Decomposed Hierarchical Planning

Jason Wolfe  Bhaskara Marthi  Stuart Russell
UC Berkeley  Willow Garage  UC Berkeley
Real World Decision Making

• Video of PR2 cleaning room
Levels of Decision Making

- Which object to put away next?
- How to arrange objects in cupboard?
- Where to place base to pick up object?
- Where to grasp object?
- What type of grasp to use?
- What is the full joint configuration at grasp?
- What path in cspace to take to achieve grasp?
- What joint efforts to apply to follow path?

*These decisions are not independent!*
Top-down Decision Making

- Make decisions in top-down order
- How to handle lower level planning failures?
- Can be unboundedly suboptimal
This work

- DASH-A* Planner
- Find hierarchically optimal plans
- For efficiency:
  - Decompose across subproblems whenever possible
  - State abstraction: reuse solutions across subproblems
  - Angelic bounds on reachable sets and costs at all levels -> pruning
Pick and Place Domain

• Video
<table>
<thead>
<tr>
<th>HLA</th>
<th>Refinements</th>
</tr>
</thead>
<tbody>
<tr>
<td>Act</td>
<td><img src="image" alt="Action hierarchies" /> MoveToGoal(o), Act</td>
</tr>
<tr>
<td>MoveToGoal(o)</td>
<td>GoPick(o), GoPlace(o,p)</td>
</tr>
<tr>
<td>GoPick(o)</td>
<td>Pick(o), ArmTuck, BaseRgn(r), Pick(o)</td>
</tr>
<tr>
<td>Pick(o)</td>
<td>ArmGraspAction(pos(o), θ), CloseGripperAction(o), TorsoAction(up)</td>
</tr>
<tr>
<td>GoPlace(o,p)</td>
<td>Place(o,p), ArmTuck, BaseRgn(r), Place(o,p)</td>
</tr>
<tr>
<td>Place(o,p)</td>
<td>ArmJointAction(θ₁), TorsoAction(down), OpenGripperAction, ArmJointAction(θ₂)</td>
</tr>
<tr>
<td>ArmTuck</td>
<td>ArmJointAction(tucked)</td>
</tr>
<tr>
<td>BaseRgn(r)</td>
<td>BaseAction(p, θ)</td>
</tr>
</tbody>
</table>
Hierarchical Uniform-Cost Search

- Nodes are (state, plan suffix) pairs

Init

Successors

Goal
Angelic Semantics

- What if we want a heuristic, for hierarchical-A*?

- Angelic semantics provides provably correct abstract transition models
  - optimistic descriptions: overestimate reachability, underestimate costs

Going to undersell things a bit here... also pessimistic descriptions, of potentially much greater interest, since they allow committing to provably correct abstract plans. For today, going to focus on the A* / optimistic-only story, may talk a bit about pessimistic at end.
### Angelic Hierarchical Search Problems

For each action $a$, input set $i$:

<table>
<thead>
<tr>
<th>Optimistic Outcome</th>
<th>Children</th>
<th>Context</th>
</tr>
</thead>
<tbody>
<tr>
<td>Set</td>
<td>Cost</td>
<td>Status</td>
</tr>
<tr>
<td>reachable states</td>
<td>lower bound</td>
<td>solved, refinable, or blocked</td>
</tr>
<tr>
<td>example:</td>
<td>5 s</td>
<td>refinable</td>
</tr>
</tbody>
</table>

---

Precise definition of what goes into angelic search problem. Unifies primitive, high-level semantics. Still have designated initial state, top-level action Act. In status, we see two ways to “refine” -- expand action, or narrow the input set. One way to think about this framework: action-generated state abstractions.
Angelic Hierarchical A* (AHA*)

- Essentially just H-UCS with a heuristic

\[ f = g + h \]

\[ 60 = 0 + 60 \]

- Other difference: can refine any refinable action
AHA* Drawback

• Number of potential plans grows exponentially
  • # refs of action 1 * # refs of action 2 # refs of action 3
  • Even when state space is small!
  • Pruning doesn’t help enough
“Singleton” DASH-A*

- First step towards full DASH-A* algorithm
- DASH-A* features
  - Decomposed
  - Angelic
  - State-abstracted
  - Hierarchical

Essentially same as “explicit DASH–A*” algorithm I talked about awhile ago.
Decomposition

• Given fixed intermediate states, planning for a sequence of HLAs decomposes into independent subproblems
• Tree decomposition of hierarchical plan space

Concatenate two planning problems together.
State Abstraction

- Can use context (relevance) to further increase sharing/compression of open list
Search: min/sum graphs

