Informed Methods: Heuristic Search

*Informed Methods* use problem-specific knowledge.

*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.

Relies on an *evaluation function* — indicates the desirability of expanding a node. E.g., *path cost*.

\[ h(n) = \text{estimated cost of the cheapest path from the state at node } n \text{ to a goal state (heuristic function)} \]

Given a list of nodes to be expanded, choose the one that the heuristic function estimates as the closest to the goal.

---

**Best-First Search**

1. Set \( L \) to be the initial node(s).

2. Let \( n \) be the node on \( L \) that is “most promising” according to the evaluation function. If \( L \) is empty, fail.

3. If \( n \) is a goal node, stop and return it (and the path from the initial node to \( n \)).

4. Otherwise, remove \( n \) from \( L \) and add all of \( n \)’s children to \( L \) (labeling each with its path from the initial node). Return to step 2.
Two Instantiations of Best-First Search

**Issue:** Best-First depends on *evaluation function.*

**Alternatives:**

**Greedy Search** minimize estimated cost to reach the goal, i.e., expand the node “closest” to the goal.

**A*** minimize total estimated path cost to reach the goal, i.e., expand the node on the “least-cost” solution path to the goal.

---

**Greedy Search**

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

1. Set \( L \) to be the initial node(s).

2. Let \( n \) be the node on \( L \) that minimizes \( h(n) \). If \( L \) is empty, fail.

3. If \( n \) is a goal node, stop and return it (and the path from the initial node to \( n \)).

4. Otherwise, remove \( n \) from \( L \) and add all of \( n \)'s children to \( L \) (labeling each with its path from the initial node). Return to step 2.
8-puzzle

Start State

![Start State Diagram]

Goal State

![Goal State Diagram]

Slide CS472 – Heuristic Search 5

Example

Slide CS472 – Heuristic Search 6
There exist strategies that enable optimal paths to be found without examining all possible paths.

**Evaluation Function for A* Algorithm**

**Goal:**

Find the shallowest, i.e. min path cost goal as quickly as possible.

$g(n)$ Cost of reaching node $n$ from start node

$h(n)$ Estimated cost from node $n$ to nearest goal

New evaluation function:

$$f(n) = g(n) + h(n)$$

$f(n)$ Estimated cost of cheapest solution through $n$
A*

\[ f(n) = \text{Estimated cost of cheapest solution through } n \]

1. Set \( L \) to be the initial node(s).

2. Let \( n \) be the node on \( L \) that minimizes to \( f(n) \). If \( L \) is empty, fail.

3. If \( n \) is a goal node, stop and return it (and the path from the initial node to \( n \)).

4. Otherwise, remove \( n \) from \( L \) and add all of \( n \)'s children to \( L \) (labeling each with its path from the initial node). Return to step 2.

Slide CS472 – Heuristic Search 9

Example

Slide CS472 – Heuristic Search 10
Admissibility

\( a(n) \)  Actual distance from a node \( n \) to a goal node.

A heuristic function \( h \) is **optimistic** or **admissible** if \( h(n) \leq a(n) \) for all nodes \( n \).

If \( h \) is **admissible**, then the A* algorithm will never return a suboptimal goal node. (**\( h \) never overestimates** the cost of reaching the goal.)
Proof of the optimality of $A^*$

Assume $f$ is monotonically increasing along any path from the root.

1. $f = g + h$; $g$ must be increasing because we’ve disallowed negative costs on operators.

2. That means that the only thing that can happen to make $f$ not monotonic along a path from the root is that our heuristic function is screwed up.

3. Situation: Node $p$, with $f = 3 + 4 = 7$; child $n$, with $f = 4 + 2 = 6$.

4. 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).

5. So, make $f = \max(f(p), g(n) + h(n))$
Proof of the optimality of $A^*$

Assume $f$ is monotonically increasing along any path from the root.

Let $G$ be an optimal goal state, with path cost $f^*$

Let $G_2$ be a suboptimal goal state, with path cost $g(G_2) > f^*$

$n$ is a leaf node on an optimal path to $G$

Because $h$ is admissible, we must have

$$f^* \geq f(n).$$

Also, if $n$ is not chosen over $G_2$, we must have

$$f(n) \geq f(G_2).$$

Gives us $f^* \geq f(G_2) = g(G_2)$. (Then $G_2$ is not suboptimal!)

Observations $A^*$

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

*Non-trivial and quite remarkable!*

Note: $A^*$ combines information of cost to get to intermediate nodes (like “uniform cost search”) with (under) estimate of cost from intermediate node to goal (“greedy search”).
A*

Optimal

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

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

See: IDA* (p. 106 R&N, “iterative deepening for A*”.)

Slide CS472 – Heuristic Search 17

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.
Each iteration uses conventional depth-first search.

Slide CS472 – Heuristic Search 18
Example: Admissible Heuristic

What if $h(n) = a(n)$?

$$f(n) = g(n) + a(n)$$

The perfect heuristic function!

Slide CS472 – Heuristic Search 19

Example: Admissible Heuristic

What if $h(n) = 0$?

$$f(n) = g(n) + h(n)$$

Slide CS472 – Heuristic Search 20
8-puzzle

1. \( h_C \) = number of misplaced tiles

2. \( h_M \) = Manhattan distance

Which one should we use?

\[ h_C \leq h_M \leq a \]

Comparison of Search Costs on 8-Puzzle

<table>
<thead>
<tr>
<th>( d )</th>
<th>IDS</th>
<th>( A^*(h_1) )</th>
<th>( A^*(h_2) )</th>
<th>IDS</th>
<th>( A^*(h_1) )</th>
<th>( A^*(h_2) )</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>10</td>
<td>6</td>
<td>6</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>112</td>
<td>13</td>
<td>12</td>
<td>2.45</td>
<td>1.79</td>
<td>1.79</td>
</tr>
<tr>
<td>6</td>
<td>680</td>
<td>20</td>
<td>18</td>
<td>2.87</td>
<td>1.48</td>
<td>1.45</td>
</tr>
<tr>
<td>8</td>
<td>6384</td>
<td>39</td>
<td>25</td>
<td>2.73</td>
<td>1.34</td>
<td>1.30</td>
</tr>
<tr>
<td>10</td>
<td>47127</td>
<td>93</td>
<td>39</td>
<td>2.80</td>
<td>1.33</td>
<td>1.24</td>
</tr>
<tr>
<td>12</td>
<td>364404</td>
<td>227</td>
<td>73</td>
<td>2.79</td>
<td>1.38</td>
<td>1.22</td>
</tr>
<tr>
<td>14</td>
<td>347394</td>
<td>539</td>
<td>113</td>
<td>2.78</td>
<td>1.42</td>
<td>1.24</td>
</tr>
<tr>
<td>16</td>
<td>–</td>
<td>1301</td>
<td>211</td>
<td>–</td>
<td>1.44</td>
<td>1.23</td>
</tr>
<tr>
<td>18</td>
<td>–</td>
<td>3056</td>
<td>363</td>
<td>–</td>
<td>1.45</td>
<td>1.25</td>
</tr>
<tr>
<td>20</td>
<td>–</td>
<td>7276</td>
<td>676</td>
<td>–</td>
<td>1.47</td>
<td>1.27</td>
</tr>
<tr>
<td>22</td>
<td>–</td>
<td>18094</td>
<td>1219</td>
<td>–</td>
<td>1.48</td>
<td>1.28</td>
</tr>
<tr>
<td>24</td>
<td>–</td>
<td>39135</td>
<td>1641</td>
<td>–</td>
<td>1.48</td>
<td>1.26</td>
</tr>
</tbody>
</table>