Graduate AI
Lecture 2: Search I

Instructors:
Nihar B. Shah (this time)
J. Zico Kolter

Carnegie Mellon University
EXAMPLE: PATHFINDING

Best route?
EXAMPLE: 8-PUZZLE

Fewest “moves”?
SEARCH PROBLEMS

• A search problem has:
  o States (configurations)
  o Start state and goal states
  o Successors: mapping of states to (action,state,cost) triples

High-level objective: Find minimum-cost path from s to t in a computationally efficient manner
EXAMPLE: PATHFINDING
EXAMPLE: PATHFINDING
EXAMPLE: PATHFINDING
EXAMPLE: PATHFINDING
Graph Representation

Cost $\geq 0$ between each pair of vertices $x$ & $y$

High-level objective: Find minimum-cost path from s to t in a computationally efficient manner
Example: 8-Puzzle
Tree Search

Inputs:
• Problem instance
• “Expansion strategy”

Output:
• Path from s to t
RECALL: BREADTH FIRST SEARCH

- Graph with each cost 1

- Define $g(x) = \text{cost till } x$

- Expansion strategy: Expand node with minimum $g$

```
g = 0
#1
```

```
g = 1
#2
```

```
g = 2
#3
```

```
g = 2
g = 2
```

```
g = 2
```

```
g = 3
```

```
⋯
```
Tree Search

function TREE-SEARCH(problem, strategy)

set of frontier nodes contains the start state of problem

loop

• if there are no frontier nodes then return failure

• choose a frontier node for expansion using strategy

• if the chosen node is t then return the corresponding solution

• else expand the node and add the resulting nodes to the set of frontier nodes
Uniform Cost Search Algorithm

Define $g(x) = \text{cost till } x$

Strategy: Expand node with smallest $g$
UNINFORMED VS. INFORMED

• Uniform cost search uses no information about the problem other than the edge costs
  o “Uninformed” search

• Often we may have more information…
  o “Informed” search
EXAMPLE: PATHFINDING

“Going this direction is generally a good idea”
**Example: 8-Puzzle**

```
5 2
6 1 3
7 8 4
```

```
1 2 3
4 5 6
7 8
```

“Having more blocks in their correct position is generally a good idea”
INFORMED SEARCH

• Additional information: For each vertex \( x \), given \( h(x) = \) heuristic evaluation of cost from \( x \) to \( t \)

\[
\begin{align*}
h_0 &= h(s) = 6 \\
h_1 &= h(c) = 7 \\
h_2 &= h(b) = 6 \\
h_3 &= h(a) = 5 \\
h_4 &= h(d) = 2 \\
h_5 &= h(e) = 1 \\
h_6 &= h(t) = 0
\end{align*}
\]

• \( h(t) = 0 \)
Greedy Search Using Heuristic

- **Strategy:** Expand node with min. value of $h$

![Diagram of graph with nodes and edges labeled with $h$ values.](image)
**A* Tree Search**

- **Strategy:** Expand node with min. value of $f(x) = g(x) + h(x)$

![Graph](image)

- **Question:** Which node is expanded fourth?
A* Tree Search

- Should we stop when we discover a goal?

- No: Only stop when we expand a goal

Slide adapted from Dan Klein
Find minimum-cost path from s to t in a **computationally efficient** manner
A* DOESN’T ALWAYS WORK

- Issue: Good path has pessimistic estimate
- Circumvent this issue by being optimistic!

Slide adapted from Dan Klein
**Admissibility Heuristics**

$h$ is admissible if for all $x$,

$$h(x) \leq h^*(x),$$

where $h^*$ is the cost of the optimal path from $x$ to $t$.

