CSE 473: Artificial Intelligence
Autumn 2016

Search: Heuristics and Pattern DBs

Travis Mandel
(subbing for Dan Weld)

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

Announcements

P0: You're good unless you saw an email from us

Now in More 220!

Project 1: "Search" - due Friday 10/14
Should have started by now!

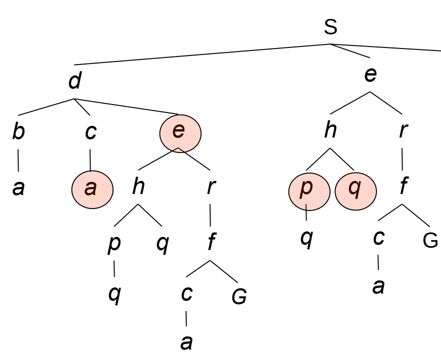
Dan will be back Friday!

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]

Tree vs Graph search

- In BFS, for example, we shouldn't bother expanding the circled nodes (why?)



Graph Search

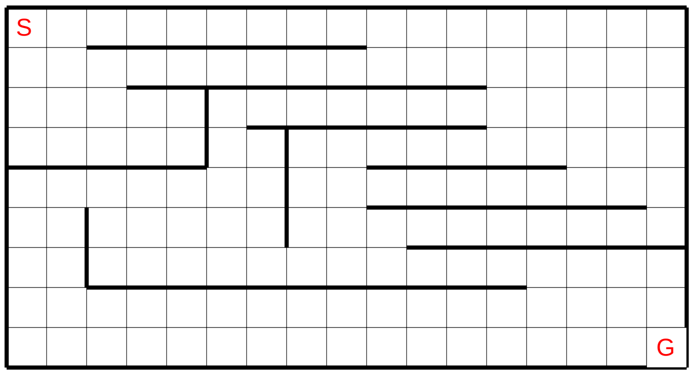
- Very simple fix: never expand a state type twice

```
function GRAPH-SEARCH( problem, fringe) returns a solution, or failure
   closed ← an empty set
   fringe ← INSERT(MAKE-NODE(INITIAL-STATE[problem]), fringe)
   loop do
      if fringe is empty then return failure
      node ← REMOVE-FRONT(fringe)
      if GOAL-TEST(problem, STATE[node]) then return node
      if STATE[node] is not in closed then
         add STATE[node] to closed
         fringe ← INSERTALL(EXPAND(node, problem), fringe)
   end
```

Some Hints

- Graph search is almost always better than tree search
- Implement your closed list as a dict or set!
- Space huge concern!

Search with Heuristics



10

A* Search

Hart, Nilsson & Rafael 1968

Best first search with $f(n) = g(n) + h(n)$

- $g(n)$ = sum of costs from start to n
- $h(n)$ = estimate of lowest cost path n $\rightarrow$ goal

  h(goal) = 0

A* Search

Hart, Nilsson & Rafael 1968

Best first search with f(n) = g(n) + h(n)

- g(n) = sum of costs from start to n
- h(n) = estimate of lowest cost path n → goal

  h(goal) = 0

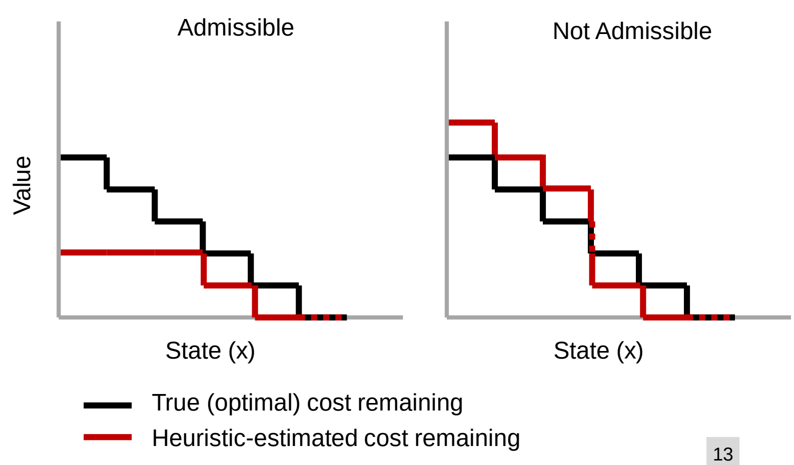
Can view as cross-breed:

