>
<tr>
<th>Set</th>
<th>mean time(s)</th>
<th>std dev time(s)</th>
<th>mean expands</th>
<th>mean cost</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>12.12</td>
<td>35.11</td>
<td>1883</td>
<td>4589</td>
</tr>
<tr>
<td>2</td>
<td>8.22</td>
<td>23.14</td>
<td>1211</td>
<td>4321</td>
</tr>
<tr>
<td>3</td>
<td>134.12</td>
<td>806.11</td>
<td>21792</td>
<td>4590</td>
</tr>
<tr>
<td>4</td>
<td>14.25</td>
<td>70.21</td>
<td>2527</td>
<td>4539</td>
</tr>
<tr>
<td>5</td>
<td>9.59</td>
<td>37.67</td>
<td>1495</td>
<td>4221</td>
</tr>
</tbody>
</table>
### TABLE III
**Weighted A* to E-Graph Ratios on Warehouse Environment (100 goals per set)**

<table>
<thead>
<tr>
<th>Set</th>
<th>mean time(s)</th>
<th>std dev time(s)</th>
<th>mean expands</th>
<th>mean cost</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>18.40</td>
<td>39.94</td>
<td>206</td>
<td>0.63</td>
</tr>
<tr>
<td>2</td>
<td>17.74</td>
<td>51.24</td>
<td>231</td>
<td>0.60</td>
</tr>
<tr>
<td>3</td>
<td>224.83</td>
<td>1674.12</td>
<td>1562</td>
<td>0.63</td>
</tr>
<tr>
<td>4</td>
<td>24.30</td>
<td>155.42</td>
<td>142</td>
<td>0.64</td>
</tr>
<tr>
<td>5</td>
<td>16.69</td>
<td>72.41</td>
<td>193</td>
<td>0.62</td>
</tr>
</tbody>
</table>
Kitchen Scenario

(a) $G^E$ after bootstrap goals

(b) $G^E$ after bootstrap goals

(c) $G^E$ after test goals

(d) $G^E$ after test goals
Contd..

**TABLE V**  
**RESULTS ON KITCHEN ENVIRONMENT (40 GOALS)**

<table>
<thead>
<tr>
<th></th>
<th>mean time(s)</th>
<th>std dev time(s)</th>
<th>mean expands</th>
<th>mean cost</th>
</tr>
</thead>
<tbody>
<tr>
<td>E-Graphs (E)</td>
<td>2.12</td>
<td>6.55</td>
<td>351</td>
<td>5646</td>
</tr>
<tr>
<td>Weighted A* (W)</td>
<td>11.54</td>
<td>20.74</td>
<td>2639</td>
<td>4236</td>
</tr>
<tr>
<td>Ratio (W/E)</td>
<td>22.29</td>
<td>38.09</td>
<td>357</td>
<td>0.79</td>
</tr>
</tbody>
</table>
Fig. 6. Learning by demonstration in a more difficult warehouse scenario

(a) A demonstrated path  (b) $G^c$ after 12 goals

Fig. 7. Tabletop manipulation experiments

(a) Grasping pipeline setup  (b) $G^c$ partway through the experiment
Tabletop Manipulation

<table>
<thead>
<tr>
<th>Method</th>
<th>Mean Time (s)</th>
<th>Std Dev Time (s)</th>
<th>Mean Expands</th>
<th>Mean Cost</th>
</tr>
</thead>
<tbody>
<tr>
<td>E-Graphs (E)</td>
<td>0.13</td>
<td>0.07</td>
<td>4</td>
<td>117349</td>
</tr>
<tr>
<td>Weighted A* (W)</td>
<td>0.26</td>
<td>0.10</td>
<td>145</td>
<td>109297</td>
</tr>
<tr>
<td>SBL (S)</td>
<td>0.24</td>
<td>0.09</td>
<td>N/A</td>
<td>N/A</td>
</tr>
<tr>
<td>Ratio (W/E)</td>
<td>2.50</td>
<td>1.23</td>
<td>66</td>
<td>1.03</td>
</tr>
<tr>
<td>Ratio (S/E)</td>
<td>2.44</td>
<td>1.50</td>
<td>N/A</td>
<td>N/A</td>
</tr>
</tbody>
</table>
Conclusion

• Planning with Experience Graph: Online search algorithm.

• Reusing previously visited paths: Significant speedups over planning from scratch.

• Planner gets better over time because of learning.

• Guarantees completeness and e-optimality.
Thank you...

Questions?

Presentation By: Ayan Dutta