Finish Search

Computer Science cpsc322, Lecture 10

(Textbook Chpt 3.6)

Sep, 27, 2013
Lecture Overview

• Finish MBA*
• Pruning Cycles and Repeated states Examples
• Dynamic Programming
• Search Recap
Heuristic value by look ahead

What is the most accurate admissible heuristic value for \( n \), given only this info?

A. 7
B. 5
C. 2
D. 8

\[ \min_{i} \left[ \text{cost}(n, n_i) + h(n_i) \right] \]
Memory-bounded $A^*$

- Iterative deepening $A^*$ and B & B use a tiny amount of memory
- What if we've got more memory to use?
- Keep as much of the fringe in memory as we can
- If we have to delete something:
  - Delete the worst paths (with $h(p)$)
  - "Back them up" to a common ancestor

![Diagram with nodes $p_1, p, p_2, \ldots, p_n$ and path $h(p)$]
MBA*: Compute New $h(p)$

A. $\text{New } h(p) = \min \left\{ \max_{i} \left[ \left( \text{cost}(p_{i}) - \text{cost}(p) \right) + h(p_{i}) \right], \text{Old } h(p) \right\}$

B. $\text{New } h(p) = \max \left\{ \min_{i} \left[ \left( \text{cost}(p_{i}) - \text{cost}(p) \right) + h(p_{i}) \right], \text{Old } h(p) \right\}$

C. $\text{New } h(p) = \max \left\{ \max_{i} \left[ \left( \text{cost}(p_{i}) - \text{cost}(p) \right) + h(p_{i}) \right], \text{Old } h(p) \right\}$
Lecture Overview

• Finish MBA*
• Pruning Cycles and Repeated states Examples
• Dynamic Programming
• Search Recap
Multiple-Path Pruning & Optimal Solutions

Problem: what if a subsequent path to $n$ is shorter than the first path to $n$?
• You can remove all paths from the frontier that use the longer path. (as these can’t be optimal)
Problem: what if a subsequent path to $n$ is shorter than the first path to $n$?

- You can change the initial segment of the paths on the frontier to use the shorter path.
Example

Pruning Cycles

Repeated States

neighbors of $n_4 = \{ n_2, n_{11} \}$

neighbors of $n_{10} = \{ n_{15}, n_{16} \}$
Lecture Overview

• Finish MBA*
• Pruning Cycles and Repeated states Examples
• Dynamic Programming
• Search Recap
Dynamic Programming

• Idea: for statically stored graphs, build a table of dist(n):
  • The \textit{actual distance} of the shortest path from node n to a goal g
  • This is the perfect

  • dist(g) = 0
  • dist(z) = 1
  • dist(c) = 3
  • dist(b) = 4
  • dist(k) = ?
  • dist(h) = ?

  • dist(h) = ?

• How could we implement that?
Dynamic Programming

This can be built backwards from the goal:

\[
\text{dist}(n) = \begin{cases} 
0 & \text{if is \_ goal}(n), \\
\min_{(n,m) \in A} (\text{cost}(n, m) + \text{dist}(m)) & \text{otherwise}
\end{cases}
\]

all the neighbors \( m \)

\[
\begin{align*}
\text{dist}(a) &= \min[(3+3), (4+2)] = 3 \\
\text{dist}(b) &= \min[(2+0)] = 2 \\
\text{dist}(c) &= \min[(3+0)] = 3 \\
\text{dist}(g) &= 0
\end{align*}
\]
But there are at least two main problems:

- You need enough space to store the graph.
- The \( \text{dist} \) function needs to be recomputed for each goal.
Lecture Overview

• Finish MBA*
• Pruning Cycles and Repeated states Examples
• Dynamic Programming
• Search Recap
# Recap Search

<table>
<thead>
<tr>
<th>Selection</th>
<th>Complete</th>
<th>Optimal</th>
<th>Time</th>
<th>Space</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>DFS</strong></td>
<td>LIFO</td>
<td>N</td>
<td>N</td>
<td>$O(b^m)$</td>
</tr>
<tr>
<td><strong>BFS</strong></td>
<td>FIFO</td>
<td>Y</td>
<td>Y</td>
<td>$O(b^m)$</td>
</tr>
<tr>
<td><strong>IDS(C)</strong></td>
<td>LIFO</td>
<td>Y</td>
<td>Y</td>
<td>$O(b^m)$</td>
</tr>
<tr>
<td><strong>LCFS</strong></td>
<td>min cost</td>
<td>Y</td>
<td>Y</td>
<td>$O(b^m)$</td>
</tr>
<tr>
<td><strong>BFS</strong></td>
<td>min $h$</td>
<td>N</td>
<td>N</td>
<td>$O(b^m)$</td>
</tr>
<tr>
<td><strong>A</strong></td>
<td>min $f = c+ h$</td>
<td>Y</td>
<td>Y</td>
<td>$O(b^m)$</td>
</tr>
<tr>
<td><strong>B&amp;B</strong></td>
<td>LIFO + pruning</td>
<td>N</td>
<td>Y</td>
<td>$O(b^m)$</td>
</tr>
<tr>
<td><strong>IDA</strong></td>
<td>LIFO</td>
<td>Y</td>
<td>Y</td>
<td>$O(b^m)$</td>
</tr>
<tr>
<td><strong>MBA</strong></td>
<td>min $f$</td>
<td>N</td>
<td>Y</td>
<td>$O(b^m)$</td>
</tr>
</tbody>
</table>
Recap Search (some qualifications)

