Heuristic (Informed) Search
(Where we try to choose smartly)

Russell and Norvig:
Chap. 3, Sect. 3.5

Slides from Jean-Claude Latombe at Stanford University
(used with permission)
Search Algorithm #2

1. INSERT(N₀, FRONTIER)

2. Repeat:
   a. If EMPTY?(FRONTIER) then return failure
   b. N = POP(FRONTIER)
   c. s = STATE(N)
   d. If GOAL?(s) then return path or goal state
   e. For every state s' in SUCCESSORS(s)
      i. Create a new node N' as a child of N
      ii. INSERT(N', FRONTIER)

Recall that the ordering of nodes in FRONTIER defines the search strategy.
Are We Smart Yet?

- So far we’ve been “blundering about in the dark” - Let’s try to be smarter!
- **Informed strategies** could find solutions more efficiently than uninformed ones
- We’ll consider a new kind of search called **Best-First Search**, which chooses nodes for expansion based on an evaluation function
Best-First Search

- It exploits **state description** to estimate how “good” each search node is.
- An **evaluation function** $f$ maps each node $N$ of the search tree to a real number:
  $$f(N) \geq 0$$
  [Traditionally, $f(N)$ is an estimated cost; so, the smaller $f(N)$, the more promising $N$]
- **Best-first search** sorts the FRONTIER in increasing $f$
  [Arbitrary order is assumed among nodes with equal $f$]
Best-First Search

- It exploits **state description** to estimate how “good” each search node is.
- An **evaluation function** $f(N) \geq 0$ maps each node $N$ of the search tree to a real number:
  
  Traditionally, $f(N)$ is an estimated cost; so, the smaller $f(N)$, the more promising $N$.

- **Best-first search** sorts the FRONTIER in increasing $f$.
  
  [Arbitrary order is assumed among nodes with equal $f$.]

“Best” does not refer to the quality of the generated path. Best-first search does not generate optimal paths in general.
How to construct $f$?

Typically, $f(N)$ estimates:

- either the cost of a solution path through $N$
  - Then $f(N) = g(N) + h(N)$, where
    - $g(N)$ is the cost of the path from the initial node to $N$
    - $h(N)$ is an estimate of the cost of a path from $N$
      to a goal node

- or the cost of a path from $N$ to a goal node
  - Then $f(N) = h(N)$  
    Greedy best-search

But there are no limitations on $f$. Any function of your choice is acceptable. But will it help the search algorithm?
How to construct $f$?

Typically, $f(N)$ estimates:

- either the cost of a solution path through $N$
  - Then $f(N) = g(N) + h(N)$, where
    - $g(N)$ is the cost of the path from the initial node to $N$
    - $h(N)$ is an estimate of the cost of a path from $N$ to a goal node
- or the cost of a path from $N$ to a goal node
  - Then $f(N) = h(N)$

But there are no limitations on $f$. Any function of your choice is acceptable. But will it help the search algorithm?
The **heuristic function** \( h(N) \geq 0 \) estimates the cost to go from \( \text{STATE}(N) \) to a goal state. Its value is independent of the current search tree; it depends only on \( \text{STATE}(N) \) and the goal test \( \text{GOAL?} \).

Example:

\[
\begin{array}{ccc}
5 & 8 \\
4 & 2 & 1 \\
7 & 3 & 6 \\
\end{array}
\quad
\begin{array}{ccc}
1 & 2 & 3 \\
4 & 5 & 6 \\
7 & 8 \\
\end{array}
\]

\( h_1(N) = \text{number of misplaced numbered tiles} = 6 \)

[Why is it an estimate of the distance to the goal?]
### Other Examples

<table>
<thead>
<tr>
<th>5</th>
<th>8</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>2</td>
</tr>
<tr>
<td>7</td>
<td>3</td>
</tr>
</tbody>
</table>