g(n) ~ uniform cost search

h(n) ~ greedy search

Best of both worlds…

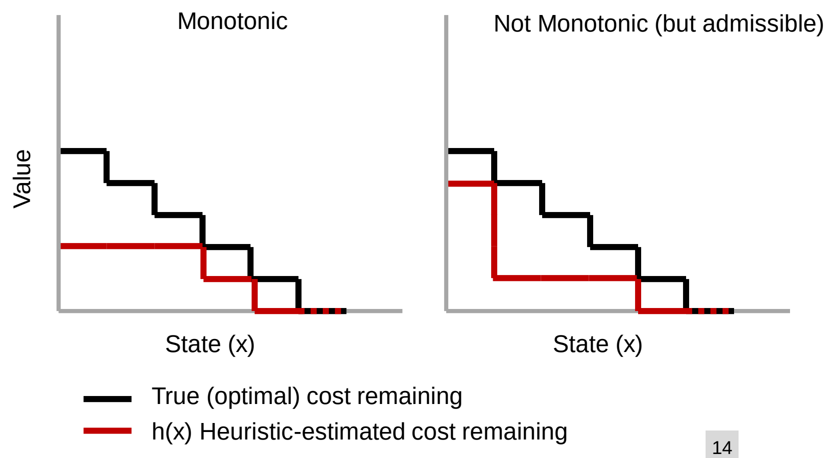
Admissible Heuristics



13

Slide credit: Travis Mandel

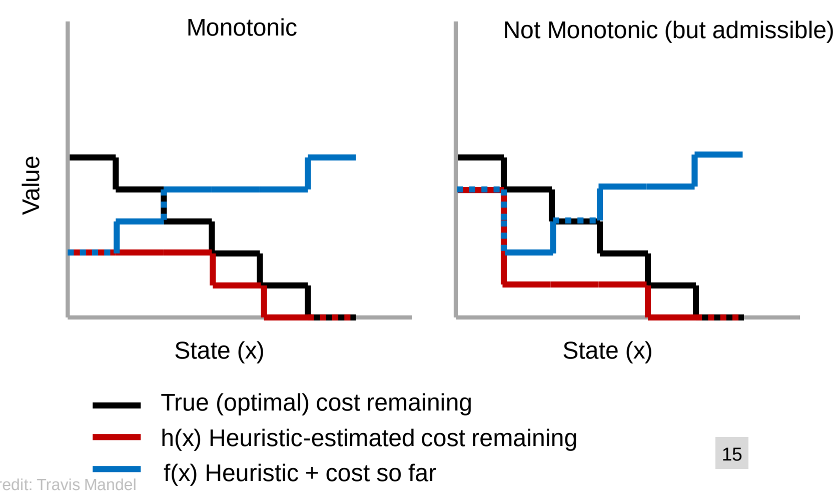
Monotonic/Consistent Heuristics



Slide credit: Travis Mandel

14

Monotonic/Consistent Heuristics

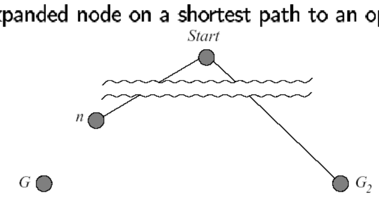


Slide credit: Travis Mandel

15

Optimality of A* (tree search)

Suppose some suboptimal goal G_2 has been generated and is in the queue.
Let n be an unexpanded node on a shortest path to an optimal goal G_1.