<table>
<thead>
<tr>
<th></th>
<th>Complete</th>
<th>Optimal</th>
<th>Time</th>
<th>Space</th>
</tr>
</thead>
<tbody>
<tr>
<td>DFS</td>
<td>N</td>
<td>N</td>
<td>$O(b^m)$</td>
<td>$O(mb)$</td>
</tr>
<tr>
<td>BFS</td>
<td>Y</td>
<td>Y</td>
<td>$O(b^m)$</td>
<td>$O(b^m)$</td>
</tr>
<tr>
<td>IDS(C)</td>
<td>Y</td>
<td>Y</td>
<td>$O(b^m)$</td>
<td>$O(mb)$</td>
</tr>
<tr>
<td>LCFS</td>
<td>Y</td>
<td>?</td>
<td>$O(b^m)$</td>
<td>$O(b^m)$</td>
</tr>
<tr>
<td>BFS</td>
<td>N</td>
<td>N</td>
<td>$O(b^m)$</td>
<td>$O(b^m)$</td>
</tr>
<tr>
<td>A*</td>
<td>Y</td>
<td>?</td>
<td>$O(b^m)$</td>
<td>$O(b^m)$</td>
</tr>
<tr>
<td>B&amp;B</td>
<td>N</td>
<td>?</td>
<td>$O(b^m)$</td>
<td>$O(mb)$</td>
</tr>
<tr>
<td>IDA*</td>
<td>Y</td>
<td>Y</td>
<td>$O(b^m)$</td>
<td>$O(mb)$</td>
</tr>
<tr>
<td>MBA*</td>
<td>N</td>
<td>Y</td>
<td>$O(b^m)$</td>
<td>$O(b^m)$</td>
</tr>
</tbody>
</table>
# Search in Practice

<table>
<thead>
<tr>
<th>Method</th>
<th>Complete</th>
<th>Optimal</th>
<th>Time</th>
<th>Space</th>
</tr>
</thead>
<tbody>
<tr>
<td>DFS</td>
<td>N</td>
<td>N</td>
<td>$O(b^m)$</td>
<td>$O(mb)$</td>
</tr>
<tr>
<td>BFS</td>
<td>Y</td>
<td>Y</td>
<td>$O(b^m)$</td>
<td>$O(b^m)$</td>
</tr>
<tr>
<td>IDS(C)</td>
<td>Y</td>
<td>Y</td>
<td>$O(b^m)$</td>
<td>$O(mb)$</td>
</tr>
<tr>
<td>LCFS</td>
<td>Y</td>
<td>Y</td>
<td>$O(b^m)$</td>
<td>$O(b^m)$</td>
</tr>
<tr>
<td>BFS</td>
<td>N</td>
<td>N</td>
<td>$O(b^m)$</td>
<td>$O(b^m)$</td>
</tr>
<tr>
<td>A*</td>
<td>Y</td>
<td>Y</td>
<td>$O(b^m)$</td>
<td>$O(b^m)$</td>
</tr>
<tr>
<td>B&amp;B</td>
<td>N</td>
<td>Y</td>
<td>$O(b^m)$</td>
<td>$O(mb)$</td>
</tr>
<tr>
<td>IDA*</td>
<td>Y</td>
<td>Y</td>
<td>$O(b^m)$</td>
<td>$O(mb)$</td>
</tr>
<tr>
<td>MBA*</td>
<td>N</td>
<td>Y</td>
<td>$O(b^m)$</td>
<td>$O(b^m)$</td>
</tr>
<tr>
<td>BDS</td>
<td>Y</td>
<td>Y</td>
<td>$O(b^{m/2})$</td>
<td>$O(b^{m/2})$</td>
</tr>
</tbody>
</table>
Search in Practice (cont’)

Informed?

Many paths to solution, no infinite paths?

Large branching factor?

F → IDS

T → BFB

T → T

T → T

F → MBA*
(Adversarial) Search: Chess

Deep Blue’s Results in the second tournament:

- second tournament: won 3 games, lost 2, tied 1
- 30 CPUs + 480 chess processors
- Searched 126,000,000 nodes per sec
- Generated 30 billion positions per move reaching depth 14 routinely

- Iterative Deepening with evaluation function (similar to a heuristic) based on 8000 features (e.g., sum of worth of pieces: pawn 1, rook 5, queen 10)
Modules we'll cover in this course: R&Rsys

- Problem
  - Deterministic
    - Arc Consistency
    - Search
  - Stochastic
    - Belief Nets
      - Var. Elimination
  - Logics
    - Search
  - STRIPS
    - Search

- Static
  - Constraint Satisfaction
  - Vars + Constraints

- Sequential
  - Query
  - Planning

- Representation
  - Reasoning Technique

CPSC 322, Lecture 2
Standard Search vs. Specific R&R systems

Constraint Satisfaction (Problems):
- State
- Successor function
- Goal test
- Solution
- Heuristic function

Planning:
- State
- Successor function
- Goal test
- Solution
- Heuristic function

Inference:
- State
- Successor function
- Goal test
- Solution
- Heuristic function
Next class

Start Constraint Satisfaction Problems (CSPs)
Textbook 4.1-4.3

Sorry no office hours today –
may need to change time :-(