**STATE(N)**

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

**Goal state**

- \( h_1(N) \) = number of misplaced numbered tiles = 6
- \( h_2(N) \) = sum of the (Manhattan) distance of every numbered tile to its goal position
  
  \[
  = 2 + 3 + 0 + 1 + 3 + 0 + 3 + 1 = 13
  '\]
- \( h_3(N) \) = sum of permutation inversions
  
  \[
  = n_5 + n_8 + n_4 + n_2 + n_1 + n_7 + n_3 + n_6
  \]
  
  \[
  = 4 + 6 + 3 + 1 + 0 + 2 + 0 + 0
  \]
  
  \[
  = 16
  \]
8-Puzzle $f(N) = h(N) =$ number of misplaced tiles

The white tile is the empty tile
8-Puzzle

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

with \( h(N) = \text{number of misplaced tiles} \)
8-Puzzle  \( f(N) = h(N) = \sum \text{distances of tiles to goals} \)
Robot Navigation

\[ h_1(N) = \sqrt{(x_N-x_g)^2 + (y_N-y_g)^2} \]  \hspace{1cm} (L_2 or Euclidean distance)

\[ h_2(N) = |x_N-x_g| + |y_N-y_g| \]  \hspace{1cm} (L_1 or Manhattan distance)
Best-First Efficiency

Local-minimum problem

\[ f(N) = h(N) = \text{straight distance to the goal} \]
How Good is Best-First?

- If the state space is **infinite**, in general the search is **not complete**
- If the state space is **finite** and we do not discard nodes that revisit **states**, in general the search is **not complete**
- If the state space is **finite** and we **discard nodes** that revisit states, the search is **complete**, but in general is **not optimal**
Admissible Heuristic

Let $h^*(N)$ be the cost of the optimal path from $N$ to a goal node.

The heuristic function $h(N)$ is admissible if:

$$0 \leq h(N) \leq h^*(N)$$

An admissible heuristic function is always optimistic!
Admissible Heuristic

- Let $h^*(N)$ be the cost of the optimal path from $N$ to a goal node.

- The heuristic function $h(N)$ is **admissible** if:
  \[ 0 \leq h(N) \leq h^*(N) \]

- An admissible heuristic function is optimistic!
  
  - G is a goal node
  \[ h(G) = 0 \]
8-Puzzle Heuristics

- \( h_1(N) = \) number of misplaced tiles \( = 6 \)
- \( h_2(N) = \) sum of the (Manhattan) distances of every tile to its goal position
  \[ = 2 + 3 + 0 + 1 + 3 + 0 + 3 + 1 = 13 \]
- \( h_3(N) = \) sum of permutation inversions
  \[ = 4 + 6 + 3 + 1 + 0 + 2 + 0 + 0 = 16 \]
8-Puzzle Heuristics

- $h_1(N) = \text{number of misplaced tiles} = 6$
  - is **ADMISSIBLE**

- $h_2(N) = \text{sum of the (Manhattan) distances of every tile to its goal position}$
  - $= 2 + 3 + 0 + 1 + 3 + 0 + 3 + 1 = 13$
  - is **??**

- $h_3(N) = \text{sum of permutation inversions}$
  - $= 4 + 6 + 3 + 1 + 0 + 2 + 0 + 0 = 16$
  - is **??**
### 8-Puzzle Heuristics

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

#### \( h_1(N) \) = number of misplaced tiles = 6

is **Admissible**

#### \( h_2(N) \) = sum of the (Manhattan) distances of every tile to its goal position

\[
= 2 + 3 + 0 + 1 + 3 + 0 + 3 + 1 = 13
\]

is **Admissible**

#### \( h_3(N) \) = sum of permutation inversions

\[
= 4 + 6 + 3 + 1 + 0 + 2 + 0 + 0 = 16
\]

is **???
8-Puzzle Heuristics

- $h_1(N)$ = number of misplaced tiles = 6
  - is admissible

- $h_2(N)$ = sum of the (Manhattan) distances of every tile to its goal position
  - $= 2 + 3 + 0 + 1 + 3 + 0 + 3 + 1 = 13$
  - is admissible

- $h_3(N)$ = sum of permutation inversions
  - $= 4 + 6 + 3 + 1 + 0 + 2 + 0 + 0 = 16$
  - is not admissible
Robot Navigation Heuristics

Cost of one horizontal/vertical step = 1
Cost of one diagonal step = $\sqrt{2}$

$h_1(N) = \sqrt{(x_N - x_g)^2 + (y_N - y_g)^2}$ is admissible
Robot Navigation Heuristics

Cost of one horizontal/vertical step = 1
Cost of one diagonal step = $\sqrt{2}$

$h_2(N) = |x_N - x_g| + |y_N - y_g|$ is ???
Robot Navigation Heuristics

Cost of one horizontal/vertical step = 1
Cost of one diagonal step = $\sqrt{2}$

$h_2(N) = |x_N-x_g| + |y_N-y_g|$ is admissible if moving along diagonals is not allowed, and not admissible otherwise.

$h^*(l) = 4\sqrt{2}$
$h_2(l) = 8$
How to create admissible h?