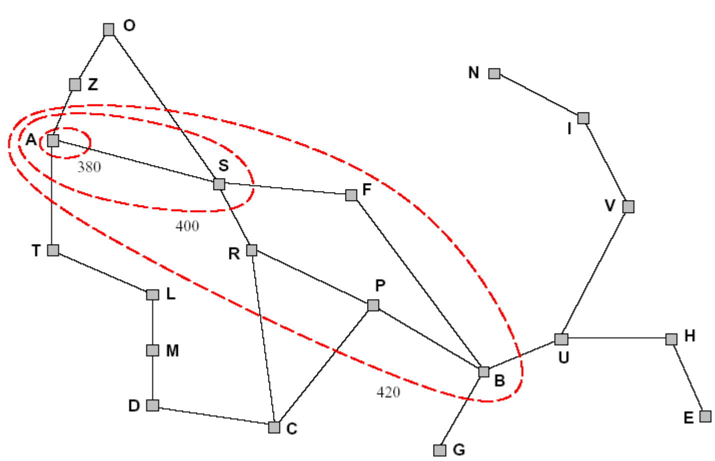
$$\begin{aligned} f(G_2) &= g(G_2) && \text{since } h(G_2) = 0 \\ &> g(G_1) && \text{since } G_2 \text{ is suboptimal} \\ &\geq f(n) && \text{since } h \text{ is admissible} \end{aligned}$$

Since $f(G_2) > f(n)$, A* will never select G_2 for expansion

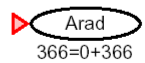
Optimality Continued

Lemma: A* expands nodes in order of increasing f value*

Gradually adds "f-contours" of nodes (cf. breadth-first adds layers)
Contour i has all nodes with $f = f_i$, where $f_i < f_{i+1}$

