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

R&N: Chap. 4, Sect. 4.1–3

Slides from Jean-Claude Latombe at Stanford University
(used with permission)

Search Algorithm #2

```
SEARCH2
1. INSERT(initial-node,FRINGE)

2. Repeat:
   a. If empty(FRINGE) then return failure
   b. N = REMOVE(FRINGE)
   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 node N' as a successor of N
      ii. INSERT(N',FRINGE)
```

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 instance of the Tree-Search/Graph-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 FRINGE in increasing $f$.
  - (Arbitrary order is assumed among nodes with equal $f$).

**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?
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?

---

**Heuristic Function**

- The heuristic function $h(N) \geq 0$ estimates the cost to go from STATE($N$) to a goal state.

  Its value is independent of the current search tree: it depends only on STATE($N$) and the goal test GOAL?

- Example:

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

  STATE($N$) is number of misplaced numbered tiles = 6

  (Why is it an estimate of the distance to the goal?)

---

**Other Examples**

- $h_1(N) = \text{number of misplaced numbered tiles} = 6$.
- $h_2(N) = \text{sum of the (Manhattan) distance of every numbered tile to its goal position} = 2 + 3 + 0 + 1 + 3 + 0 + 3 + 1 = 13$.
- $h_3(N) = \text{sum of permutation inversions} = n_5 + n_8 + n_4 + n_2 + n_1 + n_7 + n_3 + n_6 = 4 + 6 + 3 + 1 + 0 + 2 + 0 + 0 = 16$.
The white tile is the empty tile.

8-Puzzle

\[ f(N) = h(N) = \text{number of misplaced numbered tiles} \]

8-Puzzle

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

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

8-Puzzle

\[ f(N) = h(N) = \sum \text{distances of numbered tiles to their goals} \]
Robot Navigation

Best-First Efficiency

Local-minimum problem

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.

G is a goal node $\Rightarrow h(G) = 0$

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
  \[ = 4 + 6 + 3 + 1 + 0 + 2 + 0 + 0 = 16 \]
- $h_3(N)$ is not admissible.
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.
  $= 2 \times 3 + 0 + 1 \times 3 + 0 \times 3 + 1 = 13$ is admissible
- $h_3(N)$ = sum of permutation inversions
  $= 4 + 6 + 3 + 1 + 0 + 2 + 0 + 0 = 16$ is not admissible
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

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

$h^*(I) = 4\sqrt{2}$
h_2(I) = 8
How to create an 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.

- More on this topic later.

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

\( \Rightarrow \) 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)

1) If a solution exists, A* terminates and returns a solution.

- For each node N on the fringe, \( f(N) = g(N) + h(N) \geq g(N) \geq d(N) \times \epsilon \), 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 (2/2)

2) Whenever A* chooses to expand a goal node, the path to this node is optimal.

- \( C^* \): optimal cost of the solution path

- \( G' \): non-optimal goal node in the fringe, \( f(G') = g(G') + h(G') = g(G') \geq C^* \)

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.

Optimistic estimate
When a problem has no solution, A* runs for ever if the state space is infinite or states can be revisited an arbitrary number of times. In other cases, it may take a huge amount of time to terminate.

- So, in practice, A* is given a time limit. If it has not found a solution within this limit, it stops. Then there is no way to know if the problem has no solution, or if more time was needed to find it.

- When AI systems are "small" and solving a single search problem at a time, this is not too much of a concern.

- When AI systems become larger, they solve many search problems concurrently, some with no solution. What should be the time limit for each of them?

### 8-Puzzle

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

with \( h(N) \): number of misplaced tiles

### Robot Navigation
Robot Navigation

\( f(N) = h(N), \) with \( h(N) = \) Manhattan distance to the goal 
(not A*)

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

Robot Navigation

\( f(N) = h(N), \) with \( h(N) = \) Manhattan distance to the goal 
(not A*)

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

Robot Navigation

\( f(N) = g(N)+h(N), \) with \( h(N) = \) Manhattan distance to goal 
(A*):

\[
\begin{array}{cccccc}
8 & 3 & 7 & 4 & 6 & 3 \\
7 & 2 & 5 & 6 & 4 & 3 \\
6 & 1 & 3 & 2 & 9 & 3 \\
7 & 0 & 6 & 1 & 3 & 2 \\
8 & 1 & 7 & 2 & 6 & 5 \\
\end{array}
\]
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

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]