Informed Search

Chap. 4

Uninformed Search Complexity

- \( N \) = Total number of states
- \( B \) = Average number of successors (branching factor)
- \( L \) = Length for start to goal with smallest number of steps
- \( Q \) = Average size of the priority queue
- \( L_{\text{max}} \) = Length of longest path from 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, If all trans. have same cost</td>
<td>( O(\text{Min}(N,B^L)) )</td>
<td>( O(\text{Min}(N,B^L)) )</td>
</tr>
<tr>
<td>BiBFS</td>
<td>Y</td>
<td>Y, If all trans. have same cost</td>
<td>( O(\text{Min}(N,2B^{L-2})) )</td>
<td>( O(\text{Min}(N,2B^{L-2})) )</td>
</tr>
<tr>
<td>UCS</td>
<td>Y, If cost &gt; 0 I</td>
<td>Y, If cost &gt; 0 I</td>
<td>( O(\log(Q)*B^{C/\varepsilon}) )</td>
<td>( O(\text{Min}(N,B^{C/\varepsilon})) )</td>
</tr>
<tr>
<td>PCDFS</td>
<td>Y</td>
<td>N</td>
<td>( O(B^{L_{\text{max}}}) )</td>
<td>( O(B_{L_{\text{max}}}) )</td>
</tr>
<tr>
<td>MEMDFS</td>
<td>Y</td>
<td>N</td>
<td>( O(\text{Min}(N,B^{L_{\text{max}}})) )</td>
<td>( O(\text{Min}(N,B^{L_{\text{max}}})) )</td>
</tr>
<tr>
<td>IDS</td>
<td>Y</td>
<td>Y, If all trans. have same cost</td>
<td>( O(B^L) )</td>
<td>( O(BL) )</td>
</tr>
</tbody>
</table>

Material in part from http://www.cs.cmu.edu/~swm/tutorials
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 $f(.)$ 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 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, 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

![](image.png)

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 \(\Rightarrow\) If we could, we would already know the optimal path!
- $h(.)$ is based on external knowledge about the problem \(\Rightarrow\) *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) = $ Euclidean 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 $h(s)$?

• $h(s) = $ Euclidean distance to GOAL
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?

\[
\begin{align*}
(f(A) &= h(A) + g(A) = 3 + g(START) + \text{cost}(START, A) = 3 + 0 + 2) \\
&\{(A,5)\} \\
(f(C) &= h(C) + g(C) = 1 + g(A) + \text{cost}(A, C) = 1 + 2 + 4) \\
&\{(C,5)\} \\
(f(C) &= h(C) + g(C) = 1 + g(B) + \text{cost}(B, C) = 1 + 3 + 1) \\
&\{(GOAL,6)\}
\end{align*}
\]
Can A* Fix the Problem?

- Termination condition
- Revisiting states
- Algorithm
- Optimality
- Avoiding revisiting states
- Choosing good heuristics
- Reducing memory usage
A* Termination Condition

- Stop when GOAL is popped from the queue!

Queue:

{(B, 4) (A, 8)}
{(C, 4) (A, 8)}
{(D, 4) (A, 8)}
{(A, 8) (G, 10)}
A state that was already in the queue is re-visited.
How is its priority updated?

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 $s = GOAL$
    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 PQ with with $f(s') > f'$)
            Promote/Insert $s'$ with new value $f'$ in 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 $START$ to $s'$)

Under what Conditions is A* Optimal?

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

Final path: 
\{(START,6)\}
\{(GOAL,3) (A,8)\}

with cost = 3
Admissible Heuristics

- Define $h^*(s) = \text{the true minimal cost to the goal from } s$
- $h$ is admissible if $h(s) \leq h^*(s) \text{ for all states } s$
- 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

$\text{Cost}(s, s')$

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

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 \( PQ \) with with \( f(s') > f' \))
    Promote/Insert \( s' \) with new value \( f' \) in \( PQ \)
    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 \texttt{GOAL} →
Euclidean distance is an admissible heuristic

What about the puzzle?

Comparing Heuristics

<table>
<thead>
<tr>
<th>Heuristic</th>
<th>$L = 4$ steps</th>
<th>$L = 8$ steps</th>
<th>$L = 12$ steps</th>
</tr>
</thead>
<tbody>
<tr>
<td>$h_1$ = misplaced tiles</td>
<td>112</td>
<td>6,300</td>
<td>$3.6 \times 10^6$</td>
</tr>
<tr>
<td>$h_2$ = Manhattan distance</td>
<td>13</td>
<td>39</td>
<td>227</td>
</tr>
<tr>
<td>$A^*$ with heuristic $h_1$</td>
<td>12</td>
<td>25</td>
<td>73</td>
</tr>
<tr>
<td>$A^*$ with heuristic $h_2$</td>
<td>13</td>
<td>39</td>
<td>227</td>
</tr>
</tbody>
</table>

- Overestimates $A^*$ performance because of the tendency of IDS to expand states repeatedly
- Number of states expanded does not include \log() time access to queue

Example from Russell&Norvig
Comparing Heuristics

\( h_1(s) = 7 \)
\( h_2(s) = 2 + 3 + 3 + 2 + 4 + 2 + 0 + 2 = 18 \)

\( h_2 \) is larger than \( h_1 \) and, at same time, A* seems to be more efficient with \( h_2 \).

Is there a connection between these two observations?

**\( h_2 \) dominates \( h_1 \) if \( h_2(s) \geq h_1(s) \) for all \( s \)**

For any two heuristics \( h_2 \) and \( h_1 \):
- If \( h_2 \) dominates \( h_1 \), then A* is more efficient (expands fewer states) with \( h_2 \)

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 large $\rightarrow O(N)$

IDS (Iterative Deepening Search)

• Need to make DFS optimal

• IDS (Iterative Deepening Search):
  – Run DFS by searching only path of length 1 (DFS stops if length of path is greater than 1)
  – If that doesn’t find a solution, try again by running DFS on paths of length 2 or less
  – If that doesn’t find a solution, try again by running DFS on paths of length 3 or less
    – ………..
    – Continue until a solution is found
Example: 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