Sub-optimal Heuristic Search and its Application to Planning for Manipulation

Mike Phillips

Slides adapted from Maxim Likhachev
Planning for Mobile Manipulation

• What planning tasks are there?

robotic bartender demo at Search-based Planning Lab, April'12
Planning for Mobile Manipulation

- What planning tasks are there?

  - task planning (order of getting items)
  - planning for navigation
  - planning motions for an arm
  - planning coordinated base+arms (full body) motions
  - planning how to grasp

robotic bartender demo at Search-based Planning Lab, April'12
Planning for Mobile Manipulation

• What planning tasks are there?

  task planning (order of getting items)
  planning for navigation
  planning motions for an arm
  planning coordinated base+arms (full body) motions
  planning how to grasp

robotic bartender demo at Search-based Planning Lab, April'12
Planning for Mobile Manipulation

- **Example of planning for 20D arm in 2D:**
  - each state is defined by 20 discretized joint angles \( \{q_1, q_2, \ldots, q_{20}\} \)
  - each action is changing one angle (or set of angles) at a time
Planning for Mobile Manipulation

- **Example of planning for 20D arm in 2D:**
  - each state is defined by 20 discretized joint angles \(\{q_1, q_2, ..., q_{20}\}\)
  - each action is changing one angle (or set of angles) at a time

*How to search such a high-dimensional state-space?*
Typical Approaches to Motion Planning

sub-optimal heuristic searches can be fast enough for high-D motion planning and return consistent good quality solutions

planning with optimal heuristic search-based approaches (e.g., $A^*$, $D^*$)
+ completeness/optimality guarantees
+ excellent cost minimization
+ consistent solutions to similar motion queries
- slow and memory intensive
- used mostly for 2D (x,y) path planning

planning with sampling-based approaches (e.g., RRT, PRM,…)
+ typically very fast and low memory
+ used for high-D path/motion planning
- completeness in the limit
- often poor cost minimization
- hard to provide solution consistency
Planning for Mobile Manipulation

- **Example of planning for 20D arm in 2D:**
  - each state is defined by 20 discretized joint angles \( \{q_1, q_2, \ldots, q_{20}\} \)
  - each action is changing one angle (or set of angles) at a time
Sub-optimal Heuristic Search for High-D Motion Planning

• ARA* (Anytime version of A*) graph search
  – effective use of solutions to relaxed (easier) motion planning problems

• Experience graphs
  – heuristic search that learns from its planning experiences
Sub-optimal Heuristic Search for High-D Motion Planning

• ARA* (Anytime version of A*) graph search
  – effective use of solutions to relaxed (easier) motion planning problems

• Experience graphs
  – heuristic search that learns from its planning experiences
Heuristics in Heuristic Search

- A* Search: expands states in the order of $f(s) = g(s) + h(s)$ values
- In A*, when $s$ is expanded $g(s)$ is optimal
- $h(s)$ is a relaxed (simplified) version of the problem
Heuristics in Heuristic Search

• **A* Search:** expands states in the order of $f(s) = g(s) + h(s)$ values

• In A*, when $s$ is expanded $g(s)$ is optimal

• $h(s)$ is a relaxed (simplified) version of the problem

```
If $h(s)$ was a perfect estimate (let’s call it $h^*$), what does $f(s)$ represent upon expansion of $s$?
```
• **A* Search**: expands states in the order of $f(s) = g(s) + h(s)$ values

• In A*, when $s$ is expanded $g(s)$ is optimal

• $h(s)$ is a relaxed (simplified) version of the problem

If $h(s)$ was a perfect estimate (let’s call it $h^*$), what does $f(s)$ represent upon expansion of $s$?

