Fast Optimal Replanning

Sven Koenig

http://www.cc.gatech.edu/fac/Sven.Koenig/

Collaborators:
Maxim Likhachev,
David Furcy, Yaxin Liu
(Additional Programming: Colin Bauer, William Halliburton)

Motivation

- replanning (and plan reuse) is important!
- world changes over time
- model of the world changes over time
- what-if analyses

emergency management

planning task 1
slightly different planning task 2
slightly different planning task 3
...

Structure of the Talk

- overview of incremental heuristic search
- Lifelong Planning A* (LPA*) and its properties
- applications of incremental heuristic search
  - symbolic planning (with HSP)
  - continual planning
  - one-time planning
  - mobile robotics
  - mapping
  - goal-directed navigation in unknown terrain
  - computer games
  - reinforcement learning and on-line dynamic programming
  - control (with the Parti-Game algorithm)

Path Planning - Example

we assume here that the robot can move in eight directions

original eight-connected gridworld
Path Planning - Example
we assume here that the robot can move in eight directions

changed eight-connected gridworld

Path Planning - Example
we assume here that the robot can move in eight directions

original eight-connected gridworld

Artificial Intelligence
Algorithm Theory

Artificial Intelligence
Algorithm Theory

heuristic search
incremental search

how to search efficiently using heuristic to guide the search
how to search efficiently by reusing information from previous searches
### Path Planning - Lifelong Planning A*

<table>
<thead>
<tr>
<th>uninformned search</th>
<th>heuristic search</th>
</tr>
</thead>
<tbody>
<tr>
<td>Breadth-First Search</td>
<td>A* [Hart, Nilsson, Raphael, 1968]</td>
</tr>
<tr>
<td>DynamicSWSF-FP with early termination (our addition) [Ramalingam, Reps, 1996]</td>
<td>Lifelong Planning A*</td>
</tr>
</tbody>
</table>

### Path Planning - Experimental Evaluation

**original eight-connected gridworld**

<table>
<thead>
<tr>
<th>uninformned search</th>
<th>heuristic search</th>
</tr>
</thead>
<tbody>
<tr>
<td>incremental search, complete search</td>
<td>incremental search, complete search</td>
</tr>
<tr>
<td>Lifetime Planning A*</td>
<td>Lifetime Planning A*</td>
</tr>
</tbody>
</table>

**changed eight-connected gridworld**

<table>
<thead>
<tr>
<th>uninformned search</th>
<th>heuristic search</th>
</tr>
</thead>
<tbody>
<tr>
<td>incremental search, complete search</td>
<td>incremental search, complete search</td>
</tr>
<tr>
<td>Lifetime Planning A*</td>
<td>Lifetime Planning A*</td>
</tr>
</tbody>
</table>

### Path Planning - Experimental Evaluation

**original eight-connected gridworld**

<table>
<thead>
<tr>
<th>uninformned search</th>
<th>heuristic search</th>
</tr>
</thead>
<tbody>
<tr>
<td>ve = 1331.7 +/- 4.4</td>
<td>ve = 284.0 +/- 5.9</td>
</tr>
<tr>
<td>va = 26207.2 +/- 84.0</td>
<td>va = 6177.3 +/- 129.3</td>
</tr>
<tr>
<td>hp = 5985.3 +/- 19.7</td>
<td>hp = 1697.3 +/- 39.9</td>
</tr>
</tbody>
</table>

**changed eight-connected gridworld**

<table>
<thead>
<tr>
<th>uninformned search</th>
<th>heuristic search</th>
</tr>
</thead>
<tbody>
<tr>
<td>ve = 173.0 +/- 4.9</td>
<td>ve = 25.6 +/- 2.0</td>
</tr>
<tr>
<td>va = 5697.4 +/- 167.0</td>
<td>va = 1235.9 +/- 75.0</td>
</tr>
<tr>
<td>hp = 956.2 +/- 26.6</td>
<td>hp = 240.1 +/- 16.9</td>
</tr>
</tbody>
</table>

ve = vertex expansions, va = vertex accesses, hp = heap percolates
Path Planning - Lifelong Planning A*

[Koenig, Likhachev, 2001]

- applies to the same finite search problems as A*
- handles arbitrary edge cost changes
- produces the same (optimal) solution as A*
- is algorithmically very similar to A*
- is more efficient than A* in many situations
- has nice theoretical properties
Path Planning - Lifelong Planning A*

Initial State:

- Start: (1, 1) in state A
- Goal: (6, 6) in state D
- Priority Queue: D3: [4,3]; C3: [6,4]

Path Planning - Lifelong Planning A*

Updated State:

- Start: (1, 1) in state A
- Goal: (6, 6) in state D
- Priority Queue: D3: [4,3]; C3: [6,4]

Path Planning - Lifelong Planning A*

Updated State:

- Start: (1, 1) in state A
- Goal: (6, 6) in state D
- Priority Queue: D2: [4,4]; D4: [6,4]; D3: [6,5]

Path Planning - Lifelong Planning A*

Updated State:

- Start: (1, 1) in state A
- Goal: (6, 6) in state D
- Priority Queue: D2: [4,4]; D4: [6,4]; D3: [6,5]
Theorem: [Likhachev and Koenig, 2001]

ComputeShortestPath() expands every vertex at most twice and thus terminates.