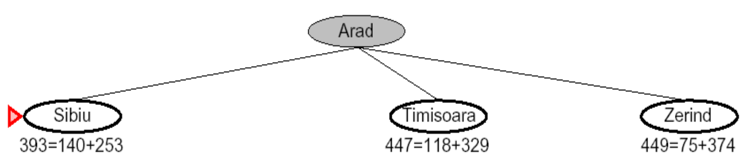


A* Example



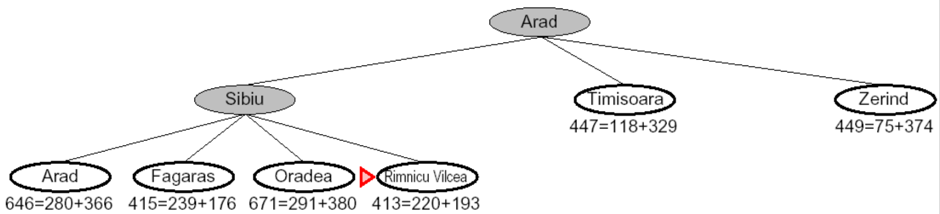
18

A* Example



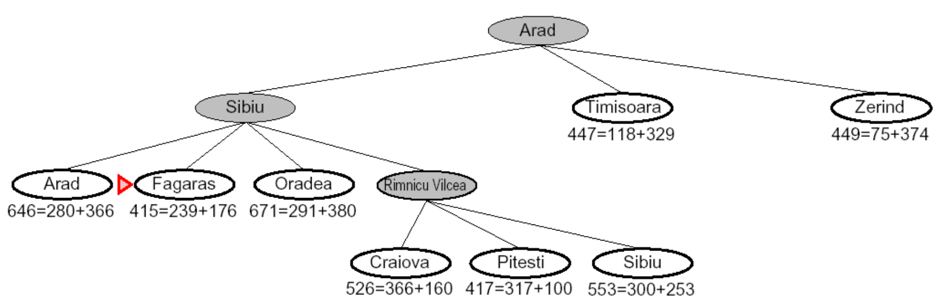
19

A* Example



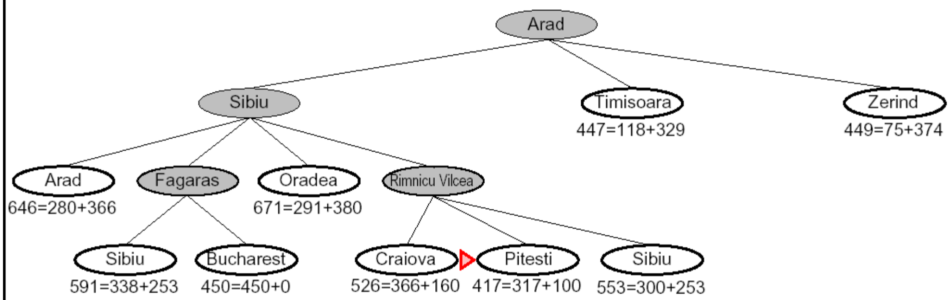
20

A* Example



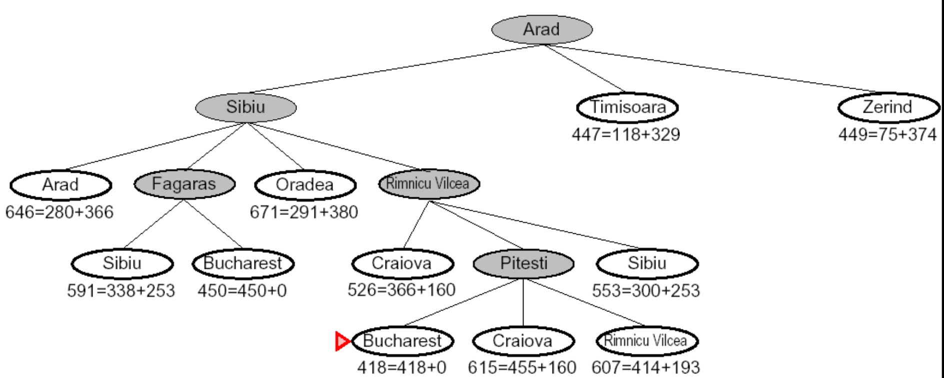
21

A* Example



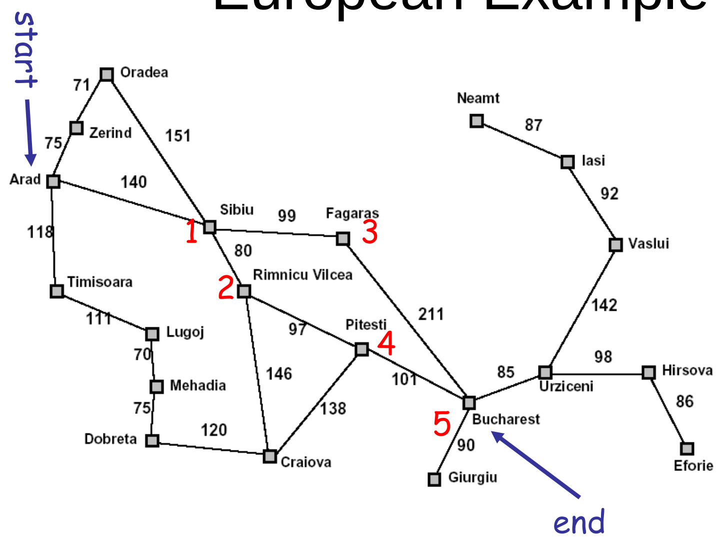
22

A* Example



23

European Example



Straight−line distance to Bucharest

Arad	366
Bucharest	0
Craiova	160
Dobreta	242
Eforie	161
Fagaras	178
Giurgiu	77
Hirsova	151
Iasi	226
Lugoj	244
Mehadia	241
Neamt	234
Oradea	380
Pitesti	98
Rimnicu Vilcea	193
Sibiu	253
Timisoara	329
Urziceni	80
Vaslui	199
Zerind	374

24

A* Summary

- Pros

  Produces optimal cost solution!

