

Summary

Discussion

- Search:
 - ① Uninformed: Breadth first search → Add states at the end → Remove states from the front → Complete + Optimal.
 - ② Uninformed: Depth first search → Add states to the front → Remove states to the front → Incomplete + Not optimal.
 - ③ Uninformed: Iterative deepening search → DFS with depth limits 1, 2, ... → Complete + Optimal.
 - ④ Informed: Uniform cost search.
 - ⑤ Informed: Best first greedy search.
 - ⑥ Informed: A search.
 - ⑦ Informed: A star search.

Uniformed vs. Informed Search

Motivation

- Uninformed search means only the goal G and the successor functions s' are given.
- Informed search means which non-goal states are better is also known.

Heuristic

Motivation

- The additional information is usually given as a heuristic cost from a state s to the goal.
- The cost of the path from the start to a vertex s in the frontier is $g(s)$.
- The cost from s to the goal, $h^*(s)$, is estimated by $h(s)$. This estimate may not be accurate.

$$h(s) \approx h^*(s)$$

Uniform Cost Search

Description

- Expand the vertices with the lowest current path cost $g(s)$ first.
- It is BFS with a priority queue based on $g(s)$.
- It is equivalent to BFS if $c = 1$ is constant on all edges.
- It is also called Dijkstra's Algorithm.

Uniform Cost Search

Algorithm

- Input: a weighted digraph (V, E, c) , initial states I and goal states G .
- Output: a path from I to G .
- EnQueue initial states into a priority queue Q . Here, Q is ordered by $g(s)$ for $s \in Q$.

$$Q = I$$

- While Q is not empty and goal is not deQueued, deQueue Q and enQueue its successors.

$$s = Q_{(0)} = \operatorname{argmin}_{s \in Q} g(s)$$

$$Q = Q + s'(s)$$

Best First Greedy Search

Description

- Expand the vertices with the lowest heuristic cost $h(s)$ first.
- Use a priority queue based on $h(s)$.
- BFGS is not an abbreviation of Best First Greedy Search: BFGS is the Broyden Fletcher Goldfarb Shanno algorithm (a version of gradient descent).

Best First Greedy Search

Algorithm

- Input: a weighted digraph (V, E, c) , initial states I and goal states G , and the heuristic function $h(s), s \in V$.
- Output: a path from I to G .
- EnQueue initial states into a priority queue Q . Here, Q is ordered by $h(s)$ for $s \in Q$.

$$Q = I$$

- While Q is not empty and goal is not deQueued, deQueue Q and enQueue its successors.

$$s = Q_{(0)} = \operatorname{argmin}_{s \in Q} h(s)$$

$$Q = Q + s'(s)$$

A Search

Description

- Expand the vertices with the lowest total cost $g(s) + h(s)$ first.
- Use a priority queue based on $g(s) + h(s)$.
- A stands for Always be optimistic?

A Search

Algorithm

- Input: a weighted digraph (V, E, c) , initial states I and goal states G , and the heuristic function $h(s), s \in V$.
- Output: a path from I to G .
- EnQueue initial states into a priority queue Q . Here, Q is ordered by $g(s) + h(s)$ for $s \in Q$.

$$Q = I$$

- While Q is not empty and goal is not deQueued, deQueue Q and enQueue its successors.

$$s = Q_{(0)} = \operatorname{argmin}_{s \in Q} g(s) + h(s)$$

$$Q = Q + s'(s)$$

Admissible Heuristic

Definition

- A heuristic is admissible if it never over estimates the true cost.

$$0 \leq h(s) \leq h^*(s)$$

Dominated Heuristic

Definition

- One heuristic, h_1 , is dominated by another, h_2 , if:

$$h_1(s) \leq h_2(s) \leq h^*(s), \forall s \in S$$

- If h_2 dominates h_1 , then h_2 is better than h_1 since A^* using h_1 expands at least as many states (or more) than A^* using h_2 .
- If h_2 dominated h_1 , A^* with h_2 is better informed than A^* with h_1 .

Non-Optimal Heuristic

Definition

- If optimality is not required and a satisfying solution is acceptable, then the heuristic should be as close as possible, either under or over, to the actual cost.
- This results in fewer states being expanded compared to using poor but admissible heuristics.

A Star Search with Revisit, Part I

Algorithm

- Input: a weighted digraph (V, E, c) , initial states I and goal states G , and the heuristic function $h(s)$, $s \in V$.
- Output: a path with minimum cost from I to G .
- EnQueue initial states into a priority queue Q . Here, Q is ordered by $g(s) + h(s)$ for $s \in Q$.

$$Q = I$$

$$g(I) = 0$$

$$g(s) = \infty, \text{ for } s \notin I$$

- Initialize the list of visited vertices, P .

$$P = \emptyset$$

A Star Search with Revisit, Part II

Algorithm

- While Q is not empty and goal is not deQueued, deQueue Q , put it on P and enQueue its successors to Q , and update the cost functions.

$$s = Q_{(0)} = \operatorname{argmin}_{s \in Q} g(s) + h(s)$$

$$P = P + s$$

$$Q = Q + s'(s), \text{ update } g(s') = \min \{g(s'), g(s) + c(s, s')\}$$

Iterative Deepening A Star Search

Discussion

- A^* can use a lot of memory.
- Do path checking without expanding any vertex with $g(s) + h(s) > 1$.
- Do path checking without expanding any vertex with $g(s) + h(s) > 2$.
- ...
- Do path checking without expanding any vertex with $g(s) + h(s) > d$.

Iterative Deepening A Star Search Performance

Discussion

- IDA* is complete.
- IDA* is optimal.
- IDA* is more costly than A^* .

Beam Search

Discussion

- Version 1: Keep a priority queue with fixed size k . Only keep the top k vertices and discard the rest.
- Version 2: Only keep the vertices that are at most ϵ worse than the best vertex in the queue. ϵ is called the beam width.

Summary

Discussion

- Search:
 - 1 Uninformed: Breadth first search.
 - 2 Uninformed: Depth first search.
 - 3 Uninformed: Iterative deepening search.
 - 4 Informed: Uniform cost search → Remove states with lowest current cost (g) → Complete, Optimal.
 - 5 Informed: Best first greedy search → Remove states with the lowest heuristic cost (h) → Incomplete, Not optimal.
 - 6 Informed: A search → Remove states with the lowest current plus heuristic cost ($g + h$) → Complete, Not Optimal.
 - 7 Informed: A star search → admissible heuristic (under-estimates true cost) → Complete, Optimal.