Hierarchical path planning for multi-size agents in heterogeneous environments

Daniel Harabor, Adi Botea
NICTA & The Australian National University

IEEE Symposium on Computational Intelligence and Games
December 18, 2008
Outline

- Motivation
- Prior Work
- Planning with Clearance Values
- Abstraction and Hierarchical Planning
- Results
- Conclusion
Motivation

Path planning literature is full of assumptions... That don't always hold in reality!

Knowledge engineering challenges:
  o Identifying relevant domain-specific information.
  o Extracting it automatically.
  o Exploiting it to guide search.

Application areas:
  • Video games.
  • GPS systems.
  • Any path planning system with heterogeneous agents.
GPS Fail

Pictures: Danfung Dennis, NYTimes (04.12.2007)
Problem definition

Example map

Example agent types

Small

Big

Terrain traversal capabilities:
- Ground
- Trees
- Ground or Trees

Movement rules

OK

OK

Not OK
Previous Work

- A* [Hart et al, 1968]
- Brushfire [Latombe, 1991]
- HPA* [Botea et al, 2004]
- PRA* [Sturtevant & Buro, 2005]
- TA*/TRA* [Demyen & Buro, 2006]
- CMM [Geraerts & Overmars, 2007]
Intuition: Calculate size of maximum traversable area at each octile (clearance value).
Results (toymap)

Clearance values for different capabilities:

Ground

Trees

Ground or Trees
Clearance values for different capabilities:

Ground

Trees

Ground or Trees

Space complexity: \[ |CV| = (|V| - |V_{HO}|) \times 2^{r-1} \]
Results (toymap)

Clearance values for different capabilities:

Ground

Trees

Ground or Trees

Space complexity: \( |CV| = (|V| - |V_{HO}|) \times 2^{r-1} \)

Compute on demand!

Emoticons: Wikimedia Foundation
• Theorem 1: Any problem involving an agent of arbitrary size and capability can be reduced into a canonical problem (agent size = 1, capability = 1 terrain).
Annotated A*

Search process:
• Similar to A*.
• Extra parameters: Agent's size and capability.
• Only expand nodes with clearance > agent size.

Pros:
• Works great!

Cons:
• For small problem sizes...
Intuition: Use hierarchical search. Apply cluster-based abstraction as per [Botea et al, 2004]
Inter-cluster Transitions

Approach: Build abstract graph by finding all entrances between clusters.

Identify entrances
Inter-cluster Transitions

Approach: Build abstract graph by finding all entrances between clusters.

Identify entrances
Approach: Build abstract graph by finding all entrances between clusters.
Approach: Build abstract graph by finding all entrances between clusters.

Identify entrances
Inter-cluster Transitions

Approach: Build abstract graph by finding all entrances between clusters.

Identify entrances

Identify transition points
Inter-cluster Transitions

Approach: Build abstract graph by finding all entrances between clusters.

Identify entrances

Identify transition points

Final result
Intra-cluster Transitions

Approach: Use AA* to find all paths between each pair of nodes inside a cluster.

If a path exists, add a new intra-edge edge to abstract graph; annotated with the capability and size parameters used by AA*
Compacting the abstract graph

Method produces a representationally complete graph but can get rather large.

Solutions:

• Strong dominance
• Weak dominance.
Intuition: Retain paths with larger clearance, all else being equal.

Result: High quality abstraction.
Strong dominance example

Intuition: Retain paths with larger clearance, all else being equal.

Result: High quality abstraction.
Intuition: Retain paths with larger clearance, all else being equal.

\[ E3 \succ E5 \]
\[ E4 \succ E6 \]

Result: High quality abstraction.
Weak dominance example

Intuition: Retain edges with large clearance traversable by many agents (freeways vs. trails)

Result: low quality abstraction
Hierarchical Annotated A*

- Extends HPA* [Botea et al 2004]
  - Insert start and goal into abstract graph
  - Find a hierarchical solution
  - Refine
- AA* for insertion step.
- Hierarchical search is a variation on A*
  - Requires agent size and capability as parameters
  - Only add successors to open list if edge is traversable
Experiments: Setup

- 120 maps from Baldur's Gate.
- 3 cluster sizes (10, 15, 20)
- 5 derivative sets
  - Randomly interspersed each map with second terrain type (10%, 20%, 30%, 40% and 50%).
- 2 agent sizes (1 and 2).
- Randomly assigned capability
- 100 valid problems per map.
- Each agent size solves each problem.
- Intel Core2 Duo @ 2.4GHz w/ 1GB RAM (OSX 10.5.2)
- Implemented using Hierarchical Open Graph
- Source code at: http://ahastar.googlecode.com
Original gridmaps averaged 4469 nodes & 16420 edges.

### Best case (Cluster size = 20)

<table>
<thead>
<tr>
<th></th>
<th>HQ</th>
<th>LQ</th>
</tr>
</thead>
<tbody>
<tr>
<td>Nodes</td>
<td>4.0%</td>
<td>2.0%</td>
</tr>
<tr>
<td>Edges</td>
<td>5.0%</td>
<td>0.9%</td>
</tr>
</tbody>
</table>

### Worst case (Cluster size = 10)

<table>
<thead>
<tr>
<th></th>
<th>HQ</th>
<th>LQ</th>
</tr>
</thead>
<tbody>
<tr>
<td>Nodes</td>
<td>16.6%</td>
<td>15.7%</td>
</tr>
<tr>
<td>Edges</td>
<td>38.4%</td>
<td>23.6%</td>
</tr>
</tbody>
</table>
Experiments: Path quality

HAA* Path Quality (Cluster size = 15)

- High Quality Abstraction
- Low Quality Abstraction

% soft obstacles vs. % error
Experiments: Search effort

Total search effort (SO=20%, Cluster size = 15)

```
<table>
<thead>
<tr>
<th>Method</th>
</tr>
</thead>
<tbody>
<tr>
<td>Annotated A*</td>
</tr>
<tr>
<td>HAA* (High Quality Abstraction)</td>
</tr>
<tr>
<td>HAA* (Low Quality Abstraction)</td>
</tr>
</tbody>
</table>
```

- **Annotated A***
- **HAA*** (High Quality Abstraction)
- **HAA*** (Low Quality Abstraction)
Conclusion

- Presented solutions for an overlooked but important problem in single-agent pathfinding.
- Clearance value based pathfinding is simple and powerful.
- Possible to build efficient hierarchical representations of complex environments.
- Detailed empirical analysis shows method is very effective.
  - Near optimal solutions to complex problems.
  - Small memory overhead in practice.

- Future:
  - Reducing insertion effort.
  - Extend ideas to multi-agent case.
  - Apply to non-tile map encodings (like navigation meshes).