Planning & Decision-making in Robotics

Search Algorithms:
Heuristic Functions, Multi-Heuristic A*

Maxim Likhachev
Robotics Institute
Carnegie Mellon University
Example problem: move picture frame on the table

- Full-body planning
- 12 Dimensions
  - 3D base pose
  - 1D torso height
  - 6DOF object pose
  - 2 redundant DOFs in arms
Design of Informative Heuristics

- For grid-based navigation:
  - Euclidean distance
  - Manhattan distance: $h(x,y) = abs(x-x_{goal}) + abs(y-y_{goal})$
  - Diagonal distance: $h(x,y) = max(abs(x-x_{goal}), abs(y-y_{goal}))$
  - More informed distances???

Which heuristics are admissible for 4-connected grid? 8-connected grid?
Design of Informative Heuristics

- For lattice-based 3D \((x, y, \Theta)\) navigation:

  Any ideas?
Design of Informative Heuristics

- For lattice-based 3D \((x,y,\Theta)\) navigation:
  - 2D \((x,y)\) distance accounting for obstacles (single Dijkstra’s on 2D grid cell starting at goal cell will give us these values)
Design of Informative Heuristics

- For lattice-based 3D \((x,y,\Theta)\) navigation:
  - 2D \((x,y)\) distance accounting for obstacles (single Dijkstra’s on 2D grid cell starting at goal cell will give us these values)

*Any problems where it will be highly uninformative?*
Design of Informative Heuristics

• For lattice-based 3D \((x,y,\theta)\) navigation:
  
  – 2D \((x,y)\) distance accounting for obstacles (single Dijkstra’s on 2D grid cell starting at goalcell will give us these values)

Any problems where it will be highly uninformative?

Any heuristic functions that will guide search well in this example?

Carnegie Mellon University
Design of Informative Heuristics

- 20DoF Planar arm planning (*forget optimal A*, *use weighted A*):
Design of Informative Heuristics

- 20DoF Planar arm planning (*forget optimal A*, *use weighted A*):

  key to finding solution fast: shallow minima for $h(s)-h^*(s)$ function
Design of Informative Heuristics

- 20DoF Planar arm planning (*forget optimal A*, use weighted A*):

  *Any ideas?*

  *key to finding solution fast: shallow minima for \( h(s)-h^*(s) \) function*
Design of Informative Heuristics

- 20DoF Planar arm planning (forget optimal A*, use weighted A*):
  - 2D end-effector distance accounting for obstacles

key to finding solution fast: shallow minima for $h(s)-h^*(s)$ function
Design of Informative Heuristics

• 20DoF Planar arm planning (*forget optimal A*, use weighted A*):  
  – 2D end-effector distance accounting for obstacles

*Example where it will miserably fail?*

*Key to finding solution fast: shallow minima for h(s)−h*(s) function*
Design of Informative Heuristics

- Arm planning in 3D:
  
  Any ideas?

key to finding solution fast: shallow minima for $h(s) - h^*(s)$ function

Carnegie Mellon University
Design of Informative Heuristics

- Arm planning in 3D:
  - 3D \((x,y,z)\) end-effector distance accounting for obstacles

\[ S_{start} \rightarrow S_{goal} \]

*key to finding solution fast: shallow minima for \(h(s)-h^*(s)\) function*
Few Properties of Heuristic Functions

- Useful properties to know:
  - $h_1(s), h_2(s)$ – consistent, then:
    \[ h(s) = \max(h_1(s), h_2(s)) \] – consistent

- if A* uses $\varepsilon$-consistent heuristics:
  \[ h(s_{\text{goal}}) = 0 \text{ and } h(s) \leq \varepsilon \ c(s, \text{succ}(s)) + h(\text{succ}(s)) \text{ for all } s \neq s_{\text{goal}}, \]
  then A* is $\varepsilon$-suboptimal:
  \[ \text{cost(solution)} \leq \varepsilon \ \text{cost(optimal solution)} \]

- weighted A* is A* with $\varepsilon$-consistent heuristics

- $h_1(s), h_2(s)$ – consistent, then:
  \[ h(s) = h_1(s) + h_2(s) \] – $\varepsilon$-consistent
Few Properties of Heuristic Functions

- Useful properties to know:
  - \( h_1(s), h_2(s) - \) consistent, then:
    \[
    h(s) = \max(h_1(s), h_2(s)) - \) consistent
  
