KI-Programmierung

Informed Search

Bernhard Beckert

Universität Koblenz-Landau

Winter Term 2007/2008
Outline

- Best-first search
- A* search
- Heuristics
function Tree-Search(\textit{problem}, \textit{fringe}) \textbf{returns} a solution or failure

\textit{fringe} $\leftarrow$ \textsc{Insert}(\textsc{Make-Node}(\textsc{Initial-State}[\textit{problem}]), \textit{fringe})

\textbf{loop do}

\textbf{if} \textit{fringe} is empty \textbf{then return} failure

\textit{node} $\leftarrow$ \textsc{Remove-First}(\textit{fringe})

\textbf{if} \textsc{Goal-Test}[\textit{problem}] applied to \textsc{State}(\textit{node}) succeeds \textbf{then}

\textbf{return} \textit{node}

\textbf{else}

\textit{fringe} $\leftarrow$ \textsc{Insert-All}(\textsc{Expand}(\textit{node}, \textit{problem}), \textit{fringe})

\textbf{end}

\textbf{Strategy}

Defines the order of node expansion
Best-first search

Idea

Use an evaluation function for each node (estimate of “desirability”)

Expand most desirable unexpanded node

Implementation

fringe is a queue sorted in decreasing order of desirability

Special cases

- Greedy search
- A* search
Romania with step costs in km

Straight-line distance to Bucharest

<table>
<thead>
<tr>
<th>City</th>
<th>Distance</th>
</tr>
</thead>
<tbody>
<tr>
<td>Arad</td>
<td>366</td>
</tr>
<tr>
<td>Bucharest</td>
<td>0</td>
</tr>
<tr>
<td>Craiova</td>
<td>160</td>
</tr>
<tr>
<td>Dobrota</td>
<td>242</td>
</tr>
<tr>
<td>Eforie</td>
<td>161</td>
</tr>
<tr>
<td>Fagaras</td>
<td>178</td>
</tr>
<tr>
<td>Giurgiu</td>
<td>77</td>
</tr>
<tr>
<td>Hirsova</td>
<td>151</td>
</tr>
<tr>
<td>Iasi</td>
<td>226</td>
</tr>
<tr>
<td>Lugoj</td>
<td>244</td>
</tr>
<tr>
<td>Mehadia</td>
<td>241</td>
</tr>
<tr>
<td>Neamț</td>
<td>234</td>
</tr>
<tr>
<td>Oradea</td>
<td>380</td>
</tr>
<tr>
<td>Piast</td>
<td>98</td>
</tr>
<tr>
<td>Rîmnicu Vîlcea</td>
<td>193</td>
</tr>
<tr>
<td>Sibiu</td>
<td>253</td>
</tr>
<tr>
<td>Timișoara</td>
<td>329</td>
</tr>
<tr>
<td>Urziceni</td>
<td>80</td>
</tr>
<tr>
<td>Vaslui</td>
<td>199</td>
</tr>
<tr>
<td>Zerind</td>
<td>374</td>
</tr>
</tbody>
</table>
Greedy search

Heuristic

Evaluation function

\[ h(n) = \text{estimate of cost from } n \text{ to } \text{goal} \]

Greedy search expands the node that appears to be closest to goal

Example

\[ h_{SLD}(n) = \text{straight-line distance from } n \text{ to Bucharest} \]

Note

Unlike uniform-cost search the node evaluation function has nothing to do with the nodes explored so far
Greedy search: Example Romania

Arad
366
Greedy search: Example Romania

- Sibiu: 253
- Timisoara: 329
- Zerind: 374

Diagram showing the connections between Sibiu, Timisoara, and Zerind, with Arad as the central node.
Greedy search: Example Romania
Greedy search: Example Romania

Diagram: A network diagram showing cities and distances in Romania. Cities include Arad, Sibiu, Fagaras, Oradea, Rimnicu Vilcea, Bucharest, Timisoara, and Zerind. Distances are indicated by the numbers next to the connections: Arad to Sibiu (366), Fagaras to Sibiu (380), Oradea to Sibiu (193), Rimnicu Vilcea to Bucharest (253), Bucharest to Arad (0), Timisoara to Arad (329), and Zerind to Arad (374).
Greedy search: Properties

Complete

Time

Space

Optimal
Greedy search: Properties

**Complete**  No

Can get stuck in loops

Example: Iasi to Oradea

\[
\text{Iasi} \rightarrow \text{Neamt} \rightarrow \text{Iasi} \rightarrow \text{Neamt} \rightarrow \cdots
\]

