Searching Without a Heuristic: Efficient Use of Abstraction

Bradford Larsen
Ethan Burns
Wheeler Rumml
Robert C. Holte

Supported in part by the DARPA CSSG program, the NSF, and the NSERC.
Good Heuristics are Hard to Find

- Glued 15-puzzle: has unmovable tile
- Manhattan distance admissible
- Glued tile reduces effectiveness
- Natural solution: construct pattern database (PDB)

<table>
<thead>
<tr>
<th>Start</th>
<th>Goal</th>
</tr>
</thead>
<tbody>
<tr>
<td>13 8 14 3</td>
<td>1 2 3</td>
</tr>
<tr>
<td>9 5 7</td>
<td>4 5 6 7</td>
</tr>
<tr>
<td>15 1 4 10</td>
<td>8 9 10 11</td>
</tr>
<tr>
<td>12 2 6 11</td>
<td>12 13 14 15</td>
</tr>
</tbody>
</table>
Example: Glued 15-Puzzle PDB

- Abstract the puzzle by obscuring tiles
- Enumerate entire abstract state space backward from goal
- Store costs to goal in look-up table
- Use look-up table for heuristic estimates
Pattern Database Shortcomings

Disadvantages when solving one or a few instances:

- Must enumerate *entire* abstract space
  - Expensive preprocessing phase
  - Database entry for *every* abstract state
- During single search, most entries go unused
- Database not reusable
  - when goal state changes
  - when operator costs change
- One abstraction for all instances
Hierarchical A* (Holte et al. 1996)

HA* uses a hierarchy of abstractions.
Hierarchical A* (Holte et al. 1996)

Objective: find the cheapest path from $S$ to $G$. 
Hierarchical A* (Holte et al. 1996)

The yellow area represents generated states.
Hierarchical A* (Holte et al. 1996)

To generate $Q$, we need to know $h(Q)$. 
Hierarchical A* (Holte et al. 1996)

To find $h(Q)$: abstract $Q$ and search at level 1.
Hierarchical A* (Holte et al. 1996)

New objective: find cheapest path from $Q'$ to $G'$. 
Hierarchical A* (Holte et al. 1996)

To generate $R'$, we need to know $h(R')$. 

Bradford Larsen (UNH) 4 / 21 Searching Without a Heuristic
Hierarchical A* (Holte et al. 1996)

To find $h(R')$: abstract $R'$ and search at level 2.
Hierarchical A* (Holte et al. 1996)

New objective: find cheapest path from $R''$ to $G''$. 
Hierarchical A* (Holte et al. 1996)

Abstraction Level 2

Abstraction Level 1

Base Level

Use $h(n) = 0$ at the top level. Small search space!
Hierarchical A* (Holte et al. 1996)

Abstraction
Level 2

$S''$ $R''$ $G''$

Abstraction
Level 1

$S'$ $Q'$ $R'$ $G'$

Base Level

$S$ $Q$ $G$

$h^*$ values along the solution path are cached.
Hierarchical A* (Holte et al. 1996)

Use cost-to-goal at level 2 as \( h(R') \).
Hierarchical A* (Holte et al. 1996)

Eventually the search at level 1 finishes.
Hierarchical A* (Holte et al. 1996)

$h^*$ values along the solution path are cached.
Hierarchical A* (Holte et al. 1996)

$R'$ is not on the solution path.
Hierarchical A* (Holte et al. 1996)

Use cost-to-goal at level 1 as $h(Q)$. 
Hierarchical A* (Holte et al. 1996)

To generate $R$, we need to know $h(R)$. 
Hierarchical A* (Holte et al. 1996)

To find $h(R)$: abstract $R$ and search at level 1.
Hierarchical A* (Holte et al. 1996)

$h^*(R')$ is not cached: must search again.
Eventually the search at level 1 finishes.

