Artificial Intelligence

Informed search methods

In which we see how information about the state space can prevent algorithms from “blundering about in the dark”.
Uninformed vs. Informed Search

- **Uninformed** search strategies
  - Find solutions to problems by systematically generating new states and **testing for goal**
  - Most are incredibly inefficient

- **Informed** search strategies
  - Use problem-specific knowledge about **progress to the goal**
  - Find solutions more efficiently

Informed Search

- Informed search strategy
  - Has **guess** on how far away the goal sits for each state

- General approach is called “Best First” search
  - Node selected for expansion based on an **evaluation function**
    - Measuring a distance to goal
  - Node with **lowest** evaluation (cost) is selected
Techniques

• Best-first search: expand “minimum cost” nodes first
  – Greedy search
    • Minimize estimated cost to reach goal:
      \((\text{estimated cost from node } n \text{ to goal})\)
  – A* search
    • Estimated total costs through node \( n \) to goal:
      \((\text{actual cost to reach } n) + (\text{estimated cost from } n \text{ to goal})\)
• Iterative improvement algorithms
  – Continually moves in the direction of increasing value
  – Hill climbing searches
Best-First Search

- Try to expand node that is “closest” to goal
  - Use evaluation function to estimate how desirable
- Queue implementation
  - Insert expanded nodes in decreasing order of desirability (most desirable first)
- Two special cases in this lecture:
  - Greedy search: expand node closest to goal
  - A* search: expand node on least-cost solution path

Greedy Search

- Simplest best-first search strategy
- Minimize estimated cost to reach goal
- Evaluation function $h(n)$ (heuristic)
  - Estimate of cost from node $n$ to goal
    - Require $h(n) = 0$ if $n$ is goal
  - e.g., For our road map navigation example, we can use $h_{SLD}(n) = \text{straight-line distance}$ from city $n$ to Bucharest
- Greedy search expands the node that appears to be closest to the goal
  - Takes the “biggest bite” out of remaining cost to reach goal
    - Each step tries to get as close to goal as possible
  - Though, not considers if its action will be the best in the long run! (i.e., greedy)
Romania Step Costs in km

This is the “estimate to the goal” for each state!

<table>
<thead>
<tr>
<th>State</th>
<th>Distance to Bucharest</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>Dobrota</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>100</td>
</tr>
<tr>
<td>Rimnicu Vilcan</td>
<td>193</td>
</tr>
<tr>
<td>Sibiu</td>
<td>253</td>
</tr>
<tr>
<td>Timisoara</td>
<td>329</td>
</tr>
<tr>
<td>Uziceni</td>
<td>80</td>
</tr>
<tr>
<td>Vadui</td>
<td>199</td>
</tr>
<tr>
<td>Zerind</td>
<td>374</td>
</tr>
</tbody>
</table>
Greedy Search Example: Romania

\[ Q = [\text{Arad}] \]

**initial** \( Q \) is start state

\text{Arad} \leftarrow Q = [\cdot] \quad \text{“pop” } Q

\text{Is it the goal? NO}

---

Greedy Search Example: Romania

\[ Q = [\text{Sibu, Timisoara, Zerind}] \]

\text{add children of Arad IN ORDER OF COST to } Q

11

12
Greedy Search Example: Romania

"pop" Q
Sibu ← Q = [Timisoara, Zerind]
Is it the goal? NO
Q = [Fagaras, Rim.Vil., Tim., Zer., Ora.]

add children of Sibu
IN ORDER OF
COST to Q

Greedy Search Example: Romania

"pop" Q
Fagaras ← Q = [Rim.Vil., Tim., Zer., Ora.]
Is it the goal? NO
Q = [Buc., Rim.Vil., Tim., Zer., Ora.]

add children of Fagaras
IN ORDER OF
COST to Q
Greedy Search Example: Romania

“pop” Q
Buc. ← Q = [Rim.Vil., Tim., Zer., Ora.]

Is it the goal? YES!

SOLUTION
PATH:
Arad
Sibiu
Fagaras
Bucharest

Properties of Greedy Search

• Not complete (can be bad)
  – Can get stuck in loops
    • Go from Iasi to Fagaras:
      Iasi – Neamt – Iasi – Neamt (only 1 link)
    – Else, complete if check for repeated states
• Time (can be bad)
  – $O(b^m)$, $m =$ maximum depth (worst case, like DFS)
• Space (can be bad)
  – $O(b^m)$, keeps all nodes in memory (worst case)
• Not optimal (can be bad)
  – Heuristic is an estimate
  
  But a good heuristic can give dramatic improvement!
A* Search

- Minimizes total estimated path cost
- Avoids expanding paths already expensive
- Evaluation function $f(n) = g(n) + h(n)$
  - Actual cost to reach node $n$ so far $\rightarrow g(n)$
  - Estimated cost from $n$ to goal $\rightarrow h(n)$
  - Estimated total cost through $n$ to goal $\rightarrow f(n) = g(n) + h(n)$
  - A* search uses admissible heuristic
    - i.e., $h(n) \leq h^*(n)$ where $h^*(n)$ is true cost from $n$ to goal
    - e.g., $h_{SLD}(n)$ never overestimates actual distance

