Heuristic Search

- Best-First
- A*
- Heuristic Functions

Some material from: D Lin, J You, JC Latombe
Search Overview

- Introduction to Search
- Blind Search Techniques
- Heuristic Search Techniques
  - Best-First
  - A*
  - Heuristic Functions
    - Stochastic Algorithms
    - Game Playing search
    - Constraint Satisfaction Problems
Heuristic Search

- “Blind” methods only know Goal / NonGoal
- Often ∃ other problem-specific knowledge that can guide search:
  - **Heuristic fn** $h(n): \text{Nodes} \rightarrow \mathbb{R}$
    - estimate of distance from $n$ to a goal
      - Eg: straight line on map, or “Manhattan distance”, or …
  - Use: Given list of nodes to expand, choose node $n$ with min'l $h(.)$
Heuristic Function

- $h(n)$ estimates cost of cheapest path from node $n$ to goal node
- Example: 8-puzzle

$$h_1(n) = \text{number of misplaced tiles} = 6$$
Heuristic Function

- $h(n)$ estimates cost of cheapest path from node $n$ to goal node.
- Example: 8-puzzle

$h_1(n) = $ number of misplaced tiles
  
  \[
  \text{h}_1(n) = 6
  \]

$h_2(n) = $ sum of the distances of every tile to its goal position
  
  \[
  \text{h}_2(n) = 3 + 1 + 3 + 0 + 2 + 1 + 0 + 3 = 13
  \]
Greedy Best-First Search

BestF_Search( start, operations, is_goal ): path
L := makeList( start )
loop
    \[ n := \arg \min_{n_i \in L} h(n_i) \]
    ;; "most promising" node in L according to h(.)
    if [ is_goal( n ) ]
        return( n )
    S := successors( n, operators )
    L := insert( S, L )
until L is empty
return( failure )

Idea: choose frontier node with smallest h-value
ie, "closest to goal"
Can also return "path from start to \( n \)"
... by identifying each node with path
Robot Navigation
Robot Navigation

Edmonton

\[ h(n) = \text{Manhattan distance to the goal} \]

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

\( h_{\text{SLD}}(n) \) is straight-line distance from \( n \) to goal (Bucharest)
Best First
Best First
BestFirst is SubOptimal

- $h_{\text{SLD}}$ finds path:
  
  \[ \text{Arad} \rightarrow \text{Sibiu} \rightarrow \text{Fagaras} \rightarrow \text{Bucharest} \]
  
  \[ \text{(Cost} = 140 + 99 + 211 = 450) \]

- Not optimal!
  
  \[ \text{C}( \text{Arad} \rightarrow \text{Sibiu} \rightarrow \text{Rimnicu} \rightarrow \text{Pitesti} \rightarrow \text{Bucharest}) \]
  
  \[ = 140 + 80 + 97 + 101 = 418 \]
  
  \[ < h_{\text{SLD}}'s \text{ solution!} \]

- BestFirst is greedy:
  
  takes BIGGEST step each time...
BestFirst can Loop

- Consider: **Iasi → Fagaras**
  - $h_{SLD}$ suggests: **Iasi → Neamt**
- Worse: Unless search alg detects repeated states, BestFirst will oscillate:
  - **Iasi → Neamț → Iasi → Neamț → ...**
- Loops are a real problem...
Properties of Greedy Best-First Search

- If state space is finite and we avoid repeated states, THEN Best-First search is complete, but in general is not optimal.
- If state space is finite and we do not avoid repeated states, THEN Best-First search is not complete.
- If the state space is infinite, THEN Best-First search is not complete.
Analysis of Greedy BestFirst

- **Complete?** No
  ...can go down $\infty$-path (oscillate)

- **Optimal?** No
  ... may not find shortest path

- **Time:** $O(b^m)$
- **Space:** $O(b^m)$
  (if $h(.) \equiv 0$, could examine entire space)

- Worst of both worlds
  - $\approx$DFS: too greedy!
  - $\approx$BFS: too much space!
A* Search

- Find cheapest path, quickly
- Consider both:
  - Path from start to $n$:
    - $g(n) = \text{cost of path found to } n$
  - Path from $n$ to goal (est.):
    - $h(n) = \text{estimate of cost from } n \text{ to a goal}$
  - $f(n) = g(n) + h(n)$
- est of cost of path from start to goal, via $n$
A* Search, con’t

- A* selects node with min’l $f(n)$
  - ...ie, node with lowest estimated distance from start to goal, constrained to go via that node

- ... mix of \{ lowest-cost-first, best-first \} searches!
Example of A* 

Note: Finds Optimal Path!