Bradford Larsen (UNH) 4 / 21 Searching Without a Heuristic
Hierarchical A* (Holte et al. 1996)

Abstraction
Level 2

Abstraction
Level 1

Base Level

\[ h^* \text{ values along the solution path are cached.} \]
Hierarchical A* (Holte et al. 1996)

Abstraction Level 2

$S'' \rightarrow R'' \rightarrow G''$

Abstraction Level 1

$S' \rightarrow Q' \rightarrow R' \rightarrow G'$

Base Level

$S \rightarrow Q \rightarrow R \rightarrow G$

$h(R) = h^*(R')$

Use cost-to-goal at level 1 as $h(R)$. 
Hierarchical A* (Holte et al. 1996)

Search proceeds in this manner until finished.
Hierarchical IDA* (Holte et al. 2005)

- Use modified IDA* at each level
- Used effective many-to-one abstractions
- Used instance-specific abstractions
  - specialized to each problem
  - more effective than domain-specific abstraction
- Complicated caching schemes a la HA*
Advantages when solving one or a few instances:

- Abstract space visited lazily
  - No expensive preprocessing!
- Only generates cache entries that are required
- Likely faster & smaller than full PDB
- Natural to use instance-specific abstractions
  - Likely better than domain-specific abstractions
The Switchback Algorithm

Switchback uses a hierarchy of abstractions.
The Switchback Algorithm

Objective: find the cheapest path from $S$ to $G$. 
The Switchback Algorithm

Abstraction Level 2

Abstraction Level 1

Base Level

The yellow area represents generated states.
The Switchback Algorithm

To generate $Q$, we need to know $h(Q)$. 
To find $h(Q)$: search backward at level 1.
New objective: find cheapest path from $G'$ to $Q'$. 
To generate $R'$, we need to know $h(R')$. 
The Switchback Algorithm

To find $h(R')$: search forward at level 2.
The Switchback Algorithm

New objective: find cheapest path from $S''$ to $R''$. 
Use $h(n) = 0$ at the top level. Small search space!
The Switchback Algorithm

Abstraction
Level 2

$S''$ $Q''$ $R''$ $G''$

$h(R') = g(R'')$

Abstraction
Level 1

$S'$ $Q'$ $R'$ $G'$

Base Level

$S$ $Q$ $G$

Use cost to $R''$ at level 2 as $h(R')$. 
To generate $Q'$, we need to know $h(Q')$. 
The Switchback Algorithm

Abstraction Level 2

S'' Q'' R''

Abstraction Level 1

S' Q' R'

Base Level

S Q

G'' G'

Cache hit!

h(Q') = g(Q'')

To find h(Q'): use cached cost to Q''.

Bradford Larsen (UNH) 8 / 21 Searching Without a Heuristic
The Switchback Algorithm

Use cost to $Q'$ at level 1 as $h(Q)$. 

$h(Q) = g(Q')$
The Switchback Algorithm

To generate $R$, we need to know $h(R)$.
The Switchback Algorithm

To find $h(R)$: use cached cost to $R'$.
The Switchback Algorithm

Search proceeds in this manner until finished.
Properties of Switchback

- Complete: terminates with solution if one exists
- Admissible: finds optimal solutions
- Efficient: expands states at most once
- See paper for details!
Outline

Introduction

The Switchback Algorithm

Properties

Experimental Results

Conclusion
Experimental Setup

- HA*, HIDA*, and Switchback
- Several semi-standard domains
- Used custom abstraction hierarchies given by Holte et al. (2005)
- C++ implementation
- 47 GB memory limit, no time limit
Experimental Results: Glued 15-Puzzle
Averages from 100 instances

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>CPU Time (s)</th>
<th>Nodes Gen. (M)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Mean</td>
<td>Max</td>
</tr>
<tr>
<td>Switchback</td>
<td>5</td>
<td>70</td>
</tr>
<tr>
<td>HA*</td>
<td>137</td>
<td>3286</td>
</tr>
<tr>
<td>HIDA*</td>
<td>21</td>
<td>172</td>
</tr>
</tbody>
</table>
Advantage of Switchback: Glued 15-Puzzle

Increase in CPU Time (s)

Optimal Solution Cost

HA*
HIDA*
The Macro Tiles Puzzle

- Sliding tile puzzle variant
- Multiple tiles can be moved in one step
- Manhattan distance inadmissible
- Shallower solutions
- Higher branching factor
- More transpositions in state graph
Experimental Results: Macro 15-Puzzle

