Informed Methods: Heuristic Search


Best-First Search: Nodes are selected for expansion based on an evaluation function, $f(n)$. Traditionally, $f$ is a cost measure.

Heuristic: Problem specific knowledge that (tries to) lead the search algorithm faster towards a goal state. Often implemented via heuristic function $h(n)$.

→ Heuristic search is an attempt to search the most promising paths first. Uses heuristics, or rules of thumb, to find the best node to expand next.

Generic Best-First Search

1. Set $L$ to be the initial node(s) representing the initial state(s).
2. If $L$ is empty, fail. Let $n$ be the node on $L$ that is “most promising” according to $f$. Remove $n$ from $L$.
3. If $n$ is a goal node, stop and return it (and the path from the initial node to $n$).
4. Otherwise, add $\text{successors}(n)$ to $L$. Return to step 2.

Greedy Best-First Search

Heuristic function $h(n)$: estimated cost from node $n$ to nearest goal node.

Greedy Search: Let $f(n) = h(n)$.

Example: 8-puzzle

<table>
<thead>
<tr>
<th>Start State</th>
<th>Goal State</th>
</tr>
</thead>
<tbody>
<tr>
<td>5 4 1 2 6 7 8</td>
<td>1 2 3 4 5</td>
</tr>
</tbody>
</table>

Example: Suboptimal Best First-Search

There exist strategies that enable optimal paths to be found without examining all possible paths.

A* Search

Idea: Use total estimated solution cost:

\[ g(n): \text{Cost of reaching node } g \text{ from initial node} \]

\[ h(n): \text{Estimated cost from node } n \text{ to nearest goal} \]

A* evaluation function: $f(n) = g(n) + h(n)$

→ $f(n)$ is estimated cost of cheapest solution through $n$. 
Comparison of Search Costs on 8-Puzzle

<table>
<thead>
<tr>
<th></th>
<th>IDS</th>
<th>A*(h₁)</th>
<th>A*(h₂)</th>
<th>A*(h*)</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>18</td>
<td>22</td>
<td>20</td>
<td>18</td>
</tr>
<tr>
<td>4</td>
<td>512</td>
<td>11</td>
<td>112</td>
<td>112</td>
</tr>
<tr>
<td>6</td>
<td>680</td>
<td>26</td>
<td>34</td>
<td>34</td>
</tr>
<tr>
<td>8</td>
<td>6184</td>
<td>39</td>
<td>28</td>
<td>28</td>
</tr>
<tr>
<td>10</td>
<td>47127</td>
<td>93</td>
<td>39</td>
<td>39</td>
</tr>
<tr>
<td>12</td>
<td>364484</td>
<td>227</td>
<td>70</td>
<td>70</td>
</tr>
<tr>
<td>14</td>
<td>3473941</td>
<td>539</td>
<td>113</td>
<td>113</td>
</tr>
<tr>
<td>16</td>
<td>-</td>
<td>1361</td>
<td>211</td>
<td>211</td>
</tr>
<tr>
<td>18</td>
<td>-</td>
<td>1856</td>
<td>363</td>
<td>363</td>
</tr>
<tr>
<td>20</td>
<td>-</td>
<td>3276</td>
<td>676</td>
<td>676</td>
</tr>
<tr>
<td>22</td>
<td>-</td>
<td>18994</td>
<td>1239</td>
<td>1239</td>
</tr>
<tr>
<td>24</td>
<td>-</td>
<td>19328</td>
<td>1848</td>
<td>1848</td>
</tr>
</tbody>
</table>

Admissibility

*h*(n) Actual cost to reach a goal from n.

**Definition:** A heuristic function $h$ is optimistic or admissible if $h(n) \leq h^*(n)$ for all nodes $n$. (*h* never overestimates the cost of reaching the goal.)