Now what if $h(s)$ wasn’t perfect?
Heuristics in Heuristic Search

- **A* Search**: expands states in the order of $f = g + h$ values

  **ComputePath function**
  
  while($s_{goal}$ is not expanded)
  
  remove $s$ with the smallest $[f(s) = g(s) + h(s)]$ from OPEN;
  
  insert $s$ into CLOSED;
  
  for every successor $s'$ of $s$ such that $s'$ not in CLOSED
  
  if $g(s') > g(s) + c(s, s')$
  
  $g(s') = g(s) + c(s, s')$;
  
  insert $s'$ into OPEN;

  **expansion of $s$**

  
  \[ h(s) \rightarrow g(s) \]

  **the cost of a shortest path from $s_{start}$ to $s$ found so far**

  **an (under) estimate of the cost of a shortest path from $s$ to $s_{goal}$**
Heuristics in Heuristic Search

• Solutions to a relaxed version of the planning problem

**2D \((x,y)\) planning**

*Euclidean distance heuristic:*

2D planning without obstacles and no grid

**20D arm planning**

*2D end-effector distance heuristic:*

2D planning for end-effector with obstacles
Heuristics in Heuristic Search

- Dijkstra’s: expands states in the order of $f = g$ values
- A* Search: expands states in the order of $f = g + h$ values
- Weighted A*: expands states in the order of $f = g + \varepsilon h$ values, $\varepsilon > 1 = \text{bias towards states that are closer to goal}$
• Dijkstra’s: expands states in the order of $f = g$ values

What are the states expanded?
Heuristics in Heuristic Search

- A* Search: expands states in the order of $f = g + h$ values

What are the states expanded?
Heuristics in Heuristic Search

- A* Search: expands states in the order of $f = g + h$ values

for high-D problems, this results in A* being too slow and running out of memory
Heuristics in Heuristic Search

- Weighted A* Search: expands states in the order of $f = g + \varepsilon h$ values, $\varepsilon > 1$
- bias towards states that are closer to goal
Heuristics in Heuristic Search

• **Weighted A* Search:**
  – trades off optimality for speed
  – $\varepsilon$-suboptimal:
    \[
    \text{cost(solution)} \leq \varepsilon \cdot \text{cost(optimal solution)}
    \]
  – in many domains, it has been shown to be orders of magnitude faster than A*

$A^*$: $\varepsilon = 1.0$

- 20 expansions
- solution = 10 moves

Weighted $A^*$: $\varepsilon = 2.5$

- 13 expansions
- solution = 11 moves
Anytime Search based on weighted A*

- Constructing anytime search based on weighted A*:  
  - find the best path possible given some amount of time for planning  
  - do it by running a series of weighted A* searches with decreasing $\varepsilon$:

\[
\begin{align*}
\varepsilon &= 2.5 \\
&= 15 \text{ expansions} \\
&= 11 \text{ moves} \\
\varepsilon &= 1.5 \\
&= 20 \text{ expansions} \\
&= 11 \text{ moves} \\
\varepsilon &= 1.0 \\
&= 13 \text{ expansions} \\
&= 10 \text{ moves}
\end{align*}
\]
Anytime Search based on weighted A*

• Constructing anytime search based on weighted A*:  
  - find the best path possible given some amount of time for planning  
  - do it by running a series of weighted A* searches with decreasing $\epsilon$:  

  $\epsilon = 2.5$  
  13 expansions  
  solution = 11 moves  

  $\epsilon = 1.5$  
  15 expansions  
  solution = 11 moves  

  $\epsilon = 1.0$  
  20 expansions  
  solution = 10 moves

• Inefficient because  
  – many state values remain the same between search iterations  
  – we should be able to reuse the results of previous searches
Anytime Search based on weighted A*

- Constructing anytime search based on weighted A*:  
  - find the best path possible given some amount of time for planning  
  - do it by running a series of weighted A* searches with decreasing $\varepsilon$:

  $\varepsilon = 2.5$
  
  13 expansions
  solution = 11 moves

  $\varepsilon = 1.5$
  
  15 expansions
  solution = 11 moves

  $\varepsilon = 1.0$
  
  20 expansions
  solution = 10 moves

- **ARA***
  
  - an efficient version of the above that reuses state values within any search iteration
ARA*

• Efficient series of weighted A* searches with decreasing $\varepsilon$:

