CS W4701
Artificial Intelligence

Fall 2013
Chapter 3 Part 4:
Informed Search

Jonathan Voris
(based on slides by Sal Stolfo)
Announcements

• Midterm: Thursday **October 24** 2:40-3:55 PM in Pupin 301
• Final: Thursday **December 5th** 2:40-3:55 PM in Pupin 301
# Summary of Uninformed Search Algorithms

<table>
<thead>
<tr>
<th>Criterion</th>
<th>Breadth-First</th>
<th>Uniform-Cost</th>
<th>Depth-First</th>
<th>Depth-Limited</th>
<th>Iterative Deepening</th>
</tr>
</thead>
<tbody>
<tr>
<td>Complete?</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Time</td>
<td>$O(b^{d+1})$</td>
<td>$O(b^{C*}/\epsilon)$</td>
<td>$O(b^m)$</td>
<td>$O(b^l)$</td>
<td>$O(b^d)$</td>
</tr>
<tr>
<td>Space</td>
<td>$O(b^{d+1})$</td>
<td>$O(b^{C*}/\epsilon)$</td>
<td>$O(bm)$</td>
<td>$O(bl)$</td>
<td>$O(bd)$</td>
</tr>
<tr>
<td>Optimal?</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
</tr>
</tbody>
</table>
Outline

• Best-first search
  – Greedy best-first search
  – A* search

• Heuristics
Recap: Tree Search

• Core concept:
  – Exploration of state space by generating successors of already-explored states (a.k.a. expanding states)

```python
function TREE-SEARCH(problem, strategy) returns a solution, or failure
  initialize the search tree using the initial state of problem
  loop do
    if there are no candidates for expansion then return failure
    choose a leaf node for expansion according to strategy
    if the node contains a goal state then return the corresponding solution
    else expand the node and add the resulting nodes to the search tree
```
Smarter Search

- Uninformed search parameter selection: crude application of domain knowledge
  - Node ordering
  - Search strategy

- But still limited to:
  - Expand successors
  - Reached goal?

- What if we had a way to assess relative state quality?
Goal-based Agents

- State
- What the world is like now
- What it will be like if I do action A
- What action I should do now
- Actuators
- Sensors
- Environment
- How the world evolves
- What my actions do
- Goals
Utility-based Agents
Best-first Search

• Recall uniform-cost search
  – Expand node with lowest path cost function value \( g(n) \)
• New idea: use an **evaluation function** \( f(n) \) for each node
  – Estimate of "desirability"
  – Expand most desirable unexpanded node
• Implementation: Order the nodes in fringe in decreasing order of desirability
• Special cases:
  – Greedy best-first search
  – A* search
Heuristics

- f(n) presents a chicken and egg problem
  - Need to know which state is closest to goal
  - If we know that, what is the point of the agent?
- Instead, utilize domain specific knowledge to estimate preferable states
- Known as a **heuristic**
  - Greek word *heuriskein*: “To discover”
  - Learning aid
  - Feedback that facilitates self learning
- h(goal) = 0 always
Greedy Best-first Search

• Evaluation function \( f(n) = h(n) \) (heuristic)
  – Estimate of cost from \( n \) to \( \text{goal} \)
• e.g., \( h_{\text{SLD}}(n) = \) straight-line distance from \( n \) to Bucharest
• Greedy best-first search expands the node that \textit{appears} to be closest to \( \text{goal} \)
Romania with Step Costs in km

