INFORMED SEARCH

Lecture 3.1
CSE368: Artificial Intelligence
June 5, 2019

*Slides are adopted from:
- Prof. Nils Napp’s CSE368 course, Fall 2018
- UC Berkeley course CS188 Introduction to AI, Spring 2019
An agent *perceives* its environment through *sensors* and *acts* upon it through *actuators*.
Recap: Intelligent Agents

What is the goal of the intelligent agent?
Recap: Intelligent Agents

Intelligent agents are supposed to maximize their performance measure.
Quiz: State Space Graphs vs. Search Trees

Consider this 4-state graph: How big is its search tree (from S)?
Quiz: State Space Graphs vs. Search Trees

Consider this 4-state graph:

How big is its search tree (from S)?
State Space Graphs vs. Search Trees

State Space Graph

Search Tree

Each NODE in the search tree is an entire PATH in the state space graph.

We construct both on demand – and we construct as little as possible.
Recap: Search Algorithms

SEARCH ALGORITHMS

UNINFORMED SEARCH
- Depth First Search
- Breadth First Search
- Uniform Cost Search

INFORMED SEARCH
- Greedy Search
- A* Search
- Graph Search
Recap: DFC vs BFS

DFS?
Recap: DFC vs BFS

Recap: DFC vs BFS

DFS: 1 → 2 → 5 → 9 → 10 → 6 → 3 → 4 → 7 → 11 → 12 → 8

BFS?
Recap: DFC vs BFS

DFS: 1 → 2 → 5 → 9 → 10 → 6 → 3 → 4 → 7 → 11 → 12 → 8

BFS: 1 → 2 → 3 → 4 → 5 → 6 → 7 → 8 → 9 → 10 → 11 → 12
Uniform Cost Search

\[
Q \leftarrow \langle \text{start} \rangle; \quad \text{// Initialize the queue with the starting node}
\]
\[
\text{while } Q \text{ is not empty do}
\]
\[
\quad \text{Pick (and remove) the path } P \text{ with lowest cost } g = w(P) \text{ from the queue } Q; \quad \text{// Reached the goal}
\]
\[
\quad \text{if } \text{head}(P) = \text{goal} \text{ then return } P; \quad \text{// for all neighbors}
\]
\[
\quad \text{foreach vertex } v \text{ such that } (\text{head}(P), v) \in E, \text{ do}
\]
\[
\quad \quad \text{add } \langle v, P \rangle \text{ to the queue } Q; \quad \text{// Add expanded paths}
\]
\[
\text{return FAILURE; \quad \text{// Nothing left to consider.}
\]
Uniform Cost Search

<table>
<thead>
<tr>
<th>path</th>
<th>cost</th>
</tr>
</thead>
<tbody>
<tr>
<td>\langle s \rangle</td>
<td>0</td>
</tr>
</tbody>
</table>

Diagram:

- Start at node \( s \)
- Path to node \( g \) with cost: \( 2 \rightarrow 2 \rightarrow 2 \rightarrow 3 \rightarrow 5 \)
Uniform Cost Search

<table>
<thead>
<tr>
<th>path</th>
<th>cost</th>
</tr>
</thead>
<tbody>
<tr>
<td>(a, s)</td>
<td>2</td>
</tr>
<tr>
<td>(b, s)</td>
<td>5</td>
</tr>
</tbody>
</table>

Diagram showing a graph with nodes labeled \(s, a, b, d, c, g\) and edges with weights.
Uniform Cost Search

Q:

<table>
<thead>
<tr>
<th>state</th>
<th>cost</th>
</tr>
</thead>
<tbody>
<tr>
<td>\langle c, a, s \rangle</td>
<td>4</td>
</tr>
<tr>
<td>\langle b, s \rangle</td>
<td>5</td>
</tr>
<tr>
<td>\langle d, a, s \rangle</td>
<td>6</td>
</tr>
</tbody>
</table>

Diagram:

- Node labels: c, a, s, d, b, g
- Edges with weights:
  - s -> a: 2
  - a -> c: 4
  - s -> d: 2
  - d -> g: 5
  - a -> b: 5
  - s -> b: 2
Uniform Cost Search

Q:

<table>
<thead>
<tr>
<th>state</th>
<th>cost</th>
</tr>
</thead>
<tbody>
<tr>
<td>(b, s)</td>
<td>5</td>
</tr>
<tr>
<td>(d, a, s)</td>
<td>6</td>
</tr>
<tr>
<td>(d, c, a, s)</td>
<td>7</td>
</tr>
</tbody>
</table>

Diagram of a weighted graph with nodes and edges labeled with costs.
Uniform Cost Search