Complete in finite space with repeated-state checking

**Time**

**Space**

**Optimal**
**Greedy search: Properties**

<table>
<thead>
<tr>
<th>Property</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Complete</strong></td>
<td>No</td>
</tr>
<tr>
<td><strong>Can get stuck in loops</strong></td>
<td>Yes</td>
</tr>
<tr>
<td><strong>Example:</strong> Iasi to Oradea</td>
<td>Iasi → Neamt → Iasi → Neamt → ···</td>
</tr>
<tr>
<td><strong>Complete in finite space with repeated-state checking</strong></td>
<td>Yes</td>
</tr>
<tr>
<td><strong>Time</strong></td>
<td>$O(b^m)$</td>
</tr>
<tr>
<td><strong>Space</strong></td>
<td>Optimal</td>
</tr>
<tr>
<td><strong>Optimal</strong></td>
<td>Yes</td>
</tr>
</tbody>
</table>
Greedy search: Properties

**Complete**  No

Can get stuck in loops

Example: Iasi to Oradea

Iasi → Neamt → Iasi → Neamt → ···

Complete in finite space with repeated-state checking

**Time**  $O(b^m)$

**Space**  $O(b^m)$

**Optimal**
## Greedy search: Properties

<table>
<thead>
<tr>
<th>Property</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Complete</td>
<td>No</td>
</tr>
<tr>
<td>Can get stuck in loops</td>
<td>Yes</td>
</tr>
<tr>
<td>Example: Iasi to Oradea</td>
<td></td>
</tr>
<tr>
<td>Complete in finite space with repeated-state checking</td>
<td>Yes</td>
</tr>
</tbody>
</table>

**Time** $O(b^m)$

**Space** $O(b^m)$

**Optimal** No
Greedy search: Properties

Complete: No
Can get stuck in loops
Example: Iasi to Oradea
   Iasi → Neamt → Iasi → Neamt → ···
Complete in finite space with repeated-state checking

Time: \(O(b^m)\)
Space: \(O(b^m)\)
Optimal: No

Note
Worst-case time same as depth-first search,
Worst-case space same as breadth-first
But a good heuristic can give dramatic improvement
A* search

Idea

Avoid expanding paths that are already expensive

Evaluation function

\[ f(n) = g(n) + h(n) \]

where

\[ g(n) = \text{cost so far to reach } n \]
\[ h(n) = \text{estimated cost to goal from } n \]
\[ f(n) = \text{estimated total cost of path through } n \text{ to goal} \]
A* search: Admissibility

**Admissibility of heuristic**

$h(n)$ is admissible if

$$h(n) \leq h^*(n) \quad \text{for all } n$$

where $h^*(n)$ is the true cost from $n$ to goal
**A* search: Admissibility**

**Admissibility of heuristic**

$h(n)$ is admissible if

$$h(n) \leq h^*(n) \quad \text{for all } n$$

where $h^*(n)$ is the true cost from $n$ to goal

**Also required**

$$h(n) \geq 0 \quad \text{for all } n$$

In particular: $h(G) = 0$ for goal $G$
A* search: Admissibility

**Admissibility of heuristic**

$h(n)$ is admissible if

$$h(n) \leq h^*(n) \quad \text{for all } n$$

where $h^*(n)$ is the true cost from $n$ to goal

**Also required**

$$h(n) \geq 0 \quad \text{for all } n$$

In particular: $h(G) = 0$ for goal $G$

**Example**

Straight-line distance never overestimates the actual road distance
A* search: Admissibility

Theorem

A* search with admissible heuristic is optimal
A* search example

Arad
366=0+366
A* search example

- Sibiu: 393 = 140 + 253
- Timisoara: 447 = 118 + 329
- Zerind: 449 = 75 + 374
A* search example
A* search example
A* search example
A* search example
A* search: $f$-contours

A* gradually adds “$f$-contours” of nodes
Optimality of A* search: Proof

Suppose a suboptimal goal $G_2$ has been generated
Let $n$ be an unexpanded node on a shortest path to an optimal goal $G$

$$f(G_2) = g(G_2) \quad \text{since } h(G_2) = 0$$
$$> g(G) \quad \text{since } G_2 \text{ suboptimal}$$
$$= g(n) + h^*(n)$$
$$\geq g(n) + h(n) \quad \text{since } h \text{ is admissible}$$
$$= f(n)$$

Thus, A* never selects $G_2$ for expansion
A* search: Properties

Complete

Time

Space

Optimal
A* search: Properties

**Complete** Yes

(unless there are infinitely many nodes \(n\) with \(f(n) \leq f(G)\))

**Time**

**Space**

**Optimal**
A* search: Properties

Complete  Yes
(unless there are infinitely many nodes \( n \) with \( f(n) \leq f(G) \))

Time  Exponential in
[relative error in \( h \times \) length of solution]