  Does so quite quickly (focused)

- Cons

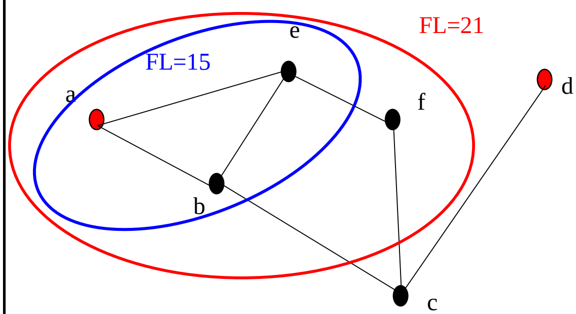
  Maintains priority queue…

  Which can get exponentially big ☹

25

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



26

IDA* Analysis

- Complete & Optimal (ala A*)
- Space usage ∝ 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 ⇒ close to # A* expands
  - In traveling salesman: each f value is unique

    ⇒ $1+2+\ldots+n = O(n^2)$ where n=nodes A* expands

    if n is too big for main memory, n^2 is too long to wait!

28

Forgetfulness

- A* used exponential memory
- How much does IDA* use?
  - During a run?
  - In between runs?
    - SMA*

© Daniel S. Weld 29

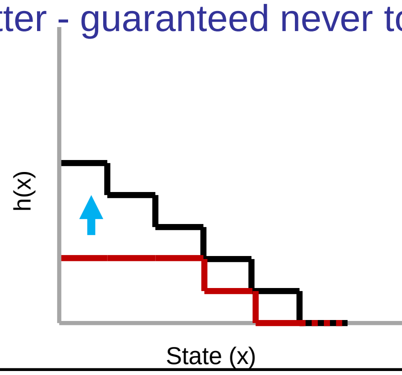
Heuristics

It's what makes search actually work

Dominance

If $h_2(n) \geq h_1(n)$ for all n (both admissible)
then h_2 dominates h_1

h_2 is better - guaranteed never to expand more nodes.



Admissable Heuristics

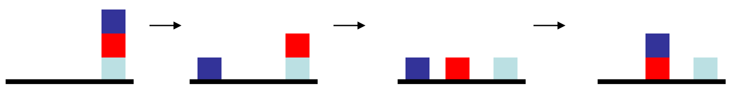
- f(x) = g(x) + h(x)
- g: cost so far
- h: underestimate of remaining costs

Where do heuristics come from?

© Daniel S. Weld

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?*



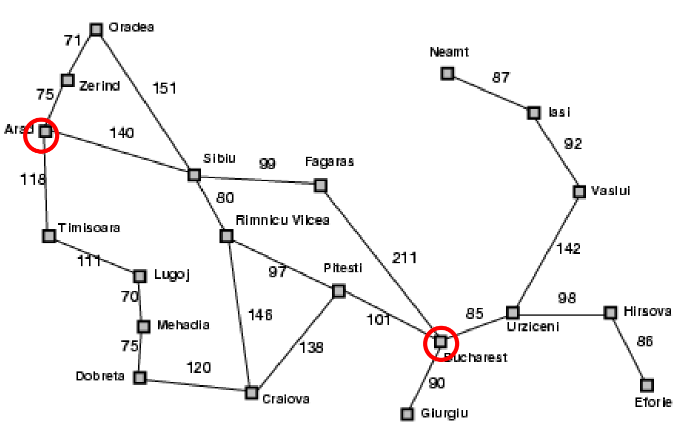
out of place = 2, true distance to goal = 3

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

© Daniel S. Weld

What's being relaxed?

