BACKWARD-FORWARD SEARCH FOR MANIPULATION PLANNING

Caelan Garrett, Tomás Lozano-Pérez, and Leslie Kaelbling
MIT CSAIL - IROS 2015
Hybrid Planning

- Mixed discrete/continuous state & actions
Hybrid Planning

- **Mixed discrete/continuous state & actions**
- e.x. robotic planning
  - Continuous - robot configuration, object poses, grasp transforms, …
  - Discrete - holding object label, object cleaned/cooked, …
Hybrid Planning

- Mixed discrete/continuous state & actions
  - e.x. robotic planning
    - Continuous - robot configuration, object poses, grasp transforms, ...
    - Discrete - holding object label, object cleaned/cooked, ...

- Hybrid Backward-Forward (HBF) algorithm
  - Probabilistically complete
  - Efficient empirical performance
High-Dimensional Manipulation

Separate **blue blocks** and **green blocks**

Early state  
Late state
Infinite Branching Factor and Long Horizon

- Pure forward or backward search overwhelmed
- Unguided action sampling ineffective
Infinite Branching Factor and Long Horizon

- Pure forward or backward search overwhelmed
- Unguided action sampling ineffective
- Approximate backwards search focuses sampling
Action Template Representation

\textbf{Place}(\texttt{config}, \texttt{obj}, \texttt{transform})
**Action Template Representation**

\[
\text{Place}(\text{config, obj, transform})
\]

\[
\text{constraints}
\]

robot = config
holding = obj
grasp = transform
**Action Template Representation**

\[
\textbf{Place}(\text{config, obj, transform})
\]

**constraints**
- `robot = config`
- `holding = obj`
- `grasp = transform`

**effects**
- `holding = None`
- `obj = \textbf{Pose}(\text{config, obj, transform})`
**Action Template Representation**

\[ \text{MoveHolding}(config_1, config_2, objA, \text{transform}) \]

**constraints**
- robot = config_1
- holding = objA
- grasp = transform
- \( objB \in \text{CollisionFreePoses}(config_1, config_2, objA, \text{transform}, objB) \text{ for } objB \neq objA \)

**effects**
- robot = config_2
- objA = \text{Pose}(config_2, objA, \text{transform})
Example Hybrid Planning Problem

- Constraints define regions of state-space
- Goal is to reach a state in the intersection of Constraint1 and Constraint2

Start State

Goal = Constraint1 & Constraint2
Unguided Search is Ineffective

- Pure forward or backward search overwhelmed by infinite branching factor and long horizon

Start State

Goal = Constraint1 & Constraint2

Constraint1

Constraint2

Constraint3
Hybrid Backward-Forward (HBF)

- Backwards approximation - constraint independence
Hybrid Backward-Forward (HBF)

- Backwards approximation - constraint independence
- Long problem becomes many short problems
Hybrid Backward-Forward (HBF)

- Backwards approximation - constraint independence
- Long problem becomes many short problems

![Diagram showing the concept of Hybrid Backward-Forward with constraints and states.]
Hybrid Backward-Forward (HBF)

- Backwards approximation - constraint independence
- Long problem becomes many short problems
- Forward search resolves approximation errors
Hybrid Backward-Forward (HBF)

- Backwards approximation - **constraint independence**
- Long problem becomes **many short problems**
- Forward search **resolves** approximation errors

Start State

Goal = Constraint1 & Constraint2
Hybrid Backward-Forward (HBF)

- Backwards approximation - constraint independence
- Long problem becomes many short problems
- Forward search resolves approximation errors

Start State

Goal = Constraint1 & Constraint2
Hybrid Backward-Forward (HBF)

- Backwards approximation - constraint independence
- Long problem becomes many short problems
- Forward search resolves approximation errors

Start State: Constraint3

Constraint1

Constraint3

Constraint2

Goal = Constraint1 & Constraint2
Backwards Search Algorithm

\textbf{BackwardsSearch}(state, goal-constraints, action-templates):

1. Queue initialized to goal-constraints

2. Repeat

   3. Pop a constraint from the queue

   4. Sample actions from each action-template that achieve the constraint

   5. Yield actions applicable from state to forward search

   6. Add the new action constraints not satisfied by state to the queue
Action Sampling

- Need to generate actions that satisfy a constraint

- For manipulation, we sample:
  - Configs/grasps/poses that satisfy the constraint
    - Monte Carlo rejection sampling
  - Other relevant configs/grasps/poses
    - Current state, intersection of manifolds
  - Actions that connect these values
    - Blackbox motion planner (e.x. RRT)

- Reuse previously sampled values/actions when possible
Push A into Yellow Cabinet

- Robot is a point
- Goal - object A in yellow cabinet
  - Need to move object B!
- Trace backwards search
Backwards Search - Iteration 1

- Search starts at the goal constraint
Backwards Search - Iteration 1

- Pop constraint “A in cabinet” from the queue

| holding = None | A = poseA0 | B = poseB0 | C = poseC0 | robot = config0 |
Backwards Search - Iteration 1

- Sample **Push** actions
- Add new constraints to queue
Backwards Search - Iteration 2

- Pop constraint “B not colliding” with the push from the queue
Backwards Search - Iteration 2

- Sample collision-free **Place** for B
- Add new constraints to the queue
Backwards Search - Continued...

- Continue process...
Move actions are forward search successors
Forward Search Algorithm

- Forward state-space heuristic search
  - Hill climbing, best-first search, ...