- A* expands
  - Rimnicu \((f = (140+80)+193 = 413)\) over
    - Faragas \((f = (140+99)+178 = 417)\)
- Why?
  - Fagaras is closer to Bucharest (than Rimnicu) but
    path taken to get to Fagaras is not as efficient at getting close to Bucharest ...
    as Rimnicu
$f(n) = g(n) + h(n)$, with $h(n)$ = Manhattan distance to goal

<table>
<thead>
<tr>
<th></th>
<th>3+10</th>
<th>4</th>
<th>5</th>
<th>6</th>
</tr>
</thead>
<tbody>
<tr>
<td>8+3</td>
<td>7+4</td>
<td>6+3</td>
<td>5+6</td>
<td>4+7</td>
</tr>
<tr>
<td>7+2</td>
<td>5+6</td>
<td>4+7</td>
<td>3+8</td>
<td></td>
</tr>
<tr>
<td>6+1</td>
<td></td>
<td></td>
<td>2+9</td>
<td>1+10</td>
</tr>
<tr>
<td>7+0</td>
<td>6+1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8+1</td>
<td>7+2</td>
<td>6+3</td>
<td>5+4</td>
<td>4+5</td>
</tr>
</tbody>
</table>
How A* Searches

- Contour-lines of “equal-f values”
- A* expands nodes with increasing $f(n)$ values
- If use $h(.) = 0$ (UniformCost) get Circles
  $\Rightarrow$ more nodes expanded (in general)!
Admissible heuristic

- $h^*(n) = \text{cost of optimal path from } n \text{ to a goal node}$

- Heuristic $h(n)$ is admissible if:
  \[ 0 \leq h(n) \leq h^*(n) \]

- Admissible heuristic is always optimistic

- True for
  - Straight Line [map traversal]
  - Manhattan distances [8-puzzle]
  - Number of attacking queens [n-queens] [place all queens, then move]

$\Rightarrow f(.) \text{ is under-estimate}$
Heuristics for 8-Puzzle

- $h_1(n) = \text{number of misplaced tiles}$ ... $= 6$
- $h_2(n) = \text{sum of distances of each tile to goal posn}$ ... $= 13$
- $h_3(n) = h_1(n) + 3 \times h_2(n)$ ... $= 45$
- $h_4(n) = 0$ ... $= 0$
- $h_5(n) = \min\{ h_1(n), h_2(n) \}$ ... $= 6$
f(n) is monotonic

- \( f(n) \leq f(n') \), as from-S-to-E-via-n is less constrained than from-S-to-E-via-n-n'

\[ g(n) \leq h(n) \leq h^*(n) \]
Monotonic $f(.)$

- $f(.)$ is "monotonic" $\equiv f(\text{Successor}(n)) \geq f(n)$

- Always true if $|h(n) - h(m)| \leq d(n,m)$
  ... $d(n,m)$ is distance from $n$ to $m$

- If true: first path that A* finds to node, is always shortest

- If $f(.)$ not monotonic, can modify to be:

  Eg, $n' \in \text{Successor}(n)$
  $f(n) = g(n)+h(n) = 3+4 = 7$
  $f(n') = g(n')+h(n') = 4+2 = 6$

  But... any path through $n'$ is also path through $n$,
  so $f(n)$ must be $\geq 7$
  $\Rightarrow$ should reset $f(n') = 7$

  $\Rightarrow$ use $f(n') = \max\{f(n), g(n')+h(n')\}$

  Called "path-max equation”
  ... ignores misleading numbers in heuristic
A* is OPTIMAL

Thrm: A* always returns optimal solution if
- ∃ solution
- h(n) is under-estimate

PROOF:
Let G be optimal goal, with \( f(G) = g(G) = f \)
G_2 be suboptimal goal, with \( f(G_2) = g(G_2) > f \)
If A* returns G_2 ⇒
G_2 is chosen over \( n \), where \( n \) is node on optimal path to G
This only happens if \( f(G_2) \leq f(n) \)
As \( f \) is monotonically increasing along every path,
⇒ \( f = f(G) \geq f(n) \)
Hence, \( f \geq f(G_2) \) ... ie, if \( g(G) \geq g(G_2) \)
... contradicting claim that G_2 is suboptimal! [ ]
Properties of A*

**A* is Optimally Efficient**
Given the information in \( h(.) \), no other optimal search method can expand fewer nodes. Non-trivial and quite remarkable!

