CSE 473: Artificial Intelligence
Autumn 2016

Search: Cost & Heuristics

Dan Weld

With slides from
Dan Klein, Stuart Russell, Andrew Moore, Luke Zettlemoyer

Announcements

Project 0: “Warm-up” – due today

Project 1: “Search” - due Friday 10/14
   Start early!

Wed: Guest lecture on heuristics by Travis Mandel
Search thru a
Problem Space / State Space

• Input:
  ▪ Set of states
  ▪ Operators [and costs]
  ▪ Start state
  ▪ Goal state [test]

• Output:
  ▪ Path: start ⇒ a state satisfying goal test
  ▪ [May require shortest path]
  ▪ [Sometimes just need state passing test]

Graduation?

▪ Getting a BS in CSE as a search problem?
  (don’t think too hard)

▪ Space of States
▪ Operators
▪ Initial State
▪ Goal State
### DFS vs BFS

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Complete</th>
<th>Optimal</th>
<th>Time</th>
<th>Space</th>
</tr>
</thead>
<tbody>
<tr>
<td>DFS</td>
<td>$N$ unless finite</td>
<td>$N$</td>
<td>$O(b^m)$</td>
<td>$O(bm)$</td>
</tr>
<tr>
<td>BFS</td>
<td>$Y$</td>
<td>$Y$</td>
<td>$O(b^d)$</td>
<td>$O(b^d)$</td>
</tr>
</tbody>
</table>

Cycle checking in DFS costs exponential memory!

---

**Memory is a Big Limitation!!**

- **Suppose:**
  - 4 GHz CPU
  - 32 GB main memory
  - 100 instructions / expansion
  - 5 bytes / node

  - 40 M expansions / sec
  - Memory filled in ... 3 min
Iterative Deepening Search

- DFS with limit; incrementally grow limit
- Evaluation

![Diagram of Iterative Deepening Search]

Iterative Deepening Search

- DFS with limit; incrementally grow limit
- Evaluation

![Diagram of Iterative Deepening Search]
Iterative Deepening Search

- DFS with limit; incrementally grow limit
- Evaluation
  - Complete?
  - Time Complexity?
  - Space Complexity?

\[ \text{Time Complexity: } O(b^d) \]
\[ \text{Space Complexity: } O(bd) \]

* Assuming branching factor is finite

Important Note: no cycle checking necessary!
Cost of Iterative Deepening

<table>
<thead>
<tr>
<th>b</th>
<th>ratio ID to DFS</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
</tr>
<tr>
<td>5</td>
<td>1.5</td>
</tr>
<tr>
<td>10</td>
<td>1.2</td>
</tr>
<tr>
<td>25</td>
<td>1.08</td>
</tr>
<tr>
<td>100</td>
<td>1.02</td>
</tr>
</tbody>
</table>

Speed Assuming 10M nodes/sec & sufficient memory

<table>
<thead>
<tr>
<th></th>
<th>BFS</th>
<th>Iter. Deep.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Nodes</td>
<td>Time</td>
</tr>
<tr>
<td>8 Puzzle</td>
<td>$10^5$</td>
<td>.01 sec</td>
</tr>
<tr>
<td>2x2x2 Rubik’s</td>
<td>$10^6$</td>
<td>.2 sec</td>
</tr>
<tr>
<td>15 Puzzle</td>
<td>$10^{13}$</td>
<td>6 days</td>
</tr>
<tr>
<td>3x3x3 Rubik’s</td>
<td>$10^{19}$</td>
<td>68k yrs</td>
</tr>
<tr>
<td>24 Puzzle</td>
<td>$10^{25}$</td>
<td>12B yrs</td>
</tr>
</tbody>
</table>

Why the difference?
- Rubik has higher branch factor
- 15 puzzle has greater depth

Slide adapted from Richard Korf presentation
Search Methods

- Depth first search (DFS)
- Breadth first search (BFS)
- Iterative deepening depth-first search (IDS)

- Best first search
- Uniform cost search (UCS)
- Greedy search
- A*
- Iterative Deepening A* (IDA*)
- Beam search
- Hill climbing
Blind vs Heuristic Search

- Costs on Actions
- Heuristic Guidance

Objective: Path with smallest overall cost
What will BFS return?

… finds the shortest path in terms of number of transitions.
It does not find the least-cost path.