<table>
<thead>
<tr>
<th>state</th>
<th>cost</th>
</tr>
</thead>
<tbody>
<tr>
<td>(d, a, s)</td>
<td>6</td>
</tr>
<tr>
<td>(d, c, a, s)</td>
<td>7</td>
</tr>
<tr>
<td>(g, b, s)</td>
<td>10</td>
</tr>
</tbody>
</table>
Uniform Cost Search

\[ \text{state} \quad \text{cost} \]
\[
\begin{align*}
\langle d, c, a, s \rangle & \quad 7 \\
\langle g, d, a, s \rangle & \quad 8 \\
\langle g, b, s \rangle & \quad 10 
\end{align*}
\]
Uniform Cost Search

<table>
<thead>
<tr>
<th>state</th>
<th>cost</th>
</tr>
</thead>
<tbody>
<tr>
<td>\langle g, d, a, s \rangle</td>
<td>8</td>
</tr>
<tr>
<td>\langle g, d, c, a, s \rangle</td>
<td>9</td>
</tr>
<tr>
<td>\langle g, b, s \rangle</td>
<td>10</td>
</tr>
</tbody>
</table>
Uniform Cost Search

• UCS is an extension of BFS to the weighted-graph case (UCS = BFS if all edges have the same cost)
• UCS is complete and optimal (assuming costs bounded away from zero)
• UCS is guided by path cost rather than path depth, so it may get in trouble if some edge costs are very small.

• Worst-case time and space complexity $O(b^{W*/\epsilon})$, where $W*$ is the optimal cost, and $\epsilon$ is such that all edge weights are no smaller than $\epsilon$.
Greedy (Best-First) Search

- UCS explores paths in all directions, with no bias towards the goal state.

- What if we try to get “closer” to the goal?

- We need a measure of distance to the goal. It would be ideal to use the length of the shortest path... but this is exactly what we are trying to compute!

- We can estimate the distance to the goal through a “heuristic function,” \( h : V \to \mathbb{R}_{\geq 0} \). In motion planning, we can use, e.g., the Euclidean distance to the goal (as the crow flies).

- A reasonable strategy is to always try to move in such a way to minimize the estimated distance to the goal: this is the basic idea of the greedy (best-first) search.
Greedy (Best-First) Search

\[
Q \leftarrow \langle \text{start} \rangle; \quad // \text{Initialize the queue with the starting node}
\]

\[
\text{while } Q \text{ is not empty do}
\]

\[
\quad \text{Pick the path } P \text{ with minimum heuristic cost } h(\text{head}(P)) \text{ from the queue } Q; \\
\quad \text{if } \text{head}(P) = \text{goal} \text{ then return } P; \quad // \text{We have reached the goal}
\]

\[
\quad \text{foreach vertex } v \text{ such that } (\text{head}(P), v) \in E, \text{ do}
\]

\[
\quad \quad \text{add } \langle v, P \rangle \text{ to the queue } Q;
\]

\[
\text{return FAILURE ;} \quad // \text{Nothing left to consider.}
\]
Greedy (Best-First) Search

Q: | path | cost | h |
---|------|-----|---|
(s) | 0    | 10  |   |

![Diagram of a graph with nodes and edges, starting at s and ending at g, with path costs and heuristics labeled.]
Greedy (Best-First) Search

Q:

<table>
<thead>
<tr>
<th>path</th>
<th>cost</th>
<th>h</th>
</tr>
</thead>
<tbody>
<tr>
<td>{a, s}</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>{b, s}</td>
<td>5</td>
<td>3</td>
</tr>
</tbody>
</table>
Greedy (Best-First) Search

<table>
<thead>
<tr>
<th>path</th>
<th>cost</th>
<th>h</th>
</tr>
</thead>
<tbody>
<tr>
<td>(c, a, s)</td>
<td>4</td>
<td>1</td>
</tr>
<tr>
<td>(b, s)</td>
<td>5</td>
<td>3</td>
</tr>
<tr>
<td>(d, a, s)</td>
<td>6</td>
<td>4</td>
</tr>
</tbody>
</table>
Greedy (Best-First) Search

<table>
<thead>
<tr>
<th>path</th>
<th>cost</th>
<th>h</th>
</tr>
</thead>
<tbody>
<tr>
<td>(\langle b, s \rangle)</td>
<td>5</td>
<td>3</td>
</tr>
<tr>
<td>(\langle d, a, s \rangle)</td>
<td>6</td>
<td>4</td>
</tr>
<tr>
<td>(\langle d, c, a, s \rangle)</td>
<td>7</td>
<td>4</td>
</tr>
</tbody>
</table>