**A* is Complete**
... unless there are \( \infty \) nodes w/ \( f(n) < f^* \)
- **A* is Complete**
  if branching factor is finite & arc costs bounded above zero
  \( (\exists \varepsilon > 0 \text{ s.t. } c(a_i) \geq \varepsilon) \)

**Time/ Space Complexity:**
Still exponential as \( \approx \) breadth-first.
... unless \( |h(n) - h(n^*)| \leq O(\log(h(n^*)) \)
\( h(n^*) = \) true cost of getting from \( n \) to goal
8-Puzzle

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

with \( h(n) = \) number of misplaced tiles
Robot navigation

\[ f(n) = g(n) + h(n), \text{ with } h(n) = \text{ straight-line distance from } n \text{ to goal} \]

Cost of one horizontal/vertical step = 1
Cost of one diagonal step = \( \sqrt{2} \)
A* Topics

- Which heuristic?
- Avoiding Loops
- Iterative Deepening A*
Heuristics for 8-Puzzle

<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>

\[ n \]

<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>

\[ \text{goal} \]

Admissible??

- \[ h_1(n) = \text{number of misplaced tiles} \ldots = 6 \]
- \[ h_2(n) = \text{sum of distances of each tile to goal posn} \ldots = 13 \]
- \[ h_3(n) = h_1(n) + 3 \times h_2(n) \ldots = 45 \]
- \[ h_4(n) = 0 \ldots = 0 \]

Many admissible heuristics ... which to use??
Importance of $h(.)$

- $A^*(h_i)$ expands all nodes with $f(n) = g(n) + h_i(n) < f^*$
  ... ie, with $h_i(n) < f^* - g(n)$

- $h_1(n) < h_2(n) \Rightarrow$
  If $A^*(h_2)$ expands $n$, then $A^*(h_1)$ expands $n$!
  ... but not vice versa

- $A^*(h_2)$ might expand FEWER nodes

- So LARGER $h_i()$ means fewer $n$'s expanded!
Importance of $h(.)$

- LARGER $h_i()$ means fewer $n$'s expanded!

- As $h_C \leq h_M \leq h^*$, prefer $h_M$!

- Gen'l:
  Want largest $h()$ that is under-estimate
Effect of Different Heuristic Functions

"Effective Branching Factor" \( b \) is solution to

\[
N = 1 + (b^*) + (b^*)^2 + (b^*)^3 + \ldots + (b^*)^d
\]

where \( N \) is # of nodes searched

\( d \) is solution depth

<table>
<thead>
<tr>
<th>( d )</th>
<th>IDS</th>
<th>( A^*(h_C) )</th>
<th>( A^*(h_M) )</th>
<th>IDS</th>
<th>( A^*(h_C) )</th>
<th>( A^*(h_M) )</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>10</td>
<td>6</td>
<td>6</td>
<td>2.45</td>
<td>1.79</td>
<td>1.79</td>
</tr>
<tr>
<td>4</td>
<td>112</td>
<td>13</td>
<td>12</td>
<td>2.87</td>
<td>1.48</td>
<td>1.45</td>
</tr>
<tr>
<td>6</td>
<td>680</td>
<td>20</td>
<td>18</td>
<td>2.73</td>
<td>1.34</td>
<td>1.30</td>
</tr>
<tr>
<td>8</td>
<td>6384</td>
<td>39</td>
<td>25</td>
<td>2.80</td>
<td>1.33</td>
<td>1.24</td>
</tr>
<tr>
<td>10</td>
<td>47127</td>
<td>93</td>
<td>39</td>
<td>2.79</td>
<td>1.38</td>
<td>1.22</td>
</tr>
<tr>
<td>12</td>
<td>364404</td>
<td>227</td>
<td>73</td>
<td>2.78</td>
<td>1.42</td>
<td>1.24</td>
</tr>
<tr>
<td>14</td>
<td>3473941</td>
<td>539</td>
<td>113</td>
<td>2.83</td>
<td>1.44</td>
<td>1.23</td>
</tr>
<tr>
<td>16</td>
<td>–</td>
<td>1301</td>
<td>211</td>
<td>–</td>
<td>1.45</td>
<td>1.25</td>
</tr>
<tr>
<td>18</td>
<td>–</td>
<td>3056</td>
<td>363</td>
<td>–</td>
<td>1.46</td>
<td>1.26</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>
About Heuristics