Space

Optimal
# A* search: Properties

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Complete</td>
<td>Yes</td>
</tr>
<tr>
<td>(unless there are infinitely many nodes $n$ with $f(n) \leq f(G)$)</td>
<td></td>
</tr>
<tr>
<td>Time</td>
<td>Exponential in $h \times$ length of solution</td>
</tr>
<tr>
<td>Space</td>
<td>Same as time</td>
</tr>
<tr>
<td>Optimal</td>
<td></td>
</tr>
</tbody>
</table>
**A* search: Properties**

<table>
<thead>
<tr>
<th>Property</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Complete</strong></td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td>(unless there are infinitely many nodes ( n ) with ( f(n) \leq f(G) ))</td>
</tr>
<tr>
<td><strong>Time</strong></td>
<td>Exponential in</td>
</tr>
<tr>
<td></td>
<td>[relative error in ( h \times ) length of solution]</td>
</tr>
<tr>
<td><strong>Space</strong></td>
<td>Same as time</td>
</tr>
<tr>
<td><strong>Optimal</strong></td>
<td>Yes</td>
</tr>
</tbody>
</table>
A* search: Properties

**Complete**
Yes

(unless there are infinitely many nodes $n$ with $f(n) \leq f(G)$)

**Time**
Exponential in

[relative error in $h \times$ length of solution]

**Space**
Same as time

**Optimal**
Yes

**Note**

A* expands all nodes with $f(n) < C^*$

A* expands some nodes with $f(n) = C^*$

A* expands no nodes with $f(n) > C^*$
Admissible heuristics: Example 8-puzzle

Addmissible heuristics

\[ h_1(n) = \text{number of misplaced tiles} \]

\[ h_2(n) = \text{total Manhattan distance} \]

(i.e., no. of squares from desired location of each tile)
Admissible heuristics: Example 8-puzzle

Admissible heuristics

\[ h_1(n) = \text{number of misplaced tiles} \]

\[ h_2(n) = \text{total } \text{Manhattan distance} \]

(i.e., no. of squares from desired location of each tile)

In the example

\[ h_1(S) = \]

\[ h_2(S) = \]
Admissible heuristics: Example 8-puzzle

Addmissible heuristics

$h_1(n) = \text{number of misplaced tiles}$

$h_2(n) = \text{total \u00f3Manhattan distance}$

(i.e., no. of squares from desired location of each tile)

In the example

$h_1(S) = 6$

$h_2(S) =$
Admissible heuristics: Example 8-puzzle

Addmissible heuristics

\[ h_1(n) = \text{number of misplaced tiles} \]
\[ h_2(n) = \text{total Manhattan distance} \]
(i.e., no. of squares from desired location of each tile)

In the example
\[ h_1(S) = 6 \]
\[ h_2(S) = 2 + 0 + 3 + 1 + 0 + 1 + 3 + 4 = 14 \]
Dominance

**Definition**

$h_1, h_2$ two admissible heuristics

$h_2$ dominates $h_1$ if

\[ h_2(n) \geq h_1(n) \quad \text{for all } n \]
Dominance

**Definition**

\( h_1, h_2 \) two admissible heuristics

\( h_2 \) dominates \( h_1 \) if

\[
h_2(n) \geq h_1(n) \quad \text{for all } n
\]

**Theorem**

If \( h_2 \) dominates \( h_1 \), then \( h_2 \) is better for search than \( h_1 \).
Dominance: Example 8-puzzle

Typical search costs

\[ d = 14 \]
IDS: 3,473,941 nodes
\( A^*(h_1) \): 539 nodes
\( A^*(h_2) \): 113 nodes

\[ d = 24 \]
IDS: too many nodes
\( A^*(h_1) \): 39,135 nodes
\( A^*(h_2) \): 1,641 nodes

\( d \): depth of first solution
IDS: iterative deepening search
Relaxed problems

Finding good admissible heuristics is an art!

Deriving admissible heuristics

Admissible heuristics can be derived from the exact solution cost of a relaxed version of the problem.
Relaxed problems

Finding good admissible heuristics is an art!

Deriving admissible heuristics

Admissible heuristics can be derived from the exact solution cost of a relaxed version of the problem

Example

If the rules of the 8-puzzle are relaxed so that a tile can move anywhere, then we get heuristic $h_1$

If the rules are relaxed so that a tile can move to any adjacent square, then we get heuristic $h_2$
Relaxed problems

Finding good admissible heuristics is an art!

Deriving admissible heuristics

Admissible heuristics can be derived from the exact solution cost of a relaxed version of the problem

Example

If the rules of the 8-puzzle are relaxed so that a tile can move anywhere, then we get heuristic $h_1$

If the rules are relaxed so that a tile can move to any adjacent square, then we get heuristic $h_2$

Key point

The optimal solution cost of a relaxed problem is not greater than the optimal solution cost of the real problem