Averages from 100 instances

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>CPU Time (s)</th>
<th>Nodes Gen. (M)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Mean</td>
<td>Max</td>
</tr>
<tr>
<td>Switchback</td>
<td>161</td>
<td>1127</td>
</tr>
<tr>
<td>HA*</td>
<td>708</td>
<td>4647</td>
</tr>
<tr>
<td>HIDA*</td>
<td>223</td>
<td>1202</td>
</tr>
</tbody>
</table>
Advantage of Switchback: Macro 15-Puzzle

![Graph showing the advantage of Switchback in the Macro 15-Puzzle. The x-axis represents the optimal solution cost, and the y-axis represents the increase in CPU time (s). The graph compares HA* and HIDA*. The data points indicate that HIDA* generally shows an advantage over HA*.](image)
The Pancake Puzzle

- \[4 \ 3 \ 2 \ 1 \ 5 \rightarrow 1 \ 2 \ 3 \ 4 \ 5\]
- Goal: arrange numbers in order
- Can flip prefixes of the sequence
- Shallow solutions
- High branching factor
## Experimental Results: 14-Pancake Puzzle

Averages from 100 instances

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>CPU Time (s)</th>
<th>Nodes Gen. (M)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Mean</td>
<td>Max</td>
</tr>
<tr>
<td>Switchback</td>
<td>22</td>
<td>114</td>
</tr>
<tr>
<td>HA*</td>
<td>168</td>
<td>1772</td>
</tr>
<tr>
<td>HIDA*</td>
<td>89</td>
<td>401</td>
</tr>
</tbody>
</table>
Advantage of Switchback: 14-Pancake Puzzle

![Graph showing the advantage of Switchback for 14-Pancake Puzzle]

- **HA***
- **HIDA***

**Optimal Solution Cost**

**Increase in CPU Time (s)**
Summary

- Hierarchical search is an alternative to PDBs
  - Fills cache lazily
  - Can solve many problems in time to build PDB
  - Natural to use instance-specific abstractions
  - No cached heuristic values wasted

- Switchback is a better hierarchical search
  - Avoids abstract state re-expansion
  - Simple caching scheme
  - Easy to implement

- Switchback should be widely applicable
  - When predecessor states easily computed
  - When good abstraction hierarchy available
Summary

- Hierarchical search is an alternative to PDBs
  - Fills cache lazily
  - Can solve many problems in time to build PDB
  - Natural to use instance-specific abstractions
  - No cached heuristic values wasted
- **Switchback** is a better hierarchical search
  - avoids abstract state re-expansion
  - simple caching scheme
  - easy to implement
Summary

- Hierarchical search is an alternative to PDBs
  - Fills cache lazily
  - Can solve many problems in time to build PDB
  - Natural to use instance-specific abstractions
  - No cached heuristic values wasted
- Switchback is a better hierarchical search
  - Avoids abstract state re-expansion
  - Simple caching scheme
  - Easy to implement
- Switchback should be widely applicable
  - When predecessor states easily computed
  - When good abstraction hierarchy available
The University of New Hampshire

Tell your students to apply to grad school in CS at UNH!

- friendly faculty
- funding
- individual attention
- beautiful campus
- low cost of living
- easy access to Boston, White Mountains
- strong in AI, infoviz, networking
## Experimental Results: 15-Puzzle

Averages from 100 instances

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>CPU Time (s)</th>
<th></th>
<th>Nodes Gen. (M)</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Mean</td>
<td>Max</td>
<td>Mean</td>
<td>Max</td>
</tr>
<tr>
<td>Switchback</td>
<td>83</td>
<td>1422</td>
<td>65</td>
<td>713</td>
</tr>
<tr>
<td>HA*</td>
<td>831</td>
<td>12034</td>
<td>110</td>
<td>1222</td>
</tr>
<tr>
<td>HIDA*</td>
<td>194</td>
<td>2563</td>
<td>197</td>
<td>2158</td>
</tr>
</tbody>
</table>

Bradford Larsen (UNH)
Advantage of Switchback: 15-Puzzle

![Graph showing the relation between Increase in CPU Time (s) and Optimal Solution Cost for HA* and HIDA* algorithms. The x-axis represents the Optimal Solution Cost, and the y-axis represents the Increase in CPU Time (s). The graph uses a logarithmic scale for both axes, with markers indicating the performance of HA* (red) and HIDA* (green) algorithms.](image-url)