- Aerial distance in pathfinding
- $h \equiv 0$
Optimality of A* Tree Search

Theorem: If the heuristic is admissible, then the path returned by A* tree search has minimum cost.
**Proof**

- Recall: A* stops when goal $t$ is expanded
- For contradiction, assume $t$ with suboptimal path is expanded before $t$ with optimal path
- There is a node $x$ on the optimal path to $t$ that has been discovered but not expanded
- $f(x) = g(x) + h(x)$
  \[ \leq g(x) + h^*(x) \]
  \[ = g(t_{opt}) < g(t_{expanded}) = f(t_{expanded}) \]
- $x$ should have been expanded before $t$! ■

Adapted from Dan Klein
8-puzzle Heuristics

• $h_1$: #tiles in wrong position

• $h_2$: sum of Manhattan distances of tiles from goal

• **Question**: Which of these is admissible?
  
  ◦ Answer: Both

• Heuristic for designing admissible heuristics: relax the problem!
DOMINANCE OF HEURISTICS

• $h$ dominates $h'$ iff $\forall x, h(x) \geq h'(x)$

• $h_1$: #tiles in wrong position

• $h_2$: sum of Manhattan distances of tiles from goal

• **Question:** What is the dominance relation between $h_1$ and $h_2$?
  - **Answer:** $h_2$ dominates $h_1$
8-Puzzle Heuristics

- The following table gives the number of nodes expanded by A* with the two heuristics, averaged over random 8-puzzles, for various solution lengths:

<table>
<thead>
<tr>
<th>Length</th>
<th>$A^*(h_1)$</th>
<th>$A^*(h_2)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>16</td>
<td>1301</td>
<td>211</td>
</tr>
<tr>
<td>18</td>
<td>3056</td>
<td>363</td>
</tr>
<tr>
<td>20</td>
<td>7276</td>
<td>676</td>
</tr>
<tr>
<td>22</td>
<td>18094</td>
<td>1219</td>
</tr>
<tr>
<td>24</td>
<td>39135</td>
<td>1641</td>
</tr>
</tbody>
</table>

- Moral: Good heuristics are crucial!
Tree Search

- Tree search can expand many nodes corresponding to the same state

- In a rectangular grid:
  - Search tree of depth $d$ has $4^d$ leaves
  - There are only $4d$ states at Manhattan distance exactly $d$ from any given state
Graph search is the same as tree search, except that it never expands a node twice.

function GRAPH-SEARCH(problem, strategy)
set of frontier nodes contains the start state of problem
loop
  • if there are no unexpanded frontier nodes then return failure
  • choose an unexpanded frontier node for expansion using strategy, and add it to the expanded set
  • if the node contains a goal then return the corresponding solution
  • else expand the node and add the resulting nodes to the set of frontier nodes, only if not in the expanded set
A* Graph Search

• Does A* graph search always find the optimal path under an admissible heuristic?

• No!

Adapted from Dan Klein
**CONSISTENT HEURISTICS**

- $c(x, y) =$ cost of cheapest path from $x$ to $y$
- $h$ is **consistent** if for every two nodes $x, y$,
  \[ h(x) \leq c(x, y) + h(y) \]

**Question:** What is the relation between admissibility and consistency?

1. Admissible $\Rightarrow$ consistent
2. **Consistent $\Rightarrow$ admissible**
3. They are equivalent
4. They are incomparable

- Set $y = t$ above
- Graph in previous slide
8-puzzle Heuristics, Consistent?

- $h_1$: #tiles in wrong position
- $h_2$: sum of Manhattan distances of tiles from goal

**Poll:** Which of these is consistent?
- Answer: Both

- Heuristic for designing admissible heuristics: relax the problem!

Consistent heuristics yield guarantees for A* graph search (next class)
SUMMARY

• Terminology and algorithms:
  o Search problems
  o Uninformed vs. informed search
  o Tree search, graph search, uniform cost search, greedy, A*
  o Admissible and consistent heuristics

• Theorems:
  o A* tree search is optimal with admissible $h$
  o A* graph search is optimal with consistent $h$

• Big ideas:
  o Don’t be too pessimistic!