- Heuristics are intended to orient the search along promising paths.
- Time spent evaluating heuristic function must be recovered by a better search.
  - “Perfect heuristic function” would mean NO search!
- Deciding which node to expand $\equiv$ “meta-reasoning”
- Heuristics...
  - may not always look like numbers
  - may involve large amount of knowledge
Inventing Heuristics

- Solve problem, then compute backwards...
- If \( \{h_1, \ldots, h_k\} \) all underestimates, use \( h_{\text{max}}(n) = \max \{ h_i(n) \} \)
  (Still an under-estimate, but larger ... )
- **Relaxation:**
  Consider SIMPLER version of problem. As heuristic, use
  - "exact answer to approx problem"
Inventing Heuristics

**Original:** Can move tile from sq A to sq B if
... A is adjacent to B and B is blank.

**Relaxed version #1:**
- Ie, can TELEPORT tile to blank
  ➞ # of misplaced tiles $h_C$

**Relaxed version #2:**
- Ie, can walk over non-blank tile
  ➞ Manhattan distance $h_M$
Other Tricks

- Patterns Databases
- Learning from part experiences
Avoiding Repeated States in A*

If the heuristic $h(.)$ is monotonic, then:

- Let $CLOSED$ be the list of states associated with expanded nodes.
- When a new node $n$ is generated:
  - If its state is in $CLOSED$, then discard $n$.
  - If it has the same state as another node in the fringe, then discard the node with the largest $f(.)$. 
Complexity of Consistent A*:

- $s = |S|$  
  - size of the state space

- $r = |A|$  
  - max number of states that can be reached by applying any operator, from any state

Assume test if state $s \in \text{CLOSED}$ is $O(1)$

$\Rightarrow$ Time complexity of A*: $O(sr \log s)$
Iterative Deepening A* (IDA*)

- Use $f(n) = g(n) + h(n)$ with admissible, consistent $h(.)$
- Each iteration is depth-first with cutoff on the value of $f(n)$ of expanded nodes

AIXploratorium
http://www.cs.ualberta.ca/~aixplore
8-Puzzle

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

with \( h(n) = \) number of misplaced tiles

Cutoff=4
8-Puzzle

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

with \( h(n) = \) number of misplaced tiles
8-Puzzle

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

with \( h(n) = \) number of misplaced tiles

Cutoff=4
8-Puzzle

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

with \( h(n) \) = number of misplaced tiles

Cutoff=4
8-Puzzle

\[ f(n) = g(n) + h(n) \]
with \( h(n) = \) number of misplaced tiles

No more nodes to expand with Cutoff = 4
Now consider Cutoff = 5
8-Puzzle

\[ f(n) = g(n) + h(n) \]
with \( h(n) = \) number of misplaced tiles
8-Puzzle

\[ f(n) = g(n) + h(n) \]
with \( h(n) = \) number of misplaced tiles

Cutoff = 5
8-Puzzle

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

with \( h(n) \) = number of misplaced tiles

\[ \text{Cutoff=5} \]
8-Puzzle

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

with \( h(n) = \) number of misplaced tiles
8-Puzzle

\[ f(n) = g(n) + h(n) \]
with \( h(n) = \) number of misplaced tiles

Cutoff = 5
8-Puzzle

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

with \( h(n) = \) number of misplaced tiles

Cutoff=5
8-Puzzle

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

with \( h(n) = \) number of misplaced tiles

Cutoff = 5
Summary

- Heuristic function
- Greedy Best-first search
- Admissible heuristic
- A* is complete and optimal
  - Optimally efficient!
- Consistent heuristic and repeated states
- Inventing Heuristics
- IDA*