**Theorem:** If $h$ is admissible, then the $A^*$ algorithm will never return a suboptimal goal node.

8-Puzzle

1. $h_C$: number of misplaced tiles
2. $h_M$: Manhattan distance

Which one should we use? $h_C \leq h_M \leq h^*$

Comparison of Search Costs on 8-Puzzle

<table>
<thead>
<tr>
<th></th>
<th>IDS</th>
<th>A*(h₁)</th>
<th>A*(h₂)</th>
<th>Effective Branching Factor</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>IDS</td>
<td>A*(h₁)</td>
<td>A*(h₂)</td>
<td>A*(h*)</td>
</tr>
<tr>
<td>2</td>
<td>18</td>
<td>16</td>
<td>16</td>
<td>179</td>
</tr>
<tr>
<td>4</td>
<td>112</td>
<td>13</td>
<td>112</td>
<td>2.45</td>
</tr>
<tr>
<td>6</td>
<td>680</td>
<td>28</td>
<td>10</td>
<td>2.73</td>
</tr>
<tr>
<td>8</td>
<td>6184</td>
<td>39</td>
<td>25</td>
<td>2.80</td>
</tr>
<tr>
<td>10</td>
<td>47127</td>
<td>63</td>
<td>39</td>
<td>2.79</td>
</tr>
<tr>
<td>12</td>
<td>364484</td>
<td>227</td>
<td>73</td>
<td>2.78</td>
</tr>
<tr>
<td>14</td>
<td>3473941</td>
<td>539</td>
<td>113</td>
<td>2.83</td>
</tr>
<tr>
<td>16</td>
<td>-</td>
<td>1361</td>
<td>211</td>
<td>1.44</td>
</tr>
<tr>
<td>18</td>
<td>-</td>
<td>1856</td>
<td>363</td>
<td>1.45</td>
</tr>
<tr>
<td>20</td>
<td>-</td>
<td>3276</td>
<td>676</td>
<td>1.47</td>
</tr>
<tr>
<td>22</td>
<td>-</td>
<td>18994</td>
<td>1239</td>
<td>1.48</td>
</tr>
<tr>
<td>24</td>
<td>-</td>
<td>19328</td>
<td>1848</td>
<td>1.48</td>
</tr>
</tbody>
</table>
Constructing Admissible Heuristics

- Use an admissible heuristic derived from a relaxed version of the problem.
- Use information from pattern databases that store exact solutions to subproblems of the problem.
- Use inductive learning methods.

Example: A*

A \begin{array}{c}
\text{h=5} \\
1 \\
\text{B h=2}
\end{array} \\
\begin{array}{c}
A \\
\text{2}
\end{array} \\
\begin{array}{c}
B \\
\text{h=2}
\end{array} \\
\begin{array}{c}
\text{h=2} \\
1 \\
\text{C}
\end{array} \\
\begin{array}{c}
\text{h=3} \\
2 \\
\text{D}
\end{array} \\
\text{goal} \\
\text{h=0}

Proof of the optimality of A*

Assume: $h$ admissible; $f$ non-decreasing along any path.

Proof [Optimality of A*]:
Let $G$ be an optimal goal state, with path cost $f^*$. Let $G_2$ be a suboptimal goal state, with path cost $g(G_2) > f^*$. Let $n$ is a node on an optimal path to $G$.

Assume that $G_2$ is expanded before $n$:
- Because $h$ is admissible, we must have $f^* \geq f(n)$.
- If $n$ is not expanded before expanding $G_2$, we must have $f(n) \geq f(G_2)$.
- Gives us $f^* \geq f(G_2) = g(G_2)$.
- Contradiction to $G_2$ suboptimal!

Proof of the optimality of $A^*$

Lemma: If $h$ is admissible, then $f^* \geq g^* + h$ can be made non-decreasing.

1. $g$ is non-decreasing since cost positive.
2. But $h$ can be increasing, while still admissible.
   Example: Node $p$, with $f^* = 3+4 = 7$; child $n$, with $f = 4+2 = 6$.
3. But because any path through $n$ is also a path through $p$, we can see that the value 6 is meaningless, because we already know the true cost is at least 7 (because $h$ is admissible).
4. So, make $f = \max (f(p), g(n) + h(n))$

$A^*$

Optimal: yes

Complete: Unless there are infinitely many nodes with $f(n) = f^*$.
Assume locally finite:
(1) finite branching, (2) every operator costs at least $g > 0$

Complexity (time and space): Still exponential because of breadth-first nature. Unless $|h(n) - h^*(n)| \leq O(\log h^*(n))$, with $h^*$ true cost of getting to goal.

$A^*$ is optimally efficient: given the information in $h$, no other optimal search method can expand fewer nodes.

IDA*

Memory is a problem for the $A^*$ algorithms.

IDA* is like iterative deepening, but uses an $f$-cost limit rather than a depth limit.

At each iteration, the cutoff value is the smallest $f$-cost of any node that exceeded the cutoff on the previous iteration.

Each iteration uses conventional depth-first search.
**Recursive best-first search (RBFS)**

Similar to a DFS, but keeps track of the $f$-value of the best alternative path available from any ancestor of the current node.

If current node exceeds this limit, recursion unwinds back to the alternative path, replacing the $f$-value of each node along the path with the best $f$-value of its children.

(RBFS remembers the $f$-value of the best leaf in the forgotten subtree.)

**SMA***

**Simplified Memory-Bounded A* Search:**

- While memory available, proceeds just like $A^*$, expanding the best leaf.
- If memory is full, drops the **worst** leaf node - the one the highest $f$-cost; and stores this value in its parent node.

(Won't know which way to go from this node, but we will have some idea of how worthwhile it is to explore the node.)