Theorem: [Likhachev and Koenig, 2001]

After ComputeShortestPath() terminates, one can trace back a shortest path from the start to the goal by always moving from the current vertex $s$, starting at the goal, to any predecessor $s'$ that minimizes $g(s') + c(s', s)$ until the start is reached (ties can be broken arbitrarily).

Theorem: [Likhachev and Koenig, 2001]

ComputeShortestPath() does not expand any vertices whose $g$-values were equal to their respective start distances before ComputeShortestPath() was called.

$LPA^*$ is efficient because it uses incremental search

ComputeShortestPath() expands at most those vertices $s$ with $[f(s); g^*(s)] \leq [f(s_{goal}); g^*(s_{goal})]$ or $[g_{old}(s) + h(s); g_{old}(s)] \leq [f(s_{goal}); g^*(s_{goal})]$, where $f(s) = g^*(s) + h(s)$ and $g_{old}(s)$ is the $g$-value of $s$ directly before the call to ComputeShortestPath().

$LPA^*$ is efficient because it uses heuristic search

In the worst case, replanning cannot be more efficient than planning from scratch. [Nebel, Koehler, 1995]

However, the overhead of $LPA^*$ is bounded.

The first search of $Lifelong Planning A^*$ is the same as that of $A^*$. Afterwards, $Lifelong Planning A^*$ operates in a very similar way to $A^*$. (The theorem makes this more concrete. For example, ComputeShortestPath() expands locally overconsistent vertices with finite $f$-values in the same order as $A^*$.)
Applications

- route planning
- in traffic networks
- in computer networks

- symbolic planning (with HSP)
- continual planning
- one-time planning
- mobile robotics
- mapping
- goal-directed navigation in unknown terrain
- computer games
- reinforcement learning and on-line dynamic programming
- control (with the Parti-Game algorithm)

Symbolic Planning (with HSP) - Continual Planning

- plan adaptation
- repair-based planning
- learning search control knowledge
- case-based planning
- transformational planning
- iterative repair methods in scheduling

CHEF, GORDIUS, LS-ADJUST-PLAN, MRL, NoLimit, PLEXUS, PRIAR, SPA...

plan quality of replanning is usually worse than plan quality of planning from scratch

- lifelong planning
  SHERPA
  plan quality of replanning is as good as plan quality of planning from scratch

Symbolic Planning (with HSP) - Continual Planning

STRIPS-type planning in the elevator domain

Operators:

- The elevator moves from floor $f_i$ to floor $f_j$ with $i \neq j$.
- Person $p_k$ boards the elevator on floor $f_i$ provided that the elevator is currently on floor $f_i$ and floor $f_i$ is the origin of person $p_k$.
- Person $p_k$ gets off the elevator on floor $f_i$, provided that person $p_k$ is in the elevator, the elevator is currently on floor $f_i$, and floor $r_i$ is the destination of person $p_k$.

Symbolic Planning (with HSP) - Continual Planning

SHERPA

Speedy HEuristic search-based RePlAnner

[S. Koenig, D. Furcy, C. Bauer, 2002]

planning problem 1  planning problem 2  planning problem 3
Symbolic Planning (with HSP) - Continual Planning

First search in the elevator domain

Similar to HSP 2.0 with the $h_{\text{max}}$ heuristic

[Bonet, Geffner, 2001]

Symbolic Planning (with HSP) - Continual Planning

Second search in the elevator domain (from scratch)

Similar to HSP 2.0 with the $h_{\text{max}}$ heuristic

[Bonet, Geffner, 2001]

Symbolic Planning (with HSP) - Continual Planning

Second search in the elevator domain (with SHERPA)

Symbolic Planning (with HSP) - Continual Planning

SHERPA achieves speedups up to 80 percent

Savings percentage vs. number of people

Planning from scratch vs. planning for elevator (5 floors)
Symbolic Planning (with HSP) - Continual Planning

- Old search tree
- Start
- Goal
- New search tree

Symbolic Planning (with HSP) - Continual Planning

- ve for blockworld
- Number of deleted ground operators
- Savings percentage
- Planning problem 1
- Planning problem 2
- Planning problem 3