Heuristic = Euclidean distance

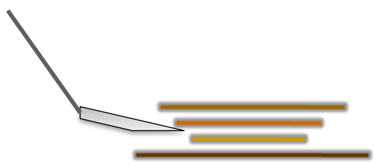


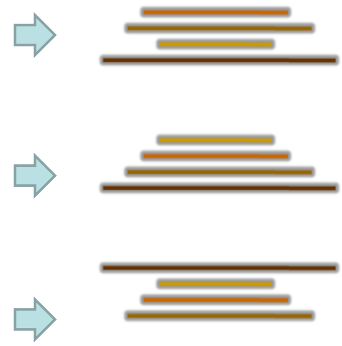
Straight–line distance to Bucharest

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

Example: Pancake Problem

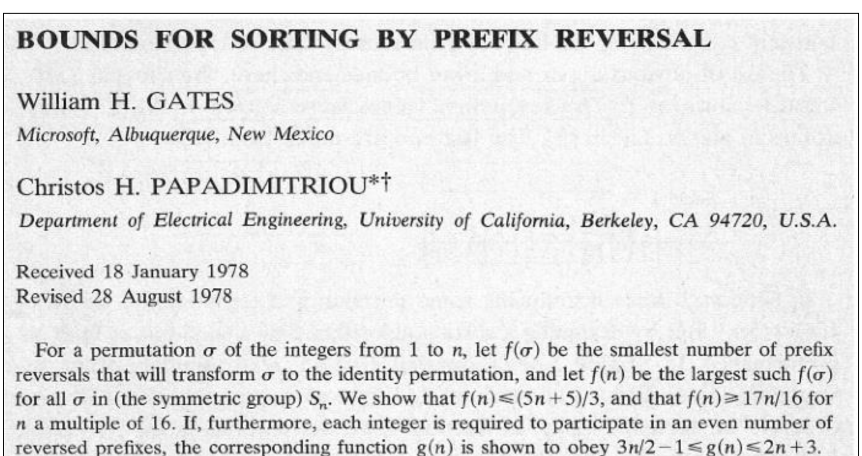
Action: Flip over the top *n* pancakes





Cost: Number of pancakes flipped

Example: Pancake Problem



BOUNDS FOR SORTING BY PREFIX REVERSAL

William H. GATES
Microsoft, Albuquerque, New Mexico

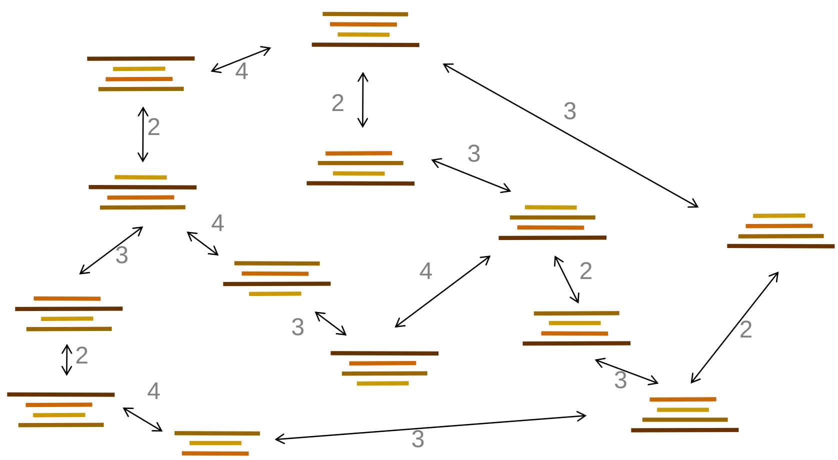
Christos H. PAPADIMITRIOU*†
Department of Electrical Engineering, University of California, Berkeley, CA 94720, U.S.A.

Received 18 January 1978
Revised 28 August 1978

For a permutation σ of the integers from 1 to n, let $f(\sigma)$ be the smallest number of prefix reversals that will transform σ to the identity permutation, and let $f(n)$ be the largest such $f(\sigma)$ for all σ in (the symmetric group) S_n. We show that $f(n) \leqslant (5n+5)/3$, and that $f(n) \geqslant 17n/16$ for n a multiple of 16. If, furthermore, each integer is required to participate in an even number of reversed prefixes, the corresponding function $g(n)$ is shown to obey $3n/2-1 \leqslant g(n) \leqslant 2n+3$.