[Diagram with min/sum graphs]
Search: AO*

1. Expand a best leaf node
   - Start at root
   - Pick min-cost child at min
   - Pick any unsolved child at +
   - Expand reached leaf

2. Propagate labels upwards
   - Rewards follow labels
   - Break ties: solved < refinable
   - Sum solved iff both children solved

3. If root not solved, goto 1
Search: AO*

8, ref. $S_0$ Act $S_g$

8, ref. $S_0 h_1 S_1 p_4 S_g$

5, ref. $S_0 h_1 S_1$

5, ref. $S_0 h_5 S_3$

5, ref. $S_0 h_2 S_3 p_1 S_1$

3, ref. $S_0 h_5 S_3$

3, ref. $S_3 p_1 S_1$

3, solved $S_3 p_1 S_1$

2, solved $S_3 p_1 S_1$

11, ref. $S_0 h_3 S_2 h_4 S_g$

6, ref. $S_1 p_4 S_g$

3, solved $S_1 p_4 S_g$

3, solved $S_3 p_2 S_2$

3, solved $S_3 p_2 S_2$

5, ref. $S_0 h_2 S_2$

5, ref. $S_0 h_2 S_2$

6, ref. $S_0 p_2 S_4 h_8 S_2$

6, ref. $S_0 p_2 S_4 h_8 S_2$

6, ref. $S_0 p_2 S_4 h_8 S_2$

6, ref. $S_0 p_2 S_4 h_8 S_2$

$∞$, ref. $S_0 p_2 S_4 h_8 S_2$

$∞$, ref. $S_0 p_2 S_4 h_8 S_2$

$∞$, ref. $S_0 p_2 S_4 h_8 S_2$

$∞$, ref. $S_0 p_2 S_4 h_8 S_2$

$∞$, ref. $S_0 p_2 S_4 h_8 S_2$

$∞$, ref. $S_0 p_2 S_4 h_8 S_2$
Search: AO*
Search: AO*
Search: AO*

Diagram showing the search process with various states and actions, connecting to 8, ref., 5, ref., 6, ref., and ∞, ref. with solved statuses.
Search: AO*

- **S0 Act Sg**
- **S0 h1 S1 p4 Sg**
- **S0 h2 S2 h4 Sg**
- **S0 h3 S3 p2 S1**
- **S0 p4 S7**

**Paths:**
- 8, ref.
- 5, ref.
- 7, solved
- 5, solved
- 2, solved
- 3, solved
- 3, solved
- 2, solved

**Solved:**
- 7, solved
- 5, solved
- 5, solved
- 2, solved
- 3, solved
- 3, solved
- 2, solved

**References:**
- 8, ref.
- 5, ref.
- 7, solved
- 5, solved
- 2, solved
- 3, solved
- 3, solved
- 2, solved
Search: AO*
Solution: p₃ p₄ p₁ p₄
Properties of “singleton” DASH-A*

• hierarchically optimal
• each subproblem solved at most once
• always works on subproblem that contributes to global cost bound
• can be exponentially faster than AH-A*
Singleton DASH-A*

(on discrete version of mobile manipulation domain)
(General) DASH-A*

- What if angelic sets are not singletons?
  - Implicit sets are much more compact
  - Focusing on concrete states can break abstraction, bringing unimportant low-level details to high-level
  - Sometimes, explicit outcomes not known in advance

Implicit outcomes of **GoPut**

Explicit outcomes of **GoPut**
Planning with implicit sets

- Win: if we avoid refining a plan due to optimistic bounds being suboptimal (enough), never need to get to level of concrete states
DASH-A*: first attempt

\[ S_0 \text{ Act } S_g \]

\[ \min \]

\[ S_0 h_1 h_2 S_g \]

7, ref.

8, ref.

\[ S_0 h_3 h_4 S_g \]

7, ref.

\[ S_0 h_1 \]

5, ref.

8, ref.

\[ h_2 S_g \]

3, ref.

\[ S_0 h_3 \]

6, ref.

\[ h_4 S_g \]

1, ref.

\[ S_0 h_3 \]

6, ref.

\[ h_4 S_g \]

1, ref.

\[ S_0 h_3 \]

6, ref.

\[ h_4 S_g \]

1, ref.

\[ S_0 h_3 \]

6, ref.

\[ h_4 S_g \]

1, ref.
DASH-A*: looking good!
DASH-A*: looking good!

