Predicting IDA* performance

- What do we need to determine performance?
  - Assume consistent heuristic
    - Analysis has been performed for inconsistent heuristics, but more complicated
  - Assume cost threshold of $c$ on last iteration
    - All nodes with $f(n) < c$ must be expanded
    - All nodes with $f(n) \leq c$ may be expanded
  - Information about the heuristic

Characterizing a heuristic - initial assessment

- Suppose there are $N$ states in the world
  - How many states have $h(a) = a$ for $a$ in $1...N$?
    - Define this as $d(a)$
    - $D(h)$ is: $(\sum d(a)) / N$ for $a = 1...h$
    - The percentage of nodes with heuristic cost less than or equal to $h$

- Heuristic distribution

Characterizing heuristic & search space

- Heuristic distribution may not represent how we exactly encounter states in our search
- This is controlled by the equilibrium distribution
  - What is the actual distribution of states we’ll see in practice
    - A bit strange, because we search from a given start state
    - Cleaned up in later analysis
Equilibrium distribution

• Consider 5-puzzle, blank is:
  • 35.321% of time in side position
  • 64.679% of time in corner position

• Consider the heuristic distribution according to where the blank is

<table>
<thead>
<tr>
<th>h</th>
<th>States</th>
<th>Sum</th>
<th>$D(h)$</th>
<th>Corner</th>
<th>Side</th>
<th>Csum</th>
<th>Ssum</th>
<th>$P(h)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0.002778</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0.02965</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>3</td>
<td>0.008333</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td>0.01028</td>
</tr>
<tr>
<td>2</td>
<td>3</td>
<td>6</td>
<td>0.016667</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>3</td>
<td>0.016915</td>
</tr>
<tr>
<td>3</td>
<td>6</td>
<td>12</td>
<td>0.033333</td>
<td>5</td>
<td>1</td>
<td>8</td>
<td>4</td>
<td>0.033333</td>
</tr>
<tr>
<td>4</td>
<td>12</td>
<td>42</td>
<td>0.166667</td>
<td>25</td>
<td>5</td>
<td>33</td>
<td>9</td>
<td>0.15424</td>
</tr>
<tr>
<td>5</td>
<td>58</td>
<td>100</td>
<td>0.777778</td>
<td>28</td>
<td>20</td>
<td>71</td>
<td>29</td>
<td>0.76701</td>
</tr>
<tr>
<td>6</td>
<td>61</td>
<td>161</td>
<td>4.472222</td>
<td>38</td>
<td>23</td>
<td>109</td>
<td>52</td>
<td>0.446818</td>
</tr>
<tr>
<td>7</td>
<td>58</td>
<td>219</td>
<td>0.608333</td>
<td>41</td>
<td>17</td>
<td>150</td>
<td>69</td>
<td>0.60734</td>
</tr>
<tr>
<td>8</td>
<td>60</td>
<td>279</td>
<td>0.775000</td>
<td>44</td>
<td>16</td>
<td>194</td>
<td>85</td>
<td>0.77507</td>
</tr>
<tr>
<td>9</td>
<td>48</td>
<td>327</td>
<td>0.908333</td>
<td>31</td>
<td>17</td>
<td>225</td>
<td>102</td>
<td>0.906504</td>
</tr>
<tr>
<td>10</td>
<td>24</td>
<td>519</td>
<td>0.795000</td>
<td>11</td>
<td>13</td>
<td>236</td>
<td>115</td>
<td>0.794503</td>
</tr>
<tr>
<td>11</td>
<td>8</td>
<td>359</td>
<td>0.997222</td>
<td>4</td>
<td>4</td>
<td>240</td>
<td>119</td>
<td>0.997057</td>
</tr>
<tr>
<td>12</td>
<td>1</td>
<td>360</td>
<td>1.000000</td>
<td>0</td>
<td>1</td>
<td>240</td>
<td>120</td>
<td>1.000000</td>
</tr>
</tbody>
</table>

Table 2: Heuristic Distributions for Manhattan Distance on the Five Puzzle

Equilibrium Distribution

• $P(h)$
  • The probability of a node having a heuristic $\leq h$
  • $P(h(n) < h \mid \text{side}) \cdot P(\text{side}) +$
  • $P(h(n) < h \mid \text{corner}) \cdot P(\text{corner})$

• Example table with manhattan distance (next slide)
  • $P(h)$ is different from $D(h)$

What does a tree look like?