- Backward search focuses forward search by:
  - Identifying successor actions
  - Incidentally creating approximate plans
    - **Heuristic** is length of an approximate plan (idea from AI planning)
**Definition** - A hybrid planning problem is **robustly feasible** if there exists a sequence of action templates with a family of solutions such that for all state and pre-image pairs on a solution, the solution family actions that partially satisfy the state and achieve the pre-image have **nonzero measure**.

**Lemma** - The backwards search performed from any state in the solution family will generate a successor action in the solution family with a **probability of one** as $n \to \infty$.

**Theorem** - HBF will solve any robustly feasible hybrid planning problem with a **probability of one** as $n \to \infty$.

**Proof** - Inductively apply the lemma.
Non-prehensile Actions

Push **green cylinder** to **green dot**
Success rate: **100%**
Runtime: **7 sec**

Python, 60 trials, 300 sec timeout, median runtime
High-Dimensional State-Space

Pick and place green block
Success Rate: 100%
Runtime: 4 sec

Python, 60 trials, 300 sec timeout, median runtime
Regrasping and Non-monotonicity

Move **blue block**, regrasp **green block**, and replace **blue block**
Success Rate: **100%**
Runtime: **6 sec**

Python, 60 trials, 300 sec timeout, median runtime
Dynamic Unstacking and Stacking

Unstack **red block** and stack **black block** on **blue block**
Success Rate: **97%**
Runtime: **12 sec**
Comparison with Srivastava et al.

Grasp red cylinder on crowded table
Single goal but must move many objects

<table>
<thead>
<tr>
<th></th>
<th>Success Rate</th>
<th>Runtime</th>
</tr>
</thead>
<tbody>
<tr>
<td>Srivastava et al.</td>
<td>63%</td>
<td>68 sec</td>
</tr>
<tr>
<td>HBF</td>
<td>98%</td>
<td>23 sec</td>
</tr>
</tbody>
</table>

Python, 60 trials, 300 sec timeout, median runtime
Comparison with FFRob

Separate **blue blocks** and **green blocks**

Many goals and must order achieving the goals

<table>
<thead>
<tr>
<th></th>
<th>Success Rate</th>
<th>Runtime</th>
</tr>
</thead>
<tbody>
<tr>
<td>FFRob</td>
<td>84%</td>
<td>157 sec</td>
</tr>
<tr>
<td>HBF</td>
<td>100%</td>
<td>82 sec</td>
</tr>
</tbody>
</table>

Python, 60 trials, 300 sec timeout, median runtime
Takeaways

- General purpose hybrid planing algorithm (HBF)
  - Approximate backward search
    - Focuses successor actions to reduce branching factor
    - Gives heuristic cost
  - Forwards search
    - Resolves backwards approximations
- Probabilistically complete
- Application - efficiently solves manipulation problems
Any Questions?

Separate **blue blocks** and **green blocks**
Success rate: **100%**
Runtime: **82 sec**
Median Runtime (300s Timeout)

- Problem 1
- Problem 2
- Problem 3
- Problem 4
- Problem 5
- Problem 6

H 0

H FF
Full Experimental Results

<table>
<thead>
<tr>
<th>P</th>
<th>%</th>
<th>runtime</th>
<th>length</th>
<th>visited</th>
<th>%</th>
<th>runtime</th>
<th>length</th>
<th>visited</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>100</td>
<td>12 (7)</td>
<td>8 (0)</td>
<td>156 (140)</td>
<td>100</td>
<td>4 (1)</td>
<td>12 (2)</td>
<td>6 (2)</td>
</tr>
<tr>
<td>2</td>
<td>97</td>
<td>62 (26)</td>
<td>16 (0)</td>
<td>208 (37)</td>
<td>100</td>
<td>7 (1)</td>
<td>16 (0)</td>
<td>10 (1)</td>
</tr>
<tr>
<td>3</td>
<td>62</td>
<td>238 (62)</td>
<td>16 (0)</td>
<td>2315 (712)</td>
<td>100</td>
<td>6 (1)</td>
<td>16 (0)</td>
<td>37 (2)</td>
</tr>
<tr>
<td>4</td>
<td>0</td>
<td>300 (0)</td>
<td>- (-)</td>
<td>1293 (93)</td>
<td>97</td>
<td>12 (4)</td>
<td>24 (4)</td>
<td>56 (24)</td>
</tr>
<tr>
<td>5</td>
<td>0</td>
<td>300 (0)</td>
<td>- (-)</td>
<td>1591 (381)</td>
<td>98</td>
<td>23 (9)</td>
<td>24 (4)</td>
<td>85 (37)</td>
</tr>
<tr>
<td>6</td>
<td>0</td>
<td>300 (0)</td>
<td>- (-)</td>
<td>637 (40)</td>
<td>100</td>
<td>82 (13)</td>
<td>72 (4)</td>
<td>191 (37)</td>
</tr>
</tbody>
</table>

- 6 problems
- 60 trials per algorithm and problem
- Timeout of 300 seconds
- Median statistics (MAD in parentheses)
- Python implementation uses OpenRAVE
Problem 6 $H_{FF}$ Runtime Histogram

Median  Mean
Prior Work

- **Manipulation Planning**
  - Lozano-Pérez - explicit configuration space
  - Siméon et al. - manipulation graph
  - Hauser - multi-modal motion planning
  - Barry et al. - multi-modal biRRT

- **Task and Motion Planning**
  - Lagriffoul, Saffiotti, Dornhege & Nebel, Cambon et al., …
  - Srivastava et al. - planner-independent interface
  - Garrett et al. - FFRob

- **Discrete Planning**
  - Bonet & Geffner - HSP
  - Hoffmann & Nebel - FF