\[ S_0^{\text{Act} \ S_g} \xrightarrow{\text{min}} S_0^{h_1 \oplus h_2 S_g} \]
\[ 8, \text{ ref.} \]
\[ S_0^{h_1} \oplus S_0^{h_2 S_g} \]
\[ 5, \text{ ref.} \]
\[ 20, \text{ ref.} \]
\[ \ldots \]
\[ 3, \text{ ref.} \]
DASH-A*: looking good!

\[
\begin{align*}
S_0 &\xrightarrow{\text{Act}} S_g \\
S_0 &\xrightarrow{\text{h}_1} (\text{h}_2) S_g \\
(\text{h}_1) &\xrightarrow{\text{h}_2 S_g} 3, \text{ blocked} \\
(\text{h}_1) &\xrightarrow{\text{p}_1 S_g} 3, \text{ blocked} \\
(\text{h}_1) &\xrightarrow{\text{p}_2 S_g} 10, \text{ blocked} \\
\end{align*}
\]
DASH-A*: what now?!

9, blocked  
\[ S_0 \text{ Act } S_g \]

9, blocked  
\[ S_0 h_1 h_2 S_g \]

6, solved  
\[ S_0 h_1 S_2 \]

6, solved  
\[ S_0 h_1 S_2 \]

8, solved  
\[ S_0 p_4 S_3 \]

3, blocked  
\[ h_2 S_g \]

3, blocked  
\[ p_1 S_g \]

10, blocked  
\[ p_2 S_g \]

20, ref.

...
DASH-A*: challenges

- Without concrete intermediate states, sequences do not cleanly decompose

*..g,
Implicit DASH-A*: challenges

- Without concrete intermediate states, sequences do not cleanly decompose

\[ \text{e.g.,} \]
Implicit DASH-A*: challenges

- Without concrete intermediate states, sequences do not cleanly decompose
  - must find multiple optimal solutions (to different states) for each subproblem
- As search proceeds, we must split outcome sets
  - structure of the graph changes as we go
  - splitting must propagate through later actions

...
DASH-A*: specialization
DASH-A*: specialization
DASH-A*: specialization
DASH-A*: specialization

Solution: \( p_3 p_1 \)
DASH-A*: Analysis and Results

• DASH-A* is systematic, hierarchically optimal
• Easy to construct examples where DASH-A* is exponentially faster than previous algorithms

<table>
<thead>
<tr>
<th>Domain</th>
<th>size</th>
<th>optimal len</th>
<th>LAMA</th>
<th>SAHTIN</th>
<th>ANA*</th>
<th>DASH-A</th>
</tr>
</thead>
<tbody>
<tr>
<td>chess</td>
<td>100x100</td>
<td>202</td>
<td>14.53</td>
<td>834</td>
<td>0.31</td>
<td>5439</td>
</tr>
<tr>
<td></td>
<td>500x500</td>
<td>1005</td>
<td>71.04</td>
<td>4014</td>
<td>1.74</td>
<td>19900</td>
</tr>
<tr>
<td>grid</td>
<td>3 objects</td>
<td>35</td>
<td>14.5</td>
<td>6067</td>
<td>14.72</td>
<td>1420</td>
</tr>
<tr>
<td></td>
<td>2 objects</td>
<td>53</td>
<td>28.52</td>
<td>51169</td>
<td>178.24</td>
<td>1728</td>
</tr>
<tr>
<td></td>
<td>3 objects</td>
<td>72</td>
<td>382.02</td>
<td>629563</td>
<td>656.80</td>
<td>1953</td>
</tr>
<tr>
<td></td>
<td>4 objects</td>
<td>101</td>
<td>1178</td>
<td>4893624</td>
<td>1594.4</td>
<td>47686</td>
</tr>
<tr>
<td>roos</td>
<td>1 object</td>
<td>15</td>
<td>13.2</td>
<td>55</td>
<td>2.73</td>
<td>168</td>
</tr>
<tr>
<td></td>
<td>2 objects</td>
<td>30</td>
<td>15.36</td>
<td>212</td>
<td>29.29</td>
<td>2473</td>
</tr>
<tr>
<td></td>
<td>3 objects</td>
<td>42</td>
<td>17.76</td>
<td>319</td>
<td>235.28</td>
<td>20145</td>
</tr>
</tbody>
</table>

1: Runtimes in seconds and number of optimistic + primitive model evaluations to optimally solve random instances of domains. Results are medians over 5 instances of each size, with a memory limit of 512 MB.
Conclusion and Future work

• DASH-A* algorithm
  • Find hierarchically optimal plans
  • Decompose across subproblems
  • State abstraction to reuse solutions
  • Angelic bounds to prune search space

• Future work
  • Bounded-suboptimal DASH-A*
  • Concurrency
  • Partially observable/stochastic domains