CSE 573: Artificial Intelligence

Search: Heuristics and Pattern DBs

Dan Weld

With slides from
Dan Weld, Dan Klein, Stuart Russell, Travis Mandel, Andrew Moore, Luke Zettlemoyer

Search thru a
Problem Space / State Space

- Input:
  - Set of states
  - Operators [and costs]
  - Start state
  - Goal state [test]

- Output:
  - Path: start $\Rightarrow$ a state satisfying goal test
  - [May require shortest path]
  - [Sometimes just need state passing test]

Heuristics

It's what makes search actually work

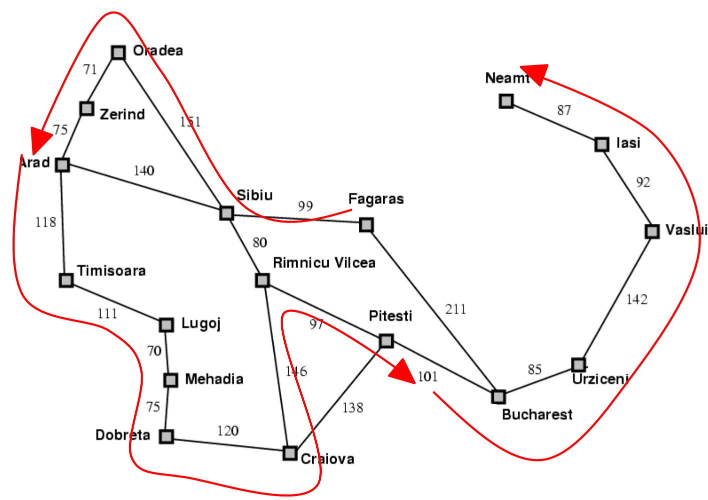
Traveling Salesman

- Input: undirected graph
- Output: connected path traversing each vertex ***exactly*** once
- As a search problem
  - States?

    Graphs w/ partial paths

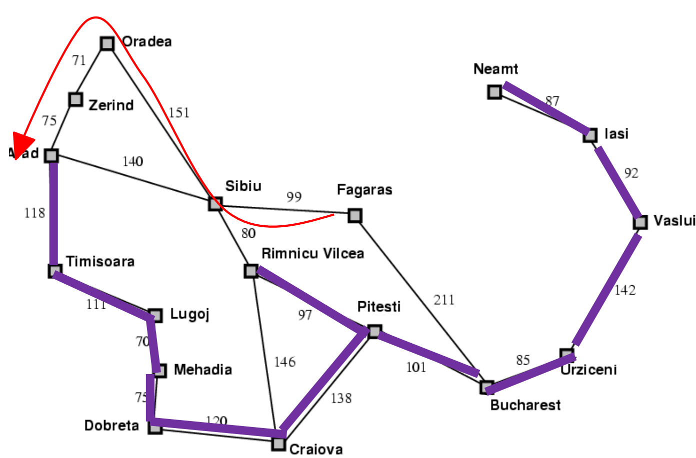
  - Operators?

    Adding a edge to the path



Traveling Salesman

- Input: undirected graph
- Output: connected path traversing each vertex ***exactly*** once
- As a search problem
  - States? Graphs w/ partial paths
  - Operators? Adding a edge to the path
  - Heuristic estimate of cost to complete a path?
    - What to relax?
    - What is a path?
      - Subgraph…
      - Degree 2
    - Min spanning tree
      - $O(n^2)$



Heuristics for eight puzzle

7	2	3
5	1	6
8	3	

→

1	2	3
4	5	6
7	8	

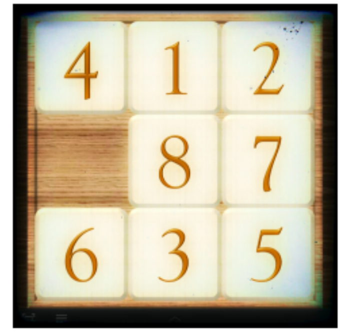
start goal

- What can we relax?

h1 = number of tiles in wrong place

h2 = Σ distances of tiles from correct loc

Relaxed Problem



- Can describe move as a Strips operator
- Predicates:

  On(x,y)   tile x is on cell y

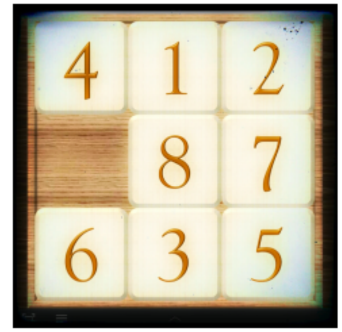
  Clear(y)   no tiles are on cell y

  Adj(y, z)   cell y is adjacent to cell z
- States are conjunctions, eg initial state:

  On(6,1-1), On(3, 2-1), …, Clear(1-2), Adj(1-1, 1-2), Adj(…
- Move(x,y,z)
  - Preconditions: on(x,y), clear(z), ~~adj(y,z)~~
  - Add-list: on(x,z), clear(y)
  - Delete-list: on(x,y), clear(z)

Num misplaced tiles

Relaxed Problem



- Can describe move as a Strips operator
- Predicates:

  On(x,y) tile x is on cell y

  Clear(y) no tiles are on cell y

  Adj(y, z) cell y is adjacent to cell z
- States are conjunctions, eg initial state:

  On(6,1-1), On(3, 2-1), …, Clear(1-2), Adj(1-1, 1-2), Adj(…
- Move(x,y,z)
  - Preconditions: on(x,y), ~~clear(z)~~, adj(y,z)
  - Add-list: on(x,z), clear(y)
  - Delete-list: on(x,y), clear(z)

Sum of Manhattan distances

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

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

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

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

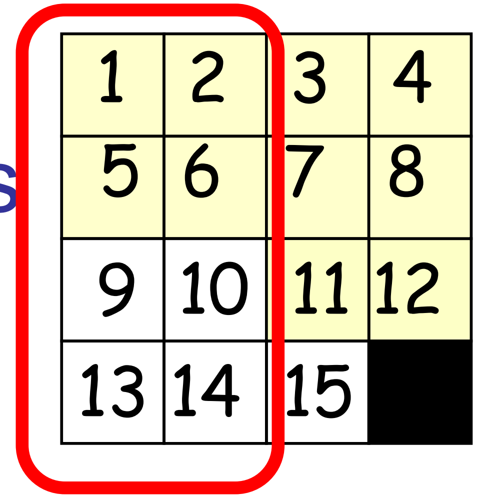
Using a Pattern Database

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

  - Admissible?
  - Monotonic?

Adapted from Richard Korf presentation

Combining Multiple Databases



- Can choose another set of tiles
  - Precompute multiple tables
- How combine table values?
  - Min, Max, Sum, RandomlyChoose
- 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

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	

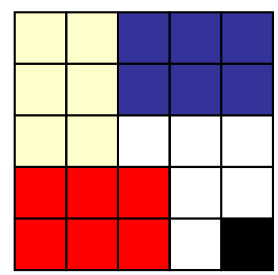
- Partition tiles into ***disjoint*** sets
  - For each set, precompute table
    - E.g. 8 tile DB has 519 million entries
    - And 7 tile DB has 58 million
- During search
  - Look up heuristic values for each set
  - *Can add values without overestimating!*

  - 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

Alternative Approach…

- Optimality is nice to have, but…
- Sometimes space is too vast! Find suboptimal solution using local search.