CSE 473: Artificial Intelligence
Spring 2018

Heuristics & Pattern Databases for Search

Steve Tanimoto

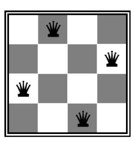
With thanks to Dan Weld, Dan Klein, Richard Korf, Stuart Russell, Andrew Moore, and Luke Zettlemoyer

Recap: Search Problem

- States
  - configurations of the world
- Successor function:
  - function from states to lists of (state, action, cost) triples
- Start state
- Goal test

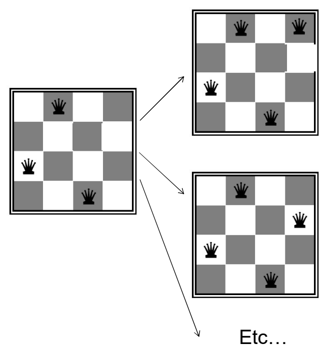
N-Queens as Search?

- 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 3

States are Board Positions



Etc…

4

Search Methods

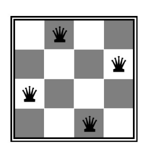
- Depth first search (DFS)
- Breadth first search (BFS)
- Iterative deepening depth-first search (IDS)
- Best first search
- Uniform cost search (UCS)
- Greedy search
- A*
- Iterative Deepening A* (IDA*)
- Beam search, hill climbing
- **Stochastic Search**
- **Constraint Satisfaction**

Heuristic search

5

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 6

Best-First Search

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

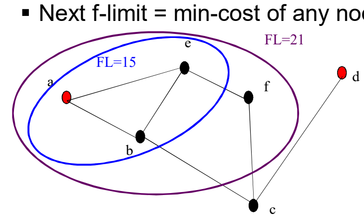
```
Add initial state to priority queue
While queue not empty
    Node = head(queue)
    If goal?(node) then return node
    Add children of node to queue
```

"expanding the node"

7

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



8

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+…+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

9

Beam Search

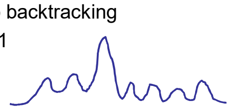
- Idea
  - Best first
  - But discard all but N best items on priority queue
- Evaluation
  - Complete?
    No
  - Time Complexity?
    O(b^d)
  - Space Complexity?
    O(b + N)

14

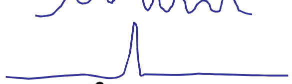
Hill Climbing

"Gradient ascent"

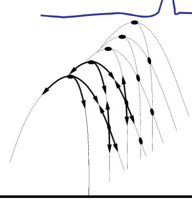
- Idea
  - Always choose best child; no backtracking
  - Beam search with |queue| = 1
- Problems?
  - Local maxima



  - Plateaus



  - Diagonal ridges



15

Heuristics

It's what makes search actually work

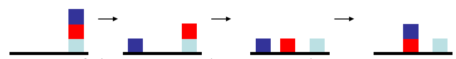
Admissible Heuristics

- 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?*

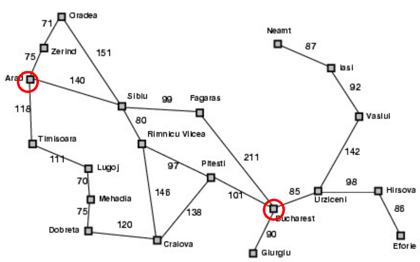


out of place = 2,   true distance to goal = 3

- Cost of optimal soln to relaxed problem ≤ cost of optimal soln for real problem

What's being relaxed?

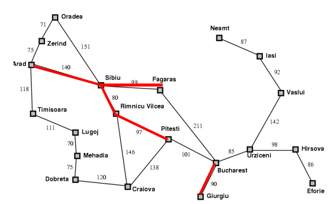
Heuristic = Euclidean distance



Straight–line distance to Bucharest	
Arad	366
Bucharest	0
Craiova	160
Dobreta	242
Eforie	161
Fagaras	176
Giurgiu	77
Hirsova	151
Iasi	226
Lugoj	244
Mehadia	241
Neamt	234
Oradea	380
Pitesti	10
Rimnicu Vilcea	193
Sibiu	253
Timisoara	329
Urziceni	80
Vaslui	199
Zerind	374

Traveling Salesman Problem

Objective: shortest path visiting every city



What can be Relaxed?

Heuristics for eight puzzle

7	2	3
5	1	6
8	3	

→

1	2	3
4	5	6
7	8	

start   goal

- What can we relax?

h1 = number of tiles in wrong place

h2 = Σ distances of tiles from correct loc

Importance of Heuristics

7	2	3
4	1	6
8	5	

h1 = number of tiles in wrong place

D	IDS	A*(h1)
2	10	6
4	112	13
6	680	20
8	6384	39
10	47127	93
12	364404	227
14	3473941	539
18		3056
24		39135

Importance of Heuristics

7	2	3
4	1	6
8	5	

h1 = number of tiles in wrong place

h2 = Σ distances of tiles from correct loc

D	IDS	A*(h1)	A*(h2)
2	10	6	6
4	112	13	12
6	680	20	18
8	6384	39	25
10	47127	93	39
12	364404	227	73
14	3473941	539	113
18		3056	363
24		39135	1641

Decrease effective branching factor

24

Need More Power!

Performance of Manhattan Distance Heuristic

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

Need even better heuristics!

 Adapted from Richard Korf presentation 25

Subgoal Interactions

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

1	2	3
4	6	5
7	8	

 Adapted from Richard Korf presentation 26

Pattern Databases

[Culberson & Schaeffer 1996]

1	2	3	4
5	6	7	8
9	10	11	12
13	14	15	

- 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)

 Adapted from Richard Korf presentation 27

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?

 Adapted from Richard Korf presentation 28

Combining Multiple Databases

1	2	3	4
5	6	7	8
9	10	11	12
13	14	15	

- 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

 Adapted from Richard Korf presentation 29

Drawbacks of Standard Pattern DBs

- Since we can only take *max*
  - Diminishing returns on additional DBs
- Would like to be able to ***add*** values

Adapted from Richard Korf presentation

Disjoint Pattern DBs

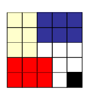
1	2	3	4
5	6	7	8
9	10	11	12
13	14	15	

- 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

Adapted from Richard Korf presentation

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



Adapted from Richard Korf presentation