  - if A* uses \( \varepsilon \)-consistent heuristics:
    \[
    h(s) = 0 \text{ and } h(s) \leq \varepsilon \cdot c(s, succ(s)) + h(succ(s)) \text{ for all } s \neq s_{goal},
    \]
    then A* is \( \varepsilon \)-suboptimal:
    \[
    \text{cost(solution)} \leq \varepsilon \cdot \text{cost(optimal solution)}
    \]

- weighted A* is A* with \( \varepsilon \)-consistent heuristics

- \( h_1(s), h_2(s) - \) consistent, then:
  \[
  h(s) = h_1(s) + h_2(s) - \varepsilon\text{-consistent}
  \]
Example problem: move picture frame on the table

- Full-body planning
- 12 Dimensions (3D base pose, 1D torso height, 6DOF object pose, 2 redundant DOFs in arms)
Admissible and Consistent Heuristic

- $h_0$: base distance
  - 2D BFS from goal state

Do you think it will guide search well?

Any other ideas for good heuristics?
Inadmissible Heuristics

- $h_1$: base distance + object orientation difference with goal
- $h_2$: base distance + object orientation difference with vertical
More generally: we can often easily generate $N$ arbitrary heuristic functions that estimate costs-to-goal.

Solutions to $N$ lower-dimensional manifolds
Solutions to $N$ problems with different constraints relaxed

- $h_2$: base distance + object orientation difference with vertical
Can we utilize a bunch of inadmissible heuristics simultaneously, leveraging their individual strengths while preserving guarantees on completeness and bounded sub-optimality?
Can we utilize a bunch of inadmissible heuristics simultaneously, leveraging their individual strengths while preserving guarantees on completeness and bounded sub-optimality?

Combining multiple heuristics into one (e.g., taking max) is often inadequate

- information is lost
- creates local minima
- requires all heuristics to be admissible
Multi-Heuristics A*: version 1

- Given N inadmissible heuristics
- Run N independent searches
- Hope one of them reaches goal

Within the while loop of the ComputePath function:

\[ \text{for } i=1 \ldots N \]
\[ \text{remove } s \text{ with the smallest } [f(s) = g(s) + w_1 \times h(s)] \text{ from OPEN}_i ; \]
\[ \text{expand } s ; \]
Multi-Heuristics A*: version 1

- Given N inadmissible heuristics
- Run N independent searches
- Hope one of them reaches goal

Problems:
- Each search has its own local minima
- N times more work
- No completeness guarantees or bounds on solution quality
Multi-Heuristics A*: version 2

- Given N inadmissible heuristics
- Run N independent searches
- Hope one of them reaches goal
- Key Idea #1: Share information (g-values) between searches!

Within the while loop of the ComputePath function (note: CLOSED is shared):

1. for $i=1 \ldots N$
   2. remove $s$ with the smallest $[f(s) = g(s)+w_1*h(s)]$ from $OPEN_i$;
   3. expand $s$ and also insert/update its successors into all other $OPEN$ lists;

Inad. Search 1

priority queue: OPEN\(_1\)
key = $g + w_1*h_1$

found paths

Inad. Search 2

priority queue: OPEN\(_2\)
key = $g + w_1*h_2$

found paths

Inad. Search 3

priority queue: OPEN\(_3\)
key = $g + w_1*h_3$
Multi-Heuristics A*: version 2

- Given N inadmissible heuristics
- Run N independent searches
- Hope one of them reaches goal

Key Idea #1: Share information (g-values) between searches!

Benefits:
- Searches help each other to circumvent local minima
- States are expanded at most once across ALL searches

Remaining Problem:
- No completeness guarantees or bounds on solution quality

<table>
<thead>
<tr>
<th>Inad. Search 1</th>
<th>Inad. Search 2</th>
<th>Inad. Search 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>priority queue: OPEN&lt;sub&gt;1&lt;/sub&gt;</td>
<td>priority queue: OPEN&lt;sub&gt;2&lt;/sub&gt;</td>
<td>priority queue: OPEN&lt;sub&gt;3&lt;/sub&gt;</td>
</tr>
<tr>
<td>key = g + w&lt;sub&gt;1&lt;/sub&gt;*h&lt;sub&gt;1&lt;/sub&gt;</td>
<td>key = g + w&lt;sub&gt;1&lt;/sub&gt;*h&lt;sub&gt;2&lt;/sub&gt;</td>
<td>key = g + w&lt;sub&gt;1&lt;/sub&gt;*h&lt;sub&gt;3&lt;/sub&gt;</td>
</tr>
</tbody>
</table>