<table>
<thead>
<tr>
<th>Location</th>
<th>Step Costs in km</th>
</tr>
</thead>
<tbody>
<tr>
<td>Arad</td>
<td>366</td>
</tr>
<tr>
<td>Bucharest</td>
<td>0</td>
</tr>
<tr>
<td>Craiova</td>
<td>160</td>
</tr>
<tr>
<td>Dobretta</td>
<td>242</td>
</tr>
<tr>
<td>Eforie</td>
<td>161</td>
</tr>
<tr>
<td>Fagaras</td>
<td>176</td>
</tr>
<tr>
<td>Giurgiu</td>
<td>77</td>
</tr>
<tr>
<td>Hirsova</td>
<td>151</td>
</tr>
<tr>
<td>Iasi</td>
<td>226</td>
</tr>
<tr>
<td>Lugoj</td>
<td>244</td>
</tr>
<tr>
<td>Mehadia</td>
<td>241</td>
</tr>
<tr>
<td>Neamt</td>
<td>234</td>
</tr>
<tr>
<td>Oradea</td>
<td>380</td>
</tr>
<tr>
<td>Pitesti</td>
<td>10</td>
</tr>
<tr>
<td>Rimnicu Vilcea</td>
<td>193</td>
</tr>
<tr>
<td>Sibiu</td>
<td>253</td>
</tr>
<tr>
<td>Timisoara</td>
<td>329</td>
</tr>
<tr>
<td>Urzicieni</td>
<td>80</td>
</tr>
<tr>
<td>Vaslui</td>
<td>199</td>
</tr>
<tr>
<td>Zerind</td>
<td>374</td>
</tr>
</tbody>
</table>
Greedy Best-first Search Example
Greedy Best-first Search Example
Greedy Best-first Search Example
Greedy Best-first Search Example
Properties of Greedy Best-first Search

- **Complete?**
  - No – Can get stuck in loops, e.g., Iasi $\rightarrow$ Neamt $\rightarrow$ Iasi $\rightarrow$ Neamt $\rightarrow$ …

- **Time?**
  - $O(b^m)$, but a good heuristic can give dramatic improvement

- **Space?**
  - $O(b^m)$ - keeps all nodes in memory

- **Optimal?**
  - No
A* Search

- Idea: Avoid expanding paths that are already expensive
- Evaluation function
  - \( f(n) = \text{estimated cost of cheapest path through } n \)
  - \( f(n) = g(n) + h(n) \)
- \( g(n) = \text{cost so far to reach } n \)
- \( h(n) = \text{estimated cost from } n \text{ to goal} \)
- \( f(n) = \text{estimated total cost of path through } n \text{ to goal} \)
Romania with Step Costs in km
A* Search Example
A* Search Example
A* Search Example
A* Search Example
A* Search Example
A* Search Example
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 state from \( n \).

• An admissible heuristic **never overestimates** the cost to reach the goal, i.e., it is **optimistic**
  – \( f(n) \) won’t overestimate then either

• Example: \( h_{SLD}(n) \)
  – Never overestimates the actual road distance

• **Theorem**: If \( h(n) \) is admissible, \( A^* \) using **TREE-SEARCH** is optimal
Proof: Optimality of A*

• Suppose some suboptimal goal $G_2$ has been generated and is in the fringe. Let $n$ be an unexpanded node in the fringe such that $n$ is on a shortest path to an optimal goal $G$.

- $g(G_2) > g(G)$ since $G_2$ is suboptimal
- $f(G_2) = g(G_2)$ since $h(G_2) = 0$
- $f(G) = g(G)$ since $h(G) = 0$
- $f(G_2) > f(G)$ from above
Proof: Optimality of $A^*$

• Suppose some suboptimal goal $G_2$ has been generated and is in the fringe. Let $n$ be an unexpanded node in the fringe such that $n$ is on a shortest path to an optimal goal $G$.

• $f(G_2) > f(G)$ from above
• $h(n) \leq h^*(n)$ since $h$ is admissible
• $g(n) + h(n) \leq g(n) + h^*(n)$
• $f(n) \leq f(G)$

So $f(G_2) > f(G) \geq f(n)$ and hence $f(G_2) > f(n)$

Therefore $A^*$ will never select $G_2$ for expansion.
Consistent Heuristics

- A heuristic is **consistent** if, for every node \( n \), every successor \( n' \) of \( n \) generated by any action \( a \):
  \[
  h(n) \leq c(n,a,n') + h(n')
  \]
- Consistency means \( f(n) \) should not decrease along path
**Theorem**: If $h(n)$ is consistent, A* using `GRAPH-SEARCH` is optimal

- Assume $h(n)$ is consistent
- $f(n)$ along path is non-decreasing
- Mathematically speaking:
  
