Informed Search

Day 2/3 of Search

Chap. 4, Russel & Norvig

Uninformed Search Complexity

- \( \text{N} \) = Total number of states
- \( \text{B} \) = Average number of successors (branching factor)
- \( \text{L} \) = Length for start to goal with smallest number of steps
- \( \text{Q} \) = Average size of the priority queue
- \( L_{\text{max}} \) = Length of longest path from \textit{START} to any state

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Complete</th>
<th>Optimal</th>
<th>Time</th>
<th>Space</th>
</tr>
</thead>
<tbody>
<tr>
<td>BFS</td>
<td>Y</td>
<td>Y</td>
<td>( \text{O}(\text{Min}(\text{N}, \text{B}^\text{L})) )</td>
<td>( \text{O}(\text{Min}(\text{N}, \text{B}^\text{L})) )</td>
</tr>
<tr>
<td>BBFS</td>
<td>Y</td>
<td>Y</td>
<td>( \text{O}(\text{Min}(\text{N}, \text{2B}^\text{L})) )</td>
<td>( \text{O}(\text{Min}(\text{N}, \text{2B}^\text{L})) )</td>
</tr>
<tr>
<td>UCS</td>
<td>Y, if cost &gt; 0</td>
<td>Y, if cost &gt; 0</td>
<td>( \text{O}(\text{log}(\text{B}^\text{L})) )</td>
<td>( \text{O}(\text{Min}(\text{N}, \text{B}^\text{L})) )</td>
</tr>
<tr>
<td>PCDFS</td>
<td>Y</td>
<td>N</td>
<td>( \text{O}(\text{B}^\text{Lmax}) )</td>
<td>( \text{O}(\text{BL}_{\text{max}}) )</td>
</tr>
<tr>
<td>MEMDFS</td>
<td>Y</td>
<td>N</td>
<td>( \text{O}(\text{Min}(\text{N}, \text{B}^\text{Lmax})) )</td>
<td>( \text{O}(\text{Min}(\text{N}, \text{B}^\text{Lmax})) )</td>
</tr>
<tr>
<td>IDS</td>
<td>Y</td>
<td>Y, if all trans. have same cost</td>
<td>( \text{O}(\text{F}) )</td>
<td>( \text{O}(\text{BL}) )</td>
</tr>
</tbody>
</table>

Search Revisited

1. Store a value \( f(s) \) at each state \( s \)
2. Choose the state with lowest \( f \) to expand next
3. Insert its successors

If \( f(\cdot) \) is chosen carefully, we will eventually find the lowest-cost sequence

Example:

- UCS (Uniform Cost Search): \( f(A) = g(A) \) = total cost of current shortest path from \textit{START} to \( A \)
- Store states awaiting expansion in a priority queue for efficient retrieval of minimum \( f \)
- Optimal \( \rightarrow \) Guaranteed to find lowest cost sequence, \textit{but}......
• Problem: No guidance as to how “far” any given state is from the goal
• Solution: Design a function $h(.)$ that gives us an estimate of the distance between a state and the goal

Our best guess is that $A$ is closer to GOAL than $B$ so maybe it is a more promising state to expand

Heuristic Functions

• $h(.)$ is a heuristic function for the search problem
• $h(s) =$ estimate of the cost of the shortest path from $s$ to GOAL
• $h(.)$ cannot be computed solely from the states and transitions in the current problem → If we could, we would already know the optimal path!
• $h(.)$ is based on external knowledge about the problem → informed search
• Questions:
  1. Typical examples of $h$?
  2. How to use $h$?
  3. What are desirable/necessary properties of $h$?

Heuristic Functions Example

• $h(s) =$ Linear-geometric distance to GOAL

The straight-line distance is lower from $s$ than from $s'$ so maybe $s$ has a better chance to be on the best path

Heuristic Functions Example

• How could we define $h(s)$?
First Attempt: Greedy Best First Search

- Simplest use of heuristic function: Always select the node with smallest $h(.)$ for expansion (i.e., $f(s) = h(s)$)

Initialize $PQ$
Insert $START$ with value $h(START)$ in $PQ$
While ($PQ$ not empty and no goal state is in $PQ$)
    Pop the state $s$ with the minimum value of $h$ from $PQ$
    For all $s'$ in $\text{succs}(s)$
        If $s'$ is not already in $PQ$ and has not already been visited
            Insert $s'$ in $PQ$ with value $h(s')$

Problem
- What solution do we find in this case?
- Greedy search clearly not optimal, even though the heuristic function is non-stupid

Trying to Fix the Problem
- $g(s)$ is the cost from $START$ to $s$ only
- $h(s)$ estimates the cost from $s$ to $GOAL$
- Key insight: $g(s) + h(s)$ estimates the total cost of the cheapest path from $START$ to $GOAL$ going through $s$
- $\rightarrow A^*$ algorithm

Can $A^*$ Fix the Problem?
- $f(A) = h(A) + g(A) = 3 + g(START) + \text{cost}(START, A) = 3 + 0 + 2)$
- $f(B) = h(B) + g(B) + h(B) = 11$
- $f(C) = h(C) + g(C) = 1 + g(A) + \text{cost}(A, C) = 1 + 2 + 4)$
- $f(GOAL) = h(GOAL) = 0$
Can A* Fix the Problem?

\[ f(A) = h(A) + g(A) = g(\text{START}) + \text{cost}(\text{START}, A) = 3 + 0 + 2 \]

\[ f(C) = h(C) + g(C) = 1 + g(A) + \text{cost}(A, C) = 1 + 2 + 4 \]

\[ f(C) = h(C) + g(C) = 1 + g(B) + \text{cost}(B, C) = 1 + 3 + 1 \]

\[ (\text{GOAL}, 6) \]

A state that was already in the queue is re-visited.
How is its priority updated?