PINCH
Prioritized, INCremental Heuristics calculation

[Liu, Koenig, Furcy, 2002]
**Symbolic Planning (with HSP) - One-Time Planning**

**PINCH**

Prioritized, INCremental Heuristics calculation

Here: for HSP 2.0 with the \( h_{add} \) heuristic [Bonet, Geffner, 2001]

\[
h_{add}(\text{state}) = \sum_{\text{proposition in goal state}} g_{\text{state}}(\text{proposition})
\]

\[
g_{\text{state}}(\text{proposition}) = \begin{cases} 
0 & \text{if proposition in state} \\
\min_{\text{operator with proposition in add list}} (1 + g_{\text{state}}(\text{operator})) & \text{otherwise}
\end{cases}
\]

\[
g_{\text{state}}(\text{operator}) = \sum_{\text{proposition on precondition list of operator}} g_{\text{state}}(\text{proposition})
\]

Order of state expansions

PINCH achieves speedups up to (another!) 80 percent.

---

**Mobile Robotics - Mapping**

we assume here that the robot can move in eight directions

Greedy Mapping always moves the robot on a shortest path to the closest unobserved (or unvisited) cell.

[Koenig, Tovey, Halliburton, 2001] [Thrun et al. 1998] [Romero, Morales, Sucar, 2001]

For example, our implementation combines greedy mapping and schema-based navigation (MissionLab) [Mackenzie, Arkin, Cameron, 1997]

Mobile Robotics - Mapping

can easily be integrated into robot architectures (“reactive planning”) does not need to be in control of the robot at all times (“reactive planning”)
Greedy Mapping always moves the robot on a shortest path to the closest unobserved (or unvisited) cell.

**A** = overhead of LPA* (D* Lite) without incremental Search (A*)

**B** = overhead of LPA* (D* Lite) without heuristic search
Mobile Robotics - Navigation in Unknown Terrain

Planning with the Freespace Assumption always moves the robot on a shortest potentially unblocked path to the goal cell.

[Brumitt and Stentz, 1998] [Hebert, McLachlan, Chang, 1999] [Matthies et al., 2000] [Stentz and Hebert, 1995] [Thayer et al., 2000]

We assume here that the robot can move in eight directions.

Planning with the Freespace Assumption always moves the robot on a shortest potentially unblocked path to the goal cell.

we assume here that the robot can move in eight directions

- Demo Vehicles of the Darpa UGV II Program
- Mars Rover Prototype
- Prototypes of Urban Reconnaissance Robots

HMMWV that navigated 1,410 meters of natural outdoor terrain in 1995

[Stentz and Hebert, 1995]
A = overhead of LPA* (D* Lite) without incremental Search (A*)
B = overhead of LPA* (D* Lite) without heuristic search

Fast Optimal Replanning; (c) Sven Koenig; Georgia Tech; January 2002.

Mobile Robotics - Navigation in Unknown Terrain

Game Playing

Reinforcement Learning and On-Line DP

while there exists at least one state with $g(s) \neq \text{rhs}(s)$
pick a state $s$ with $g(s) \neq \text{rhs}(s)$ and then set $g(s) := \text{rhs}(s)$

Prioritized Sweeping [Moore and Atkeson; 1993]
- chooses the g-value of which state to update
- updates the g-value of the chosen state in a particular way
- minimizes the expected or worst-case plan-execution cost for MDPs

Minimax LPA*
- chooses the g-value of which state to update
- updates the g-value of the chosen state in a particular way
- terminates immediate once a shortest path is found
- uses heuristics to focus the search
- minimizes the worst-case plan-execution cost for MDPs
and so on, for a total of 22 g-value updates. Minimax LPA* needs only 6.

Control (with the Parti-Game algorithm)

Parti-Game algorithm (Moore and Atkeson; 1995)

nonuniform discretization avoids these problems
### Control (with the Parti-Game algorithm)

terrains of size 2000 x 2000

<table>
<thead>
<tr>
<th>Implementation</th>
<th>Planning Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Uninformed Search from Scratch</td>
<td>362 minutes 55 seconds</td>
</tr>
<tr>
<td>Informed Search from Scratch</td>
<td>135 minutes 15 seconds</td>
</tr>
<tr>
<td>Uninformed Incremental Search</td>
<td>14 minutes 53 seconds</td>
</tr>
<tr>
<td>Informed Incremental Search (Minimax LPA*)</td>
<td>13 minutes 53 seconds</td>
</tr>
</tbody>
</table>

---

### More Information on Fast Optimal Replanning

Please see


We gratefully acknowledge funding from NSF and IBM.

The views and conclusions contained in this material are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the sponsoring organizations and agencies or the U.S. government.