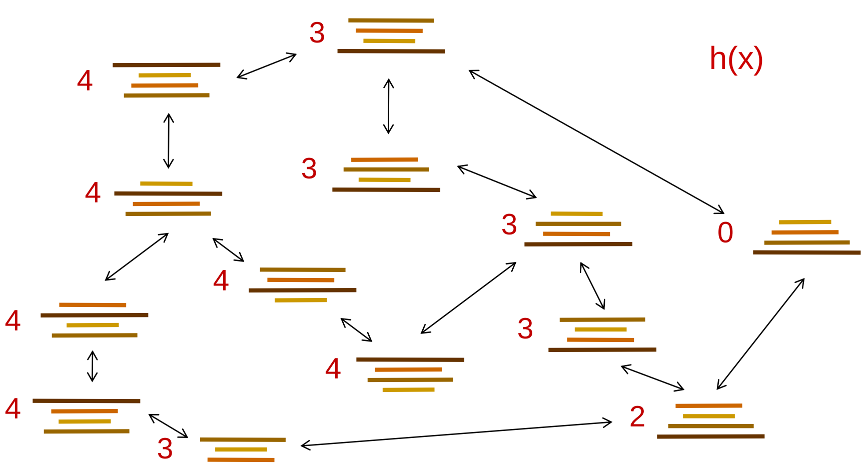
Example: Pancake Problem

State space graph with costs as weights



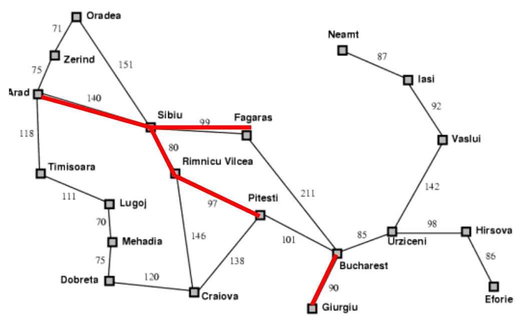
Pancake Heuristic?

Heuristic: the largest pancake that is still out of place



Traveling Salesman Problem

Objective: shortest path visiting every city



What can be Relaxed?

Groundedness.
If can fly to previously seen city → minimum spanning tree

43

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

44

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

45

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

46

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

1	2	3
4	6	5
7	8	

© Daniel S. Weld

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)

© Daniel S. Weld

Adapted from Richard Korf presentation

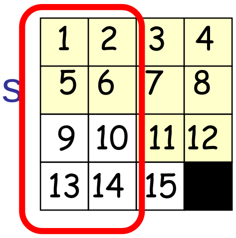
Using a Pattern Database

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

© Daniel S. Weld

Adapted from Richard Korf presentation

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

51

© 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

52

© Daniel S. Weld

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

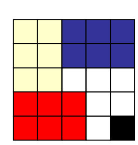
53

© Daniel S. Weld

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



54

© Daniel S. Weld

Adapted from Richard Korf presentation

Alternative Approach…

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

© Daniel S. Weld

Beam Search

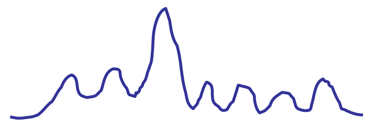
- Idea
  - Best first but only keep N best items on priority queue
- Evaluation
  - Complete?
  - Time Complexity?
  - Space Complexity?

© Daniel S. Weld

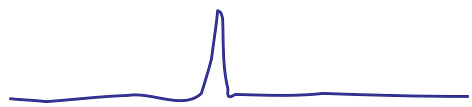
Hill Climbing

"Gradient ascent"

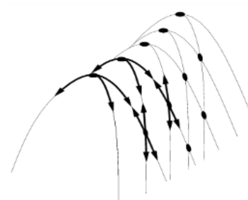
- Idea
  - Always choose best child; no backtracking
  - Beam search with |queue| = 1
- Problems?
  - Local maxima



  - Plateaus



  - Diagonal ridges



© Daniel S. Weld

58