• Given a node & consistent heuristic:
  • $h$-value is either 1 greater than or 1 less than the original heuristic value

• Divide children into buckets

• Sample tree
  • $h = 0...3$, $c = 5$
What is the total work?

• Expected number of nodes expanded
  \[ E(N, d, P) = \sum_{i=0}^{d} N_i P(d - i) \]

• \( N_i \) is simply \( b^i \) where \( b \) is the asymptotic branching factor

• Using this, what is the heuristic branching factor?

Analysis

• \( E(N, c, P) / E(N, c-1, P) \)
  \[ \sum N(i) P(c-i) / \sum N(i) P(c - i - 1) \]

• \( b^0P(c) + b^1P(c-1) \ldots b^cP(0) \)
  \[ b^0P(c-1) + b^1P(c-2) \ldots b^{c-1}P(0) \]

• \( b^0P(c) \) is less than 1, so insignificant

• factor out \( b \) from the top and get \( b \)

Analysis

• In an exponential domain the effect of a heuristic is to keep the branching factor the same

• If we increase the heuristic, we decrease the level at which we get cutoffs

• In exponential spaces, improving the heuristic just decreases the effective level of search
**Discussion**

- Perfectly predicts nodes expanded for 8-puzzle if:
  - Average over all starting states
  - Search to fixed depth $c$ (past the goal)
- Not necessarily a good predictor for an actual problem

**Interlude**

- Looked at PDBs
  - Work well for very large, implicit state spaces
  - IDA* is the best algorithm for searching these domains
  - PDBs can be inconsistent
  - Return to inconsistency and A*
    - Afterwards look at heuristics more likely to be used with A*

**When A* doesn’t work well**

- What happens when we have inconsistent heuristics?
  - First path to goal is still optimal if admissible
  - First path to other nodes is not necessarily optimal
  - Can re-open closed nodes
Definition

- Let $N$ be the number of nodes expanded by A*
  - $N$ is the number of *nodes* not the number of *expansions*

- $N$ is $O(b^{c/e})$

- A* can re-expand nodes
  - Express in terms of $N$

Analysis

- $(1+0.5*2^N)$ nodes expanded!
  - 6 nodes
  - “E” expanded 8 times
  - “D” expanded 4 times
  - “C” expanded 2 times
  - “A”, “B”, “G” expanded once

Martelli

- Suggested Algorithm “B”
  - Maintain global “F” value
  - Maximum f-value opened so far
  - If there are nodes on OPEN with $f < F$
    - Open in order of increasing g-cost
    - Dijkstra’s algorithm
**Algorithm A**

1. Put start on OPEN, \( g(\text{start}) = 0 \) \( f(\text{start}) \leftarrow h(\text{start}) \)
2. If OPEN is empty return *failure*
3. Remove lowest \( f \)-cost node \( n \) from OPEN
4. If \( n \) is goal, return path from start \( \rightarrow \) goal
5. Expand \( n \) generating successors \( n_1 \ldots n_i \)
6. Add \( n_i \) to OPEN or update costs on OPEN/CLOSED

**Algorithm B**

- Two additional steps:
  1’ \( F \leftarrow 0 \)
  3’ If there is a node in open with \( f < F \), choose among them node with smallest \( g \)-cost. Otherwise set \( F \leftarrow f(n) \)
Martelli

- Algorithm works well, does it fix everything?
  - No -- worst case still $O(n^2)$
  - Just lower cost of start heuristic to 0