A* Search Example: Romania

![A* Search Example: Romania](image)

g(n): cost so far
\[ f(n) = g(n) + h(n) = \text{cost so far} + \text{estimated cost to goal} \]
A* Search Example: Romania

\[ g(n) = \text{cost so far} \]
\[ f(n) = g(n) + h(n) = \text{cost so far + estimated cost to goal} \]
**A* Search Example: Romania**

\[ g(n): \text{cost so far} \]

\[ f(n) = g(n) + h(n) = \text{cost so far} + \text{estimated cost to goal} \]
g(n): cost so far
f(n) = g(n) + h(n) = cost so far + estimated cost to goal
A* Search Example: Romania

Recall that Greedy solution path was \([\text{Arad}, \text{Sibiu}, \text{Fagaras}, \text{Bucharest}] = 450\)

\[
g(n): \text{cost so far} \\
+f(n) = g(n) + h(n) = \text{cost so far} + \text{estimated cost to goal}
\]

Admissible Heuristics
(Not over-estimate path cost)

- 8-puzzle
- \(h_1(n) = \text{number of misplaced tiles}\)
  - Must move each misplaced tile at least once
- \(h_2(n) = \text{total Manhattan distance (city block)}\)
  - Number of squares from desired tile location (Horiz+Vert)
Admissible Heuristics

\[ h_1(\text{Start}) = 7 \text{ (# tiles misplaced)} \]
\[ h_2(\text{Start}) = 2+3+3+2+4+2+0+2 = 18 \]  
(total Manhattan distance)

[Typical solution is about 20 steps, though varies with initial state]

Heuristic Dominance

- If \( h_2(n) \geq h_1(n) \) for all \( n \) (both admissible)
  - Then \( h_2 \) dominates \( h_1 \), and \( h_2 \) is better for search

- As the larger \( h_2 \) is closer to the optimal/true total cost \( h^* \), then A* using \( h_2 \) will expand fewer nodes (on average)
  - Recall that it cannot overestimate the true cost!
    (must be less than or equal)
Properties of A*

- Complete (good)
  - Unless infinitely many nodes
- Time and space (not good)
  - Still exponential (keeps all nodes in memory)
  - Can do “Iterative Deepening A*” to conserve memory
- Optimal (good)
  - Expands fewest nodes
- [Gaming: Sim City traffic article]
Iterative Improvement Methods

Local Search and Optimization

• So far, have seen systematic state search
  – Keeping one or more paths in memory
  – Returns solution path when goal is found
• In some problems, the path to goal is irrelevant
  – Just need the final “description” or “layout” (of goal)
    • Circuit design, factory-floor layout, …
• “Local search” starts in a state and moves only to neighboring states (path is not retained)
  – Uses little memory
  – Can find reasonable solutions in large spaces
• Uses an objective function to find best neighbor
  – Iterative improvement
Hill-Climbing Search

- Visualize nodes and successors being on a surface
  - Height is given by objective function
    - Assess the best successor to follow from the current state
  - Loop continually, moving in the direction of increasing value
  - Stop when reach a maximum node (peak)
- Also called “steepest ascent”

“Like climbing Everest in thick fog with amnesia!”

Hill Climbing
Hill Climbing

- Problems
  - Can be misled by local maxima
    - Get stuck at smaller peaks
  - Can get stuck on a flat plain (plateau)
    - Flat region in state-space function
      (search does random walk)
- Random-restart hill climbing
  - Random initial states
- But tends to explore fewer nodes in comparison with blind search methods
  - This is informed method, as it knows progress toward the goal

Local Beam Search

- Keeps track of $k$ states (not paths), rather than just one
  - Begins with $k$ randomly generated states
  - At each step, all successors of all $k$ states are generated
  - If any one is a goal, then finished
  - Else select $k$ “best” successors from complete list and repeat
- Useful information is passed among $k$ parallel search threads
  - Different than random-restart approach
Genetic Algorithms

• A stochastic hill-climbing search
• Large population of states is maintained
• Successor states are generated by combining two parent states (crossover) and changing (mutating)
• Best “offspring” make up new population
• [More coming in future lecture…]

Summary

• Informed search strategies
• Best-first search
  – Expand minimum cost nodes first
  – Greedy search
    • Minimize estimated cost to reach goal
  – A* search
    • Estimated total costs through $n$ to goal
    • $(\text{actual cost to reach } n) + (\text{estimated cost from } n \text{ to goal})$
    • Admissible heuristic never overestimates actual distance to goal
• Iterative improvement
  – Continually moves in the direction of increasing value
    • Hill climbing approaches