CSE 473: Artificial Intelligence
Spring 2018

Heuristics & Pattern Databases for Search

Steve Tanimoto

With thanks to Dan Weld, Dan Klein, Richard Korf, Stuart Russell, Andrew Moore, and Luke Zettlemoyer
Recap: Search Problem

- **States**
  - configurations of the world
- **Successor function:**
  - function from states to lists of (state, action, cost) triples
- **Start state**
- **Goal test**
N-Queens as Search?

- Given N x N chess board
- Can you place N queens so they don’t fight?

Cool picture from Dan Klein & Pieter Abeel ai.berkeley.edu
States are Board Positions

Etc…
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

- Stochastic Search
- Constraint Satisfaction
IDA* for N-Queens?

- Given N x N chess board
- Can you place N queens so they don’t fight?

Cool picture from Dan Klein & Pieter Abeel ai.berkeley.edu
Best-First Search

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

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
Iterative-Deepening A*

- Like iterative-deepening depth-first, but...
- Depth bound modified to be an $f$-limit
  - Start with $f$-limit = $h$(start)
  - Prune any node if $f$(node) > $f$-limit
  - Next $f$-limit = min-cost of any node pruned
IDA* Analysis

- Complete & Optimal (a la A*)
- Space usage $\propto$ depth of solution
- Each iteration is DFS - no priority queue!
- # nodes expanded relative to A*
  - Depends on # unique values of heuristic function
  - In 8 puzzle: few values $\Rightarrow$ close to # A* expands
  - In eastern-europe travel: each f value is unique
    $\Rightarrow$ $1+2+\ldots+n = O(n^2)$ where $n=$nodes A* expands
    if $n$ is too big for main memory, $n^2$ is too long to wait!
- Generates duplicate nodes in cyclic graphs
Beam Search

- **Idea**
  - Best first
  - But discard all but N best items on priority queue

- **Evaluation**
  - Complete?
    - No
  - Time Complexity?
    - $O(b^d)$
  - Space Complexity?
    - $O(b + N)$
Hill Climbing

**Idea**
- Always choose best child; no backtracking
- Beam search with $|\text{queue}| = 1$

**Problems?**
- Local maxima
- Plateaus
- Diagonal ridges

"Gradient ascent"

© Daniel S. Weld
Heuristics

It’s what makes search actually work
Admissible 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 blocks world, distance = # move operations
  - heuristic = number of misplaced blocks
  - *What is relaxed problem?*

- Cost of optimal soln to relaxed problem $\leq$ cost of optimal soln for real problem

© Daniel S. Weld
What’s being relaxed?
Heuristic = Euclidean distance
Traveling Salesman Problem

Objective: shortest path visiting every city

What can be Relaxed?
Heuristics for eight puzzle

- What can we relax?

\[ h_1 = \text{number of tiles in wrong place} \]
\[ h_2 = \sum \text{distances of tiles from correct loc} \]
## Importance of Heuristics

$h_1 = \text{number of tiles in wrong place}$

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

| 7  | 2    | 3          |
| 4  | 1    | 6          |
| 8  | 5    |            |
## Importance of Heuristics

$h1 = \text{number of tiles in wrong place}$

$h2 = \sum \text{distances of tiles from correct loc}$

<table>
<thead>
<tr>
<th>D</th>
<th>IDS</th>
<th>A*(h1)</th>
<th>A*(h2)</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></td>
<td>3056</td>
<td>363</td>
</tr>
<tr>
<td>24</td>
<td></td>
<td>39135</td>
<td>1641</td>
</tr>
</tbody>
</table>

Decrease effective branching factor
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

© Daniel S. Weld

Adapted from Richard Korf presentation
Pattern Databases

- Pick any subset of tiles
  - E.g., 3, 7, 11, 12, 13, 14, 15
  - (or as drawn)

- Precompute a table
  - Optimal cost of solving just these tiles
  - For all possible configurations
    - 57 Million in this case
  - Use A* or IDA*
    - State = position of just these tiles (& blank)

© Daniel S. Weld
Adapted from Richard Korf presentation
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 cubie subsets )
  - Most problems solved optimally in 1 day
  - Compare with 574,000 years for IDDFS

© Daniel S. Weld
Adapted from Richard Korf presentation
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: 65,000 years