Heuristic Search

Best-First Search

- use an evaluation function \( f(n) \) for each node
- estimates "desirability"
- expand most desirable node in fringe
- enqueueing function maintains fringe in order of \( f(n) \) — smallest (lowest cost) first
- two approaches: Greedy and A* 

Romania

- map of roads between cities with distances (as used in uninformed search)
- straight-line distances to Bucharest from each city (as the crow flies)
  - Arad 366, Bucharest 0, Craiova 160, etc...
Greedy Best-First Search

- we introduce \( h(n) \): a heuristic function that estimates the cost from \( n \) to goal
- evaluation function \( f(n) = h(n) \)
- \( h(n) \) = straight line distance from state(\( n \)) to Bucharest
- greedy best-first search expands the node that appears to be closest to the goal
Greedy Best-First Search

- Sibiu 253
  - Arad 366
    - Timisoara 329
      - Oradea 380
      - Fagaras 176
      - Rimnicu 193
    - Zerind 274

Greedy Best-First Search

- Sibiu 253
  - Arad 366
    - Timisoara 329
      - Oradea 380
      - Fagaras 176
      - Rimnicu 193
    - Zerind 274
Greedy Best-First Search

Properties of Greedy Search

- complete? no (if tree search) – can get stuck in loops; yes if repeated nodes are eliminated (graph search)
- time? $O(b^m)$, but a good heuristic dramatically improves performance
- space? $O(b^m)$, keeps all nodes in memory
- optimal? no, greedy search is like heuristic depth-first
A* Search

- Avoid expanding paths that are already expensive
- \( f(n) = g(n) + h(n) \)
- \( g(n) \) = path cost from initial to \( n \)
- \( h(n) \) = estimated cost from \( n \) to goal
- \( f(n) \) = estimated cost from initial to goal through \( n \)
A* Search

A

Sini
395x78 155

Zarno
447p+5 351

Teramo
6/2 18 928

A

Sini
395x78 155

Zarno
447p+5 351

Teramo
6/2 18 928

A

Sini
395x78 155

Zarno
447p+5 351

Teramo
6/2 18 928
A* Search

A

A

A

A

A

A

A

A
Admissible Heuristics

- A heuristic $h(n)$ is admissible if for every node $n$, $h(n) \leq h^*(n)$, where $h^*(n)$ is the true cost to reach the goal from $n$
- An admissible heuristic thus never overestimates the cost to reach the goal — that is, it must be optimistic
- For example, straight line distance is admissible
- Theorem: if $h(n)$ is admissible, $A^*$ using tree-search is optimal

Proof of Optimality of $A^*$

- Suppose some suboptimal goal $G_2$ has been generated (in the fringe). Let $n$ be an unexpanded node in the fringe such that $n$ is on the shortest path to an optimal goal $G$
- $f(n) = g(n)$ since $h(G) = 0$
- $g(n) < g(G)$ since $G$ is suboptimal
- $f(n) = g(n)$ since $h(G) = 0$
- $f(G) > f(n)$ from above
- $h(n) \leq h^*(n)$ since $h$ is admissible
- $g(n) + h(n) \leq g(G) + h^*(n)$
- $f(n) \geq f(G)$
- Hence $f(G) > f(n)$ so $A^*$ will never select $G_2$ for expansion

A* Tree vs Graph Search

- $A^*$ with admissible $h$ is optimal for tree search
- Not so for graph search — $A^*$ may discards repeated states even if cheaper routes to them (i.e. $g(n)$) are found
- Fix in two ways
  - Modify graph search to check and replace repeated states with cheaper alternatives
  - Leave graph search as is, but insist on consistent $h(n)$
Consistent Heuristics

- A heuristic is consistent if for every node n, every successor n’ of n generated by action a, h(n) <= c(n,a,n’) + h(n’)
- Consistent heuristics satisfy triangle inequality
- Difficult to concoct an admissible yet inconsistent heuristic
- If h is consistent, f(n) = g(n) + h(n)
- g(n) = c(n,a,n’) + h(n’) >= g(n) + h(n)
- f(n) = g(n) + h(n)
- That is, f(n) is non-decreasing along any path
- Theorem: if h(n) is consistent, A* using graph-search is optimal

Properties of A*

- Complete? Yes (unless there are infinitely many nodes with f <= f(G))
- Time? Exponential
- Space? Keeps all nodes in memory
- Optimal? Yes

Admissible Heuristics

- For example, the 8-puzzle
- h₁(n) = number of misplaced tiles
- h₂(n) = total Manhattan distance

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

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

h₁(S) = 
h₂(S) =
**Admissible Heuristics**

- for example, the 8-puzzle
- $h_1(n) =$ number of misplaced tiles
- $h_2(n) =$ total manhattan distance

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

$h_1(S) = 8$

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

$h_2(S) =$

**Admissible Heuristics**

- for example, the 8-puzzle
- $h_1(n) =$ number of misplaced tiles
- $h_2(n) =$ total manhattan distance

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

$h_1(S) = 8$

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

$h_2(S) = 3+1+2+2+2+3+3+2 = 18$

**Dominance**

- for admissible heuristics $h_1$ and $h_2$, $h_2$ dominates $h_1$ if $h_2(n) \geq h_1(n)$ for all $n$
- typical time complexities (number of expanded nodes) for 8-puzzle
  - $d = 12$
    - IDS: 3,644,035
    - $A^*(h_1) = 227$
    - $A^*(h_2) = 73$
  - $d = 24$
    - IDS: too many
    - $A^*(h_1) = 39,131$
    - $A^*(h_2) = 1,641$
Relaxation

- Finding heuristics systematically by relaxing a problem
- A problem with fewer restrictions on actions is a relaxed problem
- The cost of an optimal solution to the relaxed problem is an admissible heuristic for the original problem
- For 8-puzzle, allowing tiles to move anywhere generates h_1 and allowing tiles to move to any adjacent square generates h_2
- For Roman's problem, straight line distance is a relaxation generating its heuristic