CSE 473

Lecture 5

Heuristics

Last Time: A* Search

- Use an evaluation function f(n) for node n.

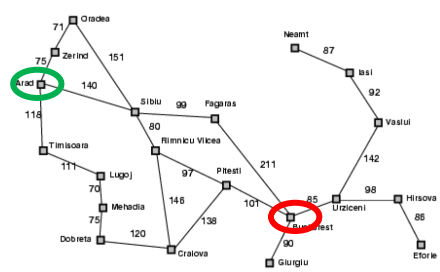
  f(n) = estimated total cost of path thru *n* to goal

- *f(n) = g(n) + h(n)*
  - *g(n)* = cost so far to reach *n*
  - *h(n)* = estimated cost from *n* to goal

- Always choose the node from frontier that has the lowest f value.

  Frontier = priority queue

Problem: Search for shortest path from start to goal



Admissible