Heuristics

CSE 473
University of Washington
473 Topics

- Perception
- NLP
- Robotics
- Multi-agent

- Supervised Learning
- Reinforcement Learning
- MDPs
- Planning
- Uncertainty
- Knowledge Representation
- Search
- Problem Spaces
- Agency

© Daniel S. Weld
Search

✓ Problem spaces
✓ Blind
  ✓ Depth-first, breadth-first, iterative-deepening, iterative broadening
✓ Informed
  ✓ Best-first, Dijkstra's, A*, IDA*, SMA*, DFB&B,
  ✓ Beam, hill climb, limited discrep, RTDP
✓ Local search
  Heuristics
    Evaluation, construction via relaxation
  Pattern databases
  Constraint satisfaction
  Adversary search
Admissable Heuristics

• $f(x) = g(x) + h(x)$
• $g$: cost so far
• $h$: underestimate of remaining costs

Where do heuristics come from?
Relaxed Problems

• Derive admissible heuristic from exact cost of a solution to a relaxed version of problem

  For transportation planning, relax requirement that car has to stay on road \(\rightarrow\) Euclidean dist

• Cost of optimal soln to relaxed problem \(\leq\) cost of optimal soln for real problem
Simplifying Integrals

vertex = formula
goal = closed form formula without integrals
arcs = mathematical transformations

\[ \int x^n \, dx \rightarrow \frac{x^{n+1}}{n+1} \]

heuristic = number of integrals still in formula

what is being relaxed?
Traveling Salesman Problem

- **Problem Space**
  
  States = partial path (not nec. connected)  
  Operator = add an edge  
  Start state = empty path  
  Goal = complete path  

- **Heuristic?**

  What can be Relaxed?
Heuristics for eight puzzle

• What can we relax?
Importance of Heuristics

- $h_1 =$ number of tiles in wrong place
- $h_2 =$ $\sum$ distances of tiles from correct loc

<table>
<thead>
<tr>
<th>D</th>
<th>IDS</th>
<th>$A^*(h_1)$</th>
<th>$A^*(h_2)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>10</td>
<td>6</td>
<td>6</td>
</tr>
<tr>
<td>4</td>
<td>112</td>
<td>13</td>
<td>12</td>
</tr>
<tr>
<td>6</td>
<td>680</td>
<td>20</td>
<td>18</td>
</tr>
<tr>
<td>8</td>
<td>6384</td>
<td>39</td>
<td>25</td>
</tr>
<tr>
<td>10</td>
<td>47127</td>
<td>93</td>
<td>39</td>
</tr>
<tr>
<td>12</td>
<td>364404</td>
<td>227</td>
<td>73</td>
</tr>
<tr>
<td>14</td>
<td>3473941</td>
<td>539</td>
<td>113</td>
</tr>
<tr>
<td>18</td>
<td>3056</td>
<td>363</td>
<td>363</td>
</tr>
<tr>
<td>24</td>
<td>39135</td>
<td>1641</td>
<td>1641</td>
</tr>
</tbody>
</table>
Need More Power!

Performance of Manhattan Distance Heuristic

- 8 Puzzle: < 1 second
- 15 Puzzle: 1 minute
- 24 Puzzle: 65000 years

Need even better heuristics!

© Daniel S. Weld

Adapted from Richard Korf presentation
Subgoal Interactions

- **Manhattan distance assumes**
  Each tile can be moved independently of others
- **Underestimates because**
  Doesn’t consider interactions between tiles
Pattern Databases

[Culberson & Schaeffer 1996]

• Pick any subset of tiles
  • E.g., 3, 7, 11, 12, 13, 14, 15

• Precompute a table
  Optimal cost of solving just these tiles
  For all possible configurations
  • 57 Million in this case

Use breadth first search back from goal state
• State = position of just these tiles (& blank)
Using a Pattern Database

- As each state is generated
  Use position of chosen tiles as index into DB
  Use lookup value as heuristic, h(n)

Admissible?
Combining Multiple Databases

• Can choose another set of tiles
  Precompute multiple tables
• How combine table values?

• E.g. Optimal solutions to Rubik’s cube
  First found w/ IDA* using pattern DB heuristics
  Multiple DBs were used (dif subsets of cubies)
  Most problems solved optimally in 1 day
  Compare with 574,000 years for IDDFS
Drawbacks of Standard Pattern DBs

• Since we can only take $max$ Diminishing returns on additional DBs

• Would like to be able to add values
Disjoint Pattern DBs

- **Partition tiles into disjoint sets**
  
  For each set, precompute table
  - E.g. 8 tile DB has 519 million entries
  - And 7 tile DB has 58 million

- **During search**
  
  Look up heuristic values for each set
  
  *Can add values without overestimating!*

  Manhattan distance is a special case of this idea where each set is a single tile
Performance

• **15 Puzzle**: 2000x speedup vs Manhattan dist
  IDA* with the two DBs shown previously solves 15 Puzzles optimally in 30 milliseconds

• **24 Puzzle**: 12 million x speedup vs Manhattan
  IDA* can solve random instances in 2 days.
  Requires 4 DBs as shown
  • Each DB has 128 million entries
  Without PDBs: 65000 years