*found paths*
Multi-Heuristics A* [Aine et al., ’14]

- Given N inadmissible heuristics
- Run N independent searches
- Hope one of them reaches goal
- Key Idea #1: Share information (g-values) between searches!
- Key Idea #2: Search with admissible heuristics controls expansions

Benefits:
- Algorithm is complete and provides bounds on solution quality
Multi-Heuristics A* [Aine et al.,’14]

- Given N inadmissible heuristics
- Run N independent searches
- Hope one of them reaches goal
- Key Idea #1: Share information (g-values) between searches!
- Key Idea #2: Search with admissible heuristics controls expansions

Benefits:
- Algorithm is complete and provides bounds on solution quality
Multi-Heuristics A* [Aine et al.,’14]

• Given N inadmissible heuristics
• Run N independent searches
• Hope one of them reaches goal
• Key Idea #1: Share information (g-values) between searches!
• Key Idea #2: Search with admissible heuristics controls expansions

Benefits:
• Algorithm is complete and provides bounds on solution quality

Within the while loop of the ComputePath function
(note: CLOSED is shared among searches 1…N. Search 0 has its own CLOSED):

\[
\text{for } i=1 \ldots N \\
\text{if(min. } f\text{-value in } OPEN_i \leq w_2 \times \text{min. } f\text{-value in } OPEN_0) \\
\text{remove } s \text{ with the smallest } [f(s) = g(s) + w_1 \times h_i(s)] \text{ from } OPEN_i; \\
\text{expand } s \text{ and also insert/update its successors into all other } OPEN \text{ lists}; \\
\text{else} \\
\text{remove } s \text{ with the smallest } [f(s) = g(s) + w_1 \times h_0(s)] \text{ from } OPEN_0; \\
\text{expand } s \text{ and also insert/update its successors into all other } OPEN \text{ lists;}
\]
Multi-Heuristics A* [Aine et al., ’14]

Within the while loop of the ComputePath function
(note: CLOSED is shared among searches 1…N. Search 0 has its own CLOSED):

for i=1…N

if(min. f-value in OPEN_i ≤ min. f-value in OPEN_0)
    remove s with the smallest \[f(s) = g(s) + w_i \cdot h_i(s)\] from OPEN_i;
    expand s and also insert/update its successors into all other OPEN lists;
else
    remove s with the smallest \[f(s) = g(s) + w_i \cdot h_0(s)\] from OPEN_0;
    expand s and also insert/update its successors into all other OPEN lists;

Carnegie Mellon University
Within the while loop of the ComputePath function (note: CLOSED is shared among searches 1...N. Search 0 has its own CLOSED):

```plaintext
for i = 1...N 
  if(min. f-value in OPEN_i <= w_2 * min. f-value in OPEN_0 )
    remove s with the smallest [f(s) = g(s)+w_1*h_i(s)] from OPEN_i;
    expand s and also insert/update its successors into all other OPEN lists;
  else
    remove s with the smallest [f(s) = g(s)+w_1*h_0(s)] from OPEN_0;
    expand s and also insert/update its successors into all other OPEN lists;
```

Benefits:
- Algorithm is complete
- Provides bounds on solution quality
- Multi-Heuristics A* [Aine et al.,’14]

**Theorem 1:** min. key of OPEN_0 <= w_1*optimal solution cost

**Theorem 2:** min. key of OPEN_i <= w_2*w_1*optimal solution cost

**Theorem 3:** The algorithm is complete and the cost of the found solution is no more than w_2*w_1*optimal solution cost

**Theorem 4:** Each state is expanded at most twice: at most once by one of the inadmissible searches and at most once by the Anchor search
Multi-Heuristics A* [Aine et al.,’14]

- Given N inadmissible heuristics
- Run N independent searches
- Hope one of them reaches goal
- Key Idea #1: Share information (g-values) between searches!
- Key Idea #2 Search with admissible heuristics controls expansions
What You Should Know…

• Examples of heuristic functions
  – for X-connected grids
  – For higher dimensional planning problems derived by lower-dimensional search

• Be able to come up with a good heuristic function for a given problem

• Properties of heuristic functions

• How Multi-heuristic A* works