  $f(n') = g(n') + h(n')$
  
  $= g(n) + c(n,a,n') + h(n')$
  
  $\geq g(n) + h(n)$
  
  $\geq f(n)$

- Whenever $n$ is expanded, we’ve found the best path to $n$
  - Otherwise we would’ve followed the better path first
- Thus all nodes expanded in non-decreasing order of $f(n)$
- $f(\text{goal}) = g(\text{goal})$
  - Because $f(\text{goal}) = g(\text{goal}) + h(\text{goal})$ and $h(\text{goal}) = 0$
- First goal node expanded must be therefore be least expensive goal
Admissibility and Consistency

• Consistency is stricter than admissibility
  – All consistent heuristics are admissible
  – Not all admissible heuristics are consistent
Optimality of A*  

- $A^*$ expands nodes in order of increasing $f$ value  
- Gradually adds "$f$-contours" of nodes  
- Contour $i$ has all nodes with $f=f_i$, where $f_i < f_{i+1}$
Properties of A*

• **Complete?**
  – Yes (unless there are infinitely many nodes with \( f(n) \leq \text{optimal solution cost } C^* \))

• **Time?**
  – Exponential
  – \( O(b^{(h^*-h)}) \)

• **Space?**
  – Keeps all nodes in memory

• **Optimal?**
  – Yes
  – Also optimally efficient
    • Expanding fewer nodes may miss optimal solution
Admissible Heuristics

E.g., for the 8-puzzle:

- $h_1(n) =$ number of misplaced tiles
- $h_2(n) =$ total Manhattan distance
  (i.e., no. of squares from desired location of each tile)

- $h_1(S) =$ ?
- $h_2(S) =$ ?
Admissible Heuristics

E.g., for the 8-puzzle:

- \( h_1(n) = \) number of misplaced tiles
- \( h_2(n) = \) total Manhattan distance
  (i.e., no. of squares from desired location of each tile)

- \( h_1(S) = 8 \)
- \( h_2(S) = ? \)
Admissible Heuristics

E.g., for the 8-puzzle:

• $h_1(n) = \text{number of misplaced tiles}$
• $h_2(n) = \text{total Manhattan distance}$
  (i.e., no. of squares from desired location of each tile)

\[
\begin{align*}
\text{Start State} & : 7 & 2 & 4 \\
& & 5 & 6 \\
& 8 & 3 & 1 \\
\text{Goal State} & : 1 & 2 \\
& 3 & 4 & 5 \\
& 6 & 7 & 8 \\
\end{align*}
\]

• $h_1(S) = 8$
• $h_2(S) = 3 + 1 + 2 + 2 + 2 + 3 + 3 + 2 = 18$
Dominance

- If $h_2(n) \geq h_1(n)$ for all $n$ (both admissible)
- then $h_2$ dominates $h_1$
- $h_2$ is better for search
- Typical search costs (average number of nodes expanded) over 100 8-puzzle instances:
- $d=12$
  - IDS = 364,404 nodes
  - $A^*(h_1) = 227$ nodes
  - $A^*(h_2) = 73$ nodes
- $d=24$
  - IDS = too many nodes
  - $A^*(h_1) = 39,135$ nodes
  - $A^*(h_2) = 1,641$ nodes
Relaxed Problems

- A problem with fewer restrictions on the actions is called a **relaxed problem**
- The cost of an optimal solution to a relaxed problem is an admissible heuristic for the original problem
- If the rules of the 8-puzzle are relaxed so that a tile can move anywhere, then $h_1(n)$ gives the shortest solution
- If the rules are relaxed so that a tile can move to any adjacent square, then $h_2(n)$ gives the shortest solution
Summary

• Heuristic functions estimate costs of shortest paths
• Good heuristics can dramatically reduce search cost
• Greedy best-first search expands lowest $h$
  – incomplete and not always optimal
• A* search expands lowest $g + h$
  – Complete
  – Optimal
  – Also optimally efficient (up to tie-breaks, for forward search)
    • Can’t explore fewer nodes due to risk of missing optimal solution
• Admissible heuristics can be derived from exact solution of relaxed problems