- An admissible heuristic can usually be seen as the cost of an optimal solution to a relaxed problem (one obtained by removing constraints).

In robot navigation:
- The Manhattan distance corresponds to removing the obstacles.
- The Euclidean distance corresponds to removing both the obstacles and the constraint that the robot moves on a grid.
A* Search (most popular algorithm in AI)

1. \[ f(N) = g(N) + h(N) \], where:
   - \( g(N) \) = cost of best path found so far to \( N \)
   - \( h(N) \) = admissible heuristic function

2. for all arcs: \( c(N, N') \geq \varepsilon > 0 \)

3. SEARCH#2 algorithm is used

Best-first search is then called A* search
Result #1

A* is complete and optimal
[This result holds if nodes revisiting states are not discarded]
Proof (1/2)

If a solution exists, A* terminates and returns a solution

- For each node N on the fringe, 
  \[ f(N) = g(N) + h(N) \geq g(N) \geq d(N) \times \varepsilon, \]
  where \( d(N) \) is the depth of N in the tree.

- As long as A* hasn’t terminated, a node K on the fringe lies on a solution path.
Proof (1/2)

If a solution exists, A* terminates and returns a solution

- For each node $N$ on the fringe, $f(N) = g(N) + h(N) \geq g(N) \geq d(N) \times \varepsilon$, where $d(N)$ is the depth of $N$ in the tree.

- As long as A* hasn’t terminated, a node $K$ on the fringe lies on a solution path.

- Since each node expansion increases the length of one path, $K$ will eventually be selected for expansion, unless a solution is found along another path.
Proof (2/2)

Whenever $A^*$ chooses to expand a goal node, the path to this node is optimal:

- $C^* = h^*(\text{initial-node})$  
  \[ \text{[cost of the optimal solution path]} \]

- $G'$: non-optimal goal node in the fringe
  \[ f(G') = g(G') + h(G') = g(G') > C^* \]

- A node $K$ in the fringe lies on an optimal path:
  \[ f(K) = g(K) + h(K) \leq C^* \]

- So, $G'$ will not be selected for expansion.
8-Puzzle

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

with \( h(N) = \text{number of misplaced tiles} \)
Robot Navigation
Robot Navigation

\[ f(N) = h(N), \text{ with } h(N) = \text{Manhattan distance to the goal (not } A^*) \]
Robot Navigation

\[ f(N) = h(N), \text{ with } h(N) = \text{Manhattan distance to the goal} \] (not A*)

![Matrix with numbers]

\[
\begin{array}{cccccccccccc}
8 & 7 & 6 & 5 & 4 & 3 & 2 & 3 & 4 & 5 & 6 \\
7 & 5 & 4 & 3 & & & & & & 5 \\
6 & 3 & 2 & 1 & 0 & 1 & 2 & & & 4 \\
7 & 6 & & & & & & & & & \\
8 & 7 & 6 & 5 & 4 & 3 & 2 & 3 & 4 & 5 & 6 \\
\end{array}
\]
Robot Navigation

\[ f(N) = g(N) + h(N), \text{ with } h(N) = \text{ Manhattan distance to goal} \]

\[(A^*)\]

\[
\begin{array}{cccccccc}
8+3 & 7+4 & 6+3 & 5+6 & 4+7 & 3+8 & 2+9 & 3+10 & 4 & 5 & 6 \\
7+2 & 5+6 & 4+7 & 3+8 & & & & & & \\
6+1 & & 3 & 2+9 & 1+10 & 0+11 & 1 & 2 & & & 4 \\
7+0 & 6+1 & & & & & & & & & 5 \\
8+1 & 7+2 & 6+3 & 5+4 & 4+5 & 3+6 & 2+7 & 3+8 & 4 & 5 & 6 \\
\end{array}
\]
Best-First Search

- An **evaluation function** \( f \) maps each node \( N \) of the search tree to a real number:
  \[
  f(N) \geq 0
  \]

- **Best-first search** sorts the FRINGE in increasing \( f \)
**A* Search** (most popular algorithm in AI)

1. \( f(N) = g(N) + h(N) \), where:
   - \( g(N) \) = cost of best path found so far to \( N \)
   - \( h(N) \) = admissible heuristic function

2. for all arcs: \( c(N,N') \geq \varepsilon > 0 \)

3. SEARCH#2 algorithm is used

- Best-first search is then called A* search
Result #1

A* is complete and optimal
[This result holds if nodes revisiting states are not discarded]