Best-First Search

- Generalization of breadth-first search
- Fringe = Priority queue of nodes to be explored
- Cost function $f(n)$ applied to each node
Priority Queue Refresher

- A priority queue is a data structure in which you can insert and retrieve (key, value) pairs with the following operations:

<table>
<thead>
<tr>
<th>Operation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>pq.push(key, value)</td>
<td>inserts (key, value) into the queue.</td>
</tr>
<tr>
<td>pq.pop()</td>
<td>returns the key with the lowest value, and removes it from the queue.</td>
</tr>
</tbody>
</table>

- You can decrease a key’s priority by pushing it again
- Unlike a regular queue, insertions aren’t constant time, usually $O(\log n)$
- We’ll need priority queues for cost-sensitive search methods

Best-First Search

- Generalization of breadth-first search
- Fringe = Priority queue of nodes to be explored
- Cost function $f(n)$ applied to each node

```plaintext
Add initial state to priority queue
While queue not empty
    Node = head(queue)
    If goal?(node) then return node
    Add children of node to queue
```

"expanding the node"
Old Friends

- Breadth First =
  - Best First
  - with \( f(n) = \text{depth}(n) \)

- Dijkstra’s Algorithm (Uniform cost) =
  - Best First
  - with \( f(n) = \text{the sum of edge costs from start to } n \)

Uniform Cost Search

Best first, where
\( f(n) = \text{“cost from start to } n \text{”} \)

aka “Dijkstra’s Algorithm”
Uniform Cost Search

Expansion order:
S, p, d, b, e, a, r, f, e, G

Cost contours (not all shown)

Uniform Cost Search

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Complete</th>
<th>Optimal</th>
<th>Time</th>
<th>Space</th>
</tr>
</thead>
<tbody>
<tr>
<td>DFS w/ Path Checking</td>
<td>Y if finite</td>
<td>N</td>
<td>O(bm)</td>
<td>O(bm)</td>
</tr>
<tr>
<td>BFS</td>
<td>Y</td>
<td>Y*</td>
<td>O(b^2)</td>
<td>O(b^2)</td>
</tr>
<tr>
<td>UCS</td>
<td>Y*</td>
<td>Y</td>
<td>O(b^{C*})</td>
<td>O(b^{C*})</td>
</tr>
</tbody>
</table>

C*/e tiers

C* = Optimal cost
e = Minimum cost of an action
Uniform Cost Issues

- Remember: explores increasing cost contours
- The good: UCS is complete and optimal!
- The bad:
  - Explores options in every “direction”
  - No information about goal location

Uniform Cost: Pac-Man

- Cost of 1 for each action
- Explores all of the states, but one
What is a **Heuristic**?

- An *estimate* of how close a state is to a goal
- Designed for a particular search problem

Examples: Manhattan distance: $10 + 5 = 15$
Euclidean distance: $11.2$

---

What is a **Heuristic**?

- An *estimate* of how close a state is to a goal
- Designed for a particular search problem

- Actual distance to goal: $2 + 4 + 2 + 1 + 8 =$
Greedy Search

*Best first with $f(n) = \text{heuristic estimate of distance to goal}*

Expand the node that seems closest...

What can go wrong?
Greedy Search

- **Common case:**
  - Best-first takes you straight to a (suboptimal) goal

- **Worst-case:** like a badly-guided DFS
  - Can explore everything
  - Can get stuck in loops if no cycle checking

- **Like DFS in completeness**
  - Complete w/ cycle checking
  - *If* finite # states

A* Search

Hart, Nilsson & Rafael 1968

Best first search with $f(n) = g(n) + h(n)$

- $g(n)$ = sum of costs from start to $n$
- $h(n)$ = estimate of lowest cost path $n \rightarrow$ goal
  $$h(\text{goal}) = 0$$

If $h(n)$ is admissible and monotonic
then A* is optimal

- Underestimates ($\leq$) cost of reaching goal from node
- $f$ values never decrease
  - From node to descendants (triangle inequality)
A* Search
Hart, Nilsson & Rafael 1968

Best first search with $f(n) = g(n) + h(n)$
- $g(n) =$ sum of costs from start to $n$
- $h(n) =$ estimate of lowest cost path $n \rightarrow$ goal
  $h(\text{goal}) = 0$

Can view as cross-breed:
- $g(n) \sim$ uniform cost search
- $h(n) \sim$ greedy search

Best of both worlds…

Is Manhattan distance **admissible**?
- **Underestimate?**
Is Manhattan distance monotonic?

- f values increase from node to children
- (triangle inequality)

Monotonicity

\[ F(a) \geq F(b) \]
\[ G(a) + H(a) \geq G(b) + H(b) \]
Euclidean Distance

- Admissible?
- Monotonic?