A* Termination Condition

- Stop when GOAL is popped from the queue!
Revisiting States

A state that had been already expanded is re-visited.
(Careful: This is a different example.)

A* Algorithm (inside loop)

- Pop state $s$ with lowest $f(s)$ in queue
- If $s = \text{GOAL}$
  - return $\text{SUCCESS}$
- Else expand $s'$:
  - For all $s'$ in $\text{succs}(s)$:
    - $f' = g(s') + h(s') = g(s) + \text{cost}(s,s') + h(s')$
    - If ($s'$ not seen before OR $s'$ previously expanded with $f(s') > f'$ OR $s'$ in $\text{PQ}$ with $f(s') > f'$)
      - Promote/Insert $s'$ with new value $f'$ in $\text{PQ}$
      - $\text{previous}(s') \leftarrow s$
    - Else
      - Ignore $s'$ (because it has been visited and its current path cost $f(s')$ is still the lowest path cost from $\text{START}$ to $s'$)

Under what Conditions is $A^*$ Optimal?

- Problem: $h(.)$ is a poor estimate of path cost to the goal state

Admissible Heuristics

- Define $h^*(s) = \text{the true minimal cost to the goal from } s$
- $h$ is admissible if
  - $h(s) \leq h^*(s)$ for all states $s$
- In words: An admissible heuristic never overestimates the cost to the goal. “Optimistic” estimate of cost to goal.

$A^*$ is guaranteed to find the optimal path if $h$ is admissible
Consistent (Monotonic) Heuristics

\[ h(s) \leq h(s') + \text{cost}(s, s') \]

Sort of triangular inequality implies that path cost always increases + need to expand node only once

\[ h(s) \leq h(s') + \text{cost}(s, s') \]

Pop state \( s \) with lowest \( f(s) \) in queue
If \( s = \text{GOAL} \)
return SUCCESS
Else expand \( s \):
For all \( s' \) in \( \text{succs} \) (s):
\[ f' = g(s') + h(s') = g(s) + \text{cost}(s, s') + h(s) \]
If \( s' \) not seen before OR
\( s' \) previously expanded with \( f(s') > f' \) OR
\( s' \) in \( \text{PQ} \) with with \( f(s') > f' \)
Promote/Insert \( s' \) with new value \( f' \) in \( \text{PQ} \)
previous(\( s' \)) \( \leftarrow \) s
Else
Ignore \( s' \) (because it has been visited and its current path cost \( f(s') \) is still the lowest path cost from \( \text{START} \) to \( s' \))

Examples

For the navigation problem:
The length of the shortest path is at least the distance between \( s \) and \( \text{GOAL} \)
Euclidean distance is an admissible heuristic

What about the puzzle?
Comparing Heuristics

$\text{h}_1(s) = 7$

$\text{h}_2(s) = 2 + 3 + 3 + 2 + 4 + 2 + 0 + 2 = 18$

- $\text{h}_2$ dominates $\text{h}_1$ if $\text{h}_2(s) \geq \text{h}_1(s)$ for all $s$

Intuition: since $h \leq h^*$, a larger $h$ is a better approximation of the true path cost.

Limitations

- Computation: In the worst case, we may have to explore all the states $\Rightarrow O(N)$

- The good news: $A^*$ is optimally efficient $\Rightarrow$ For a given $h(.)$, no other optimal algorithm will expand fewer nodes

- The bad news: Storage is also potentially exponential $\Rightarrow O(N)$
**IDA* (Iterative Deepening A*)**

- Same idea as Iterative Deepening DFS except use \( f(s) \) to control depth of search instead of the number of transitions.
- Example, assuming integer costs:
  1. Run DFS, stopping at states \( s \) such that \( f(s) > 0 \)
      Stop if goal reached
  2. Run DFS, stopping at states \( s \) such that \( f(s) > 1 \)
      Stop if goal reached
  3. Run DFS, stopping at states \( s \) such that \( f(s) > 2 \)
      Stop if goal reached
      .......Keep going by increasing the limit on \( f \) by 1 every time

- Complete (assuming we use loop-avoiding DFS)
- Optimal
- More expensive in computation cost than A*
- Memory order \( L \) as in DFS

**Summary**

- Informed search and heuristics
- First attempt: Best-First Greedy search
- A* algorithm
  - Optimality
  - Condition on heuristic functions
  - Completeness
  - Limitations, space complexity issues
  - Extensions

Chapters 3&4 Russel & Norvig