Q:
Greedy (Best-First) Search

Q:

<table>
<thead>
<tr>
<th>path</th>
<th>cost</th>
<th>h</th>
</tr>
</thead>
<tbody>
<tr>
<td>\langle g, b, s \rangle</td>
<td>10</td>
<td>0</td>
</tr>
<tr>
<td>\langle d, a, s \rangle</td>
<td>6</td>
<td>4</td>
</tr>
<tr>
<td>\langle d, c, a, s \rangle</td>
<td>7</td>
<td>4</td>
</tr>
</tbody>
</table>

Diagram:

- Start node: s
- Nodes: a, b, c, d, g
- Edges with costs: 2, 3, 4, 5
- Path from s to g:
  - s → b → d → g
  - Total cost: 10
  - heuristic: 5
Greedy (Best-First) Search

- Greedy (Best-First) search is similar in spirit to Depth-First Search: it keeps exploring until it has to back up due to a dead end.

- Greedy search is not complete and not optimal, but is often fast and efficient, depending on the heuristic function $h$.

- Worst-case time and space complexity $O(b^m)$. 

29
The A search algorithm

The problems
- Uniform-Cost search is optimal, but may wander around a lot before finding the goal.
- Greedy search is not optimal, but in some cases it is efficient, as it is heavily biased towards moving towards the goal. The non-optimality comes from neglecting “the past.”

The idea
- Keep track both of the cost of the partial path to get to a vertex, say $g(v)$, and of the heuristic function estimating the cost to reach the goal from a vertex, $h(v)$.
- In other words, choose as a “ranking” function the sum of the two costs:

$$f(v) = g(v) + h(v)$$

- $g(v)$: cost-to-come (from the start to $v$).
- $h(v)$: cost-to-go estimate (from $v$ to the goal).
- $f(v)$: estimated cost of the path (from the start to $v$ and then to the goal).
The A search algorithm

\[ Q \leftarrow \langle \text{start} \rangle; \quad \text{// Initialize the queue with the starting node} \]

\textbf{while} \quad Q \quad \text{is not empty} \quad \textbf{do}

\quad \text{Pick the path } P \text{ with minimum estimated cost } f(P) = g(P) + h(\text{head}(P)) \text{ from the queue } Q; \]

\quad \text{if } \text{head}(P) = \text{goal} \text{ then return } P; \quad \text{// We have reached the goal} \]

\quad \textbf{foreach} \text{ vertex } v \text{ such that } (\text{head}(P), v) \in E, \text{ do} \]

\quad \quad \text{add } \langle v, P \rangle \text{ to the queue } Q; \]

\textbf{return} \text{ FAILURE;} \quad \text{// Nothing left to consider.}
The A search algorithm

Q:

<table>
<thead>
<tr>
<th>path</th>
<th>g</th>
<th>h</th>
<th>f</th>
</tr>
</thead>
<tbody>
<tr>
<td>s</td>
<td>0</td>
<td>10</td>
<td>10</td>
</tr>
</tbody>
</table>

Diagram of a graph with nodes and edges.
The A search algorithm

Q:

\[
\begin{array}{cccc}
\text{path} & g & h & f \\
\{s\} & 0 & 10 & 10 \\
\end{array}
\]
The A search algorithm

<table>
<thead>
<tr>
<th>path</th>
<th>g</th>
<th>h</th>
<th>f</th>
</tr>
</thead>
<tbody>
<tr>
<td>(a, s)</td>
<td>2</td>
<td>2</td>
<td>4</td>
</tr>
<tr>
<td>(b, s)</td>
<td>5</td>
<td>3</td>
<td>8</td>
</tr>
</tbody>
</table>
The A search algorithm

<table>
<thead>
<tr>
<th>path</th>
<th>g</th>
<th>h</th>
<th>f</th>
</tr>
</thead>
<tbody>
<tr>
<td>⟨c, a, s⟩</td>
<td>4</td>
<td>1</td>
<td>5</td>
</tr>
<tr>
<td>⟨b, s⟩</td>
<td>5</td>
<td>3</td>
<td>8</td>
</tr>
<tr>
<td>⟨d, a, s⟩</td>
<td>6</td>
<td>5</td>
<td>11</td>
</tr>
</tbody>
</table>

![Diagram of A search algorithm]
The A search algorithm

<table>
<thead>
<tr>
<th>path</th>
<th>g</th>
<th>h</th>
<th>f</th>
</tr>
</thead>
<tbody>
<tr>
<td>(b, s)</td>
<td>5</td>
<td>3</td>
<td>8</td>
</tr>
<tr>
<td>(d, a, s)</td>
<td>6</td>
<td>5</td>
<td>11</td>
</tr>
<tr>
<td>(d, c, a, s)</td>
<td>7</td>
<td>5</td>
<td>12</td>
</tr>
</tbody>
</table>

