Search

Tristan Slater

October 6, 2022

Contents

1 Defining Search Problems

1. Initial state
2. Successor
3. Goal test
4. Step cost

1.1 Successor Function

Tells us where we can go from a state:

$$S(s_0) = \{(a_1, s_1), ..., (a_n, s_n)\} \tag{1}$$

When actions are inferrable from states:

$$S(s_0) = \{s_1, ..., s_n\} \tag{2}$$

1.2 Goal Test Function

Tells when solution has been reached. **Explicit** defines which state is a goal. **Implicit** is some set of conditions that could happen at various states under various conditions.

1.3 Step Cost Function

Tells how much a single step costs:

$$c(s_1, a, s_2) \geq 0 \tag{3}$$

1.4 Solution

Any path from initial state to goal state:

$$\{a_1, ..., a_n\}$$

When actions are inferrable from states:

$$\{s_1, ..., s_n\}$$

2 Basic Search

A **node** is an abstract data structure that contains information about a state, parent node, action, path cost, and depth (optional).

The **frontier** is the collection of nodes that have been *generated*, but *not expanded*.

Table 1: Comparison of Basic Search Strategies

	BFS	UCS	DFS
Node ADS	Queue	Min queue[3]	Stack
Nodes to Expand First	Shallowest	Cheapest	Deepest
Complete?	Yes[1]	Yes[4]	No
Optimal?	Yes[2]	Yes	No
Time Complexity	$O(b^{d+1})$	$O(b^{\lceil C^*/\varepsilon \rceil})$[5]	$O(b^m)$
Space Complexity	$O(b^{d+1})$	$O(b^{\lceil C^*/\varepsilon \rceil})$	$O(bm)$

[1] If b finite
[2] If step costs equal
[3] Path cost
[4] If step cost > 0
[5] C^* cost of solution

DLS is exactly like DFS except with the depth limited.
IDS is exactly like DLS except it tries all depths starting at 0.
IDS is *complete*.
IDS is *optimal* if the step cost is 1.

3 Best-First Search

Like UCS, but uses a **evaluation function** $f(n)$ to pick the *best* node to expand first.

Table 2: Comparison of Best-First Search Strategies

	Greedy	A*
Evaluation Function	$h(n)$	$g(n) + h(n)$
Complete?	No	Yes
Optimal?	No	Yes[7]
Time Complexity	$O(b^m)$	-
Space Complexity	$O(b^m)$	-

[7] For tree search, if $h(n)$ is admissible

3.1 A*

$g(n)$: Cost so far to reach n.

3.2 Heuristics

3.2.1 Admissibility

If heuristic is at most the true cost for every node:

$$h(n) \leq c(n) \text{ for all } n$$

Time/space complexity depends on heuristics.

3.2.2 Dominance

$h_2(n)$ *dominates* $h_1(n)$ if $h_2(n) \geq h_1(n)$ for all n.

If $h_2(n)$ is admissible, then $h_1(n)$ is admissible.