**ComputePathwithReuse function**

\[
\text{while}(f(s_{\text{goal}}) > \text{minimum f-value in OPEN}) \quad \text{remove s with the smallest } [g(s) + \varepsilon h(s)] \text{ from OPEN;}
\]

\[
\text{insert s into CLOSED;}
\]

\[
\text{for every successor } s' \text{ of } s
\]

\[
\text{if } g(s') > g(s) + c(s,s')
\]

\[
g(s') = g(s) + c(s,s');
\]

\[
\text{if } s' \text{not in CLOSED then insert } s' \text{ into OPEN;}
\]
ARA*

- Efficient series of weighted A* searches with decreasing $\varepsilon$:

```
ComputePathwithReuse function
while(f(s_{goal}) > minimum f-value in OPEN )
  remove s with the smallest [g(s) + $\varepsilon$h(s)] from OPEN;
  insert s into CLOSED;
for every successor s’ of s
  if g(s’) > g(s) + c(s,s’)
    g(s’) = g(s) + c(s,s’);
  if s’ not in CLOSED then insert s’ into OPEN;
  otherwise insert s’ into INCONS
```
ARA*

- Efficient series of weighted A* searches with decreasing $\varepsilon$:

**ComputePathwithReuse function**

while($f(s_{goal}) >$ minimum f-value in OPEN )

  remove s with the smallest $[g(s) + \varepsilon h(s)]$ from OPEN;
  insert s into CLOSED;

for every successor s’ of s

  if $g(s') > g(s) + c(s,s')$
    $g(s') = g(s) + c(s,s')$;
  if s’ not in CLOSED then insert s’ into OPEN;
  otherwise insert s’ into INCONS

set $\varepsilon$ to large value;
$g(s_{start}) = 0$; OPEN = $\{s_{start}\}$;
while $\varepsilon \geq 1$

  CLOSED = $\{}$; INCONS = $\{}$;
  ComputePathwithReuse();
  publish current $\varepsilon$ suboptimal solution;
  decrease $\varepsilon$;
  initialize OPEN = OPEN U INCONS;
ARA*

- A series of weighted A* searches
  - $\epsilon = 2.5$
    - 13 expansions
    - solution = 11 moves

- ARA*
  - $\epsilon = 2.5$
    - 13 expansions
    - solution = 11 moves
When using ARA*, the research is in finding a graph representation $G$ and a corresponding heuristic function $h$ that lead to shallow local minima for the search.
ARA*-based Planning for Manipulation

- ARA*-based motion planning for a single 7DOF arm [Cohen et al., ICRA’10, ICRA’11]
  - goal is given as 6D (x, y, z, r, p, y) pose of the end-effector
  - each state is: 4 non-wrist joint angles \{q_1, ..., q_4\} when far away from the goal, all 7 joint angles \{q_1, ..., q_7\} when close to the goal
  - actions are: {moving one joint at a time by fixed \(\Delta\); IK-based motion to snap to the goal when close to it}
ARA*-based Planning for Manipulation

- ARA*-based motion planning for a single 7DOF arm [Cohen et al., ICRA’10, ICRA’11]
  - goal is given as 6D \((x,y,z,r,p,y)\) pose of the end-effector
  - each state is: 4 non-wrist joint angles \(\{q1,...,q4\}\) when far away from the goal, all 7 joint angles \(\{q1,...,q7\}\) when close to the goal
  - actions are: \{moving one joint at a time by fixed \(\Delta\); IK-based motion to snap to the goal when close to it\}
  - heuristics are: 3D BFS distances for the end-effector

expansions (shown as dots corresponding to ef poses) based on 3D BFS heuristics (shown as curve)
expansions based on Euclidean distance heuristics (shown as curve)
ARA*-based Planning for Manipulation

- ARA*-based motion planning for a single 7DOF arm [Cohen et al., ICRA’10, ICRA’11]
  - goal is given as 6D \((x,y,z,r,p,y)\) pose of the end-effector
  - each state is: 4 non-wrist joint angles \(\{q_1,\ldots,q_4\}\) when far away from the goal, all 7 joint angles \(\{q_1,\ldots,q_7\}\) when close to the goal
  - actions are: \{moving one joint at a time by fixed \(\Delta\); IK-based motion to snap to the goal when close to it\}
  - heuristics are: 3D BFS distances for the end-effector