Diagram with nodes and edges.
The A search algorithm

<table>
<thead>
<tr>
<th>path</th>
<th>g</th>
<th>h</th>
<th>f</th>
</tr>
</thead>
<tbody>
<tr>
<td>{g, b, s}</td>
<td>10</td>
<td>0</td>
<td>10</td>
</tr>
<tr>
<td>{d, a, s}</td>
<td>6</td>
<td>5</td>
<td>11</td>
</tr>
<tr>
<td>{d, c, a, s}</td>
<td>7</td>
<td>5</td>
<td>12</td>
</tr>
</tbody>
</table>
The A search algorithm

- A search is similar to UCS, with a bias induced by the heuristic $h$. If $h = 0$, $A = UCS$.

- The A search is complete, but is not optimal. What is wrong? (Recall that if $h = 0$ then $A = UCS$, and hence optimal...)

**A* Search**

- Choose an admissible heuristic, i.e., such that $h(v) \leq h^*(v)$. (The star means “optimal.”)

- The A search with an admissible heuristic is called $A^*$, which is guaranteed to be optimal.
Example of A* Search

Q:

<table>
<thead>
<tr>
<th>path</th>
<th>g</th>
<th>h</th>
<th>f</th>
</tr>
</thead>
<tbody>
<tr>
<td>(s)</td>
<td>0</td>
<td>10</td>
<td>10</td>
</tr>
</tbody>
</table>

Diagram:

- Start at node `s`.
- Path to node `g` with distances: `s → b → d → g`.
- Total cost: `6 + 3 + 2 = 11`. 
Example of A* Search

<table>
<thead>
<tr>
<th>path</th>
<th>g</th>
<th>h</th>
<th>f</th>
</tr>
</thead>
<tbody>
<tr>
<td>⟨a, s⟩</td>
<td>2</td>
<td>2</td>
<td>4</td>
</tr>
<tr>
<td>⟨b, s⟩</td>
<td>5</td>
<td>3</td>
<td>8</td>
</tr>
</tbody>
</table>

Diagram:

- Start node (s) with distance 6
- Node a with distance 2
- Node d with distance 1
- Node b with distance 3
- Node g with distance 0

Arrows represent the connections and distances between nodes.
Example of A* Search

<table>
<thead>
<tr>
<th>path</th>
<th>g</th>
<th>h</th>
<th>f</th>
</tr>
</thead>
<tbody>
<tr>
<td>(c, a, s)</td>
<td>4</td>
<td>1</td>
<td>5</td>
</tr>
<tr>
<td>(d, a, s)</td>
<td>6</td>
<td>1</td>
<td>7</td>
</tr>
<tr>
<td>(b, s)</td>
<td>5</td>
<td>3</td>
<td>8</td>
</tr>
</tbody>
</table>

Diagram:

- Start node (s) connected to node b with weight 5.
- Node b connected to node d with weight 2.
- Node d connected to node c with weight 3.
- Node c connected to goal node g with weight 0.
Example of A* Search

Q:

<table>
<thead>
<tr>
<th>path</th>
<th>g</th>
<th>h</th>
<th>f</th>
</tr>
</thead>
<tbody>
<tr>
<td>(d, a, s)</td>
<td>6</td>
<td>1</td>
<td>7</td>
</tr>
<tr>
<td>(b, s)</td>
<td>5</td>
<td>3</td>
<td>8</td>
</tr>
<tr>
<td>(d, c, a, s)</td>
<td>7</td>
<td>1</td>
<td>8</td>
</tr>
</tbody>
</table>

Diagram with nodes and edges showing the A* search process.
Example of A* Search

<table>
<thead>
<tr>
<th>path</th>
<th>g</th>
<th>h</th>
<th>f</th>
</tr>
</thead>
<tbody>
<tr>
<td>(&lt;g, d, a, s&gt;)</td>
<td>8</td>
<td>0</td>
<td>8</td>
</tr>
<tr>
<td>(&lt;b, s&gt;)</td>
<td>5</td>
<td>3</td>
<td>8</td>
</tr>
<tr>
<td>(&lt;d, c, a, s&gt;)</td>
<td>7</td>
<td>1</td>
<td>8</td>
</tr>
</tbody>
</table>
Recap: Agents and Environments

- Uniform-cost orders by path cost, or *backward cost* $g(n)$
- Greedy orders by goal proximity, or *forward cost* $h(n)$

**A* Search** orders by the sum: $f(n) = g(n) + h(n)$