UCS and Greedy

▲□▶ ▲□▶ ▲ 三▶ ▲ 三▶ 三 のへぐ

CS540 Introduction to Artificial Intelligence Lecture 18

Young Wu Based on lecture slides by Jerry Zhu, Yingyu Liang, and Charles Dyer

July 26, 2021

UCS and Greedy

▲ロ ▶ ▲周 ▶ ▲ 国 ▶ ▲ 国 ▶ ● の Q @

Uniformed vs. Informed Search

- 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.

UCS and Greedy

▲□▶ ▲□▶ ▲□▶ ▲□▶ ■ ●の00

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, $h^{\star}(s)$, is estimated by h(s). This estimate may not be accurate.

$$h(s) \approx h^{\star}(s)$$

UCS and Greedy

Heuristic Diagram

▲□▶ ▲□▶ ▲ 三▶ ▲ 三▶ 三三 - のへぐ

UCS and Greedy

▲ロ ▶ ▲周 ▶ ▲ 国 ▶ ▲ 国 ▶ ● の Q @

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.

UCS and Greedy

◆□▶ ◆□▶ ◆三▶ ◆三▶ 三三 - のへで

Uniform Cost Search Maze Example

UCS and Greedy

Uniform Cost Search Simple Example

▲□▶ ▲□▶ ▲□▶ ▲□▶ □ のQで

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 ∈ Q.

$$Q = I$$

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

$$\begin{split} s &= Q_{(0)} = \arg\min_{s \in Q} g\left(s\right) \\ Q &= Q + s'\left(s\right) \end{split}$$

UCS and Greedy

▲□▶ ▲□▶ ▲三▶ ▲三▶ 三三 のへで

Uniform Cost Search Performance

- UCS is complete.
- UCS is optimal with any c.

UCS and Greedy

▲ロ ▶ ▲周 ▶ ▲ 国 ▶ ▲ 国 ▶ ● の Q @

Best First Greedy Search

- Expand the vertices with the lowest heuristic cost h(s) first.
- Use a priority queue based on h(s).

UCS and Greedy

▲□▶ ▲□▶ ▲ 三▶ ▲ 三▶ 三三 - のへぐ

Greedy Search Maze Example

UCS and Greedy

Greedy Search Simple Example

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 ∈ 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 ∈ Q.

$$Q = I$$

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

$$s = Q_{(0)} = \arg \min_{s \in Q} h(s)$$
$$Q = Q + s'(s)$$

UCS and Greedy

▲□▶ ▲□▶ ▲ 三▶ ▲ 三▶ 三 のへぐ

Best First Greedy Search Performance

- Greedy is incomplete.
- Greedy is not optimal.

UCS and Greedy

▲ロ ▶ ▲周 ▶ ▲ 国 ▶ ▲ 国 ▶ ● の Q @

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?

UCS and Greedy

A Search Maze Example

Definition



UCS and Greedy

A Search Simple Example 1

UCS and Greedy

A Search Simple Example 2

▲□▶ ▲□▶ ▲ 三▶ ▲ 三▶ 三三 - のへぐ

UCS and Greedy

A Search Simple Example 3

▲□▶ ▲□▶ ▲ 三▶ ▲ 三▶ 三三 - のへぐ

▲□▶ ▲□▶ ▲□▶ ▲□▶ □ のQで

A Search

Algorithm

- Input: a weighted digraph (V, E, c), initial states I and goal states G, and the heuristic function h (s), s ∈ 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 ∈ Q.

$$Q = I$$

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

$$\begin{split} s &= Q_{(0)} = \arg\min_{s \in Q} g\left(s\right) + h\left(s\right) \\ Q &= Q + s'\left(s\right) \end{split}$$

UCS and Greedy

◆□▶ ◆□▶ ◆三▶ ◆三▶ 三三 - のへで

A Search Performance

Discussion

- A is complete.
- A is not optimal.

UCS and Greedy

◆□▶ ◆□▶ ◆三▶ ◆三▶ 三三 - のへで

A Star Search Description

• A^* search is A search with an admissible heuristic.

UCS and Greedy

◆□▶ ◆□▶ ◆三▶ ◆三▶ 三三 - のへで

Admissible Heuristic

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

$$0\leqslant h\left(s
ight)\leqslant h^{\star}\left(s
ight)$$

UCS and Greedy

Admissible Heuristic 8 Puzzle Example

Definition



UCS and Greedy

Dominated Heuristic

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

 $h_{1}\left(s
ight)\leqslant h_{2}\left(s
ight)\leqslant h^{\star}\left(s
ight),\,\,\forall\,s\in S$

- If h₂ dominates h₁, then h₂ is better than h₁ since A^{*} using h₁ expands at least as many states (or more) than A^{*} using h₂.
- If h₂ dominated h₁, A^{*} with h₂ is better informed than A^{*} with h₁.

UCS and Greedy

▲ロ ▶ ▲周 ▶ ▲ 国 ▶ ▲ 国 ▶ ● の Q @

Non-Optimal Heuristic

- 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.

Admissible Heuristic 8 Puzzle Example

- Which ones (select multiple) of the following are admissible heuristic function for the 8 Puzzle?
- A: h(s) = number of tiles in the wrong position.
- B: h(s) = 0.
- C: h(s) = 1.
- D: h(s) = sum of Manhattan distance between each tile and its goal location.
- E: h(s) = sum of Euclidean distance between each tile and its goal location.

▲□▶ ▲□▶ ▲□▶ ▲□▶ □ のQで

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 ∈ 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 ∈ Q.

$$Q = I$$

$$g(I) = 0$$

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

• Initialize the list of visited vertices, P.

$$P= \emptyset$$

▲ロ ▶ ▲周 ▶ ▲ 国 ▶ ▲ 国 ▶ ● の Q @

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.

$$\begin{split} s &= Q_{(0)} = \arg\min_{s \in Q} g\left(s\right) + h\left(s\right) \\ P &= P + s \\ Q &= Q + s'\left(s\right), \text{ update } g\left(s'\right) = \min\left\{g\left(s'\right), g\left(s\right) + c\left(s,s'\right)\right\} \end{split}$$

UCS and Greedy

◆□▶ ◆□▶ ◆三▶ ◆三▶ 三三 - のへで

A Search Performance

Discussion

- A^{*} is complete.
- A^{*} is optimal.

▲□▶ ▲□▶ ▲□▶ ▲□▶ □ のQで

Iterative Deepening A Star Search

- 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.

▲ロ ▶ ▲周 ▶ ▲ 国 ▶ ▲ 国 ▶ ● の Q @

Iterative Deepening A Star Search Performance

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

UCS and Greedy

▲ロ ▶ ▲周 ▶ ▲ 国 ▶ ▲ 国 ▶ ● の Q @

Beam Search

- 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.

UCS and Greedy

A 00000000000000000000000000

◆□▶ ◆□▶ ◆三▶ ◆三▶ 三三 - のへで

Beam Search Performance

Discussion

- Beam is incomplete.
- Beam is not optimal.