![Image](attachment:image.png)

expansions (shown as dots corresponding to ef poses) based on 3D BFS heuristics (shown as curve)

expansions based on Euclidean distance heuristics (shown as curve)
ARA*-based Planning for Manipulation

- ARA*-based motion planning for a single 7DOF arm [Cohen et al., ICRA’10, ICRA’11]
  - goal is given as 6D \((x, y, z, r, p, y)\) pose of the end-effector
  - each state is: 4 non-wrist joint angles \(\{q_1, \ldots, q_4\}\) when far away from the goal, all 7 joint angles \(\{q_1, \ldots, q_7\}\) when close to the goal
  - actions are: \{moving one joint at a time by fixed \(\Delta\); IK-based motion to snap to the goal when close to it\}
  - **heuristics are:** 3D BFS distances for the end-effector

expansions (shown as dots corresponding to ef poses) based on 3D BFS heuristics (shown as curve)
expansions based on Euclidean distance heuristics (shown as curve)
ARA*-based Planning for Manipulation

• ARA*-based motion planning for a single 7DOF arm [Cohen et al., ICRA’10, ICRA’11]

Shown on PR2 robot but planner has been run on other arms (kuka)
ARA*-based Planning for Manipulation

- ARA*-based motion planning for dual-arm motion [Cohen et al., ICRA’12]
  - goal is given as 4D (x,y,z,y) pose for the object
  - each state is 6D: 4D (x,y,z,y) pose of the object, \{q1,q2\} of the elbow angles
  - actions are: \{changing the 4D pose of the object or q1, q2\}
  - heuristics are: 3D BFS distances for the object
ARA*-based Planning for Manipulation

- ARA*-based motion planning for dual-arm motion [Cohen et al., ICRA’12]

motions as generated by the planner - no smoothing
ARA*-based Planning for Manipulation

- ARA*-based motion planning for dual-arm motion [Cohen et al., ICRA’12]

*consistency of motions across similar start-goal trials*
ARA*-based Planning for Manipulation

Discussion

**Pros**

- Explicit cost minimization
- Consistent plans (similar input generates similar output)
- Can handle arbitrary goal sets (such as set of grasps from a grasp planner)
- Many path constraints are easy to implement (upright orientation of gripper)

**Cons**

- Requires good graph and heuristic design to plan fast
- Number of motions from a state affects speed and path quality
Sub-optimal Heuristic Search for High-D Motion Planning

- ARA* (Anytime version of A*) graph search
  - effective use of solutions to relaxed (easier) motion planning problems

- Experience graphs
  - heuristic search that learns from its planning experiences
Planning with Experience Graphs

• Many planning tasks are repetitive
  - loading a dishwasher
  - opening doors
  - moving objects around a warehouse
  - …

• Can we re-use prior experience to accelerate planning, in the context of search-based planning?

• Would be especially useful for high-dimensional problems such as mobile manipulation!
Planning with Experience Graphs

Given a set of previous paths (experiences)…
Planning with Experience Graphs

Put them together into an $E$-graph (Experience graph)
Planning with Experience Graphs

• *E-Graph* [Phillips et al., RSS’12]:
  – *Collection of previously computed paths or demonstrations*
  – *A sub-graph of the original graph*
Planning with Experience Graphs

Given a new planning query…
Planning with Experience Graphs

…re-use E-graph. For repetitive tasks, planning becomes much faster
Planning with Experience Graphs

...re-use E-graph. For repetitive tasks, planning becomes much faster

Theorem 1: Algorithm is complete with respect to the original graph

Theorem 2: The cost of the solution is within a given bound on sub-optimality
Planning with Experience Graphs

- Reuse $E$-Graph by:
  - Introducing a new heuristic function
  - Heuristic guides the search toward expanding states on the $E$-Graph within sub-optimality bound $\varepsilon$
Planning with Experience Graphs

- Focusing search towards E-graph **within sub-optimality bound** $\varepsilon$

Heuristic computation finds a min cost path using two kinds of “edges”

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

Traveling off the E-Graph uses an inflated original heuristic

Traveling on E-Graph uses actual costs

Heuristic computation finds a min cost path using two kinds of “edges”

Traveling off the E-Graph uses an inflated original heuristic

Traveling on E-Graph uses actual costs
Planning with Experience Graphs

- Focusing search towards E-graph within sub-optimality bound $\varepsilon$

Heuristic computation finds a min cost path using two kinds of “edges”

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

Traveling off the E-Graph uses an inflated original heuristic

Traveling on E-Graph uses actual costs
Planning with Experience Graphs

- Focusing search towards E-graph within sub-optimality bound $\varepsilon$

Heuristic computation finds a min cost path using two kinds of “edges”

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

- Focusing search towards E-graph within sub-optimality bound $\varepsilon$

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

Theorem 5. Completeness w.r.t. the original graph G: Planning with E-graphs is guaranteed to find a solution, if one exists in G.

Theorem 6. Bounds on sub-optimality: The cost of the solution found by planning with E-graphs is guaranteed to be at most $\varepsilon^E$-suboptimal:

$$\text{cost(solution)} \leq \varepsilon^E \text{cost(optimal solution in G)}$$
Theorem 5. Completeness w.r.t. the original graph $G$: Planning with E-graphs is guaranteed to find a solution, if one exists in $G$.

Theorem 6. Bounds on sub-optimality: The cost of the solution found by planning with E-graphs is guaranteed to be at most $\varepsilon$-suboptimal:

$$\text{cost}(\text{solution}) \leq \varepsilon \cdot \text{cost}(\text{optimal solution in } G)$$

These properties hold even if quality or placement of the experiences is arbitrarily bad.
Planning with $E$-Graphs for Mobile Manipulation

- Dual-arm mobile manipulation (10 DoF)
Planning with $E$-Graphs for Mobile Manipulation

- Dual-arm mobile manipulation (10 DoF)
  - 3 for base navigation
  - 1 for spine
  - 6 for arms (upright object constraint, roll and pitch are 0)

- Experiments in multiple environments
Planning with $E$-Graphs for Mobile Manipulation

Kitchen environment:
- moving objects around a kitchen
- bootstrap $E$-Graph with 10 representative goals
- tested on 40 goals in natural kitchen locations
Planning with $E$-Graphs for Mobile Manipulation

Kitchen environment – Planning times

<table>
<thead>
<tr>
<th>Method</th>
<th>Success (of 40)</th>
<th>Mean Time (s)</th>
<th>Std. Dev. (s)</th>
<th>Max (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>E-Graphs</td>
<td>40</td>
<td>0.33</td>
<td>0.25</td>
<td>1.00</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Method</th>
<th>Success (of 40)</th>
<th>Mean Speed-up</th>
<th>Std. Dev.</th>
<th>Max</th>
</tr>
</thead>
<tbody>
<tr>
<td>Weighted A*</td>
<td>37</td>
<td>34.62</td>
<td>87.74</td>
<td>506.78</td>
</tr>
</tbody>
</table>

- Max planning time of 2 minutes
- Sub-optimality bound of 20 (for E-Graphs and Weighted A*)
Planning with *E*-Graphs for Mobile Manipulation

**Kitchen environment – Path Quality**

<table>
<thead>
<tr>
<th>Method</th>
<th>Success (of 40)</th>
<th>Object XYZ Path Length Ratio</th>
<th>Std. Dev.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Weighted A*</td>
<td>37</td>
<td>0.91</td>
<td>0.68</td>
</tr>
</tbody>
</table>
Planning with $E$-Graphs for Mobile Manipulation

Discussion

**Pros**
- Can lead the search around local minima (such as obstacles)
- Provided paths can have arbitrary quality
- Can reuse many parts of multiple experiences
- User can control bound on solution quality

**Cons**
- Local minima can be created getting on to the E-Graph
- The E-Graph can grow arbitrarily large
Conclusions

• Heuristic Search within sub-optimality bounds
  - can be much more suitable for high-dimensional planning for manipulation than optimal heuristic search
  - still provides
    - good cost minimization
    - consistent behavior
    - rigorous theoretical guarantees

• Planning for manipulation solves similar tasks over and over again (unlike planning for navigation, flight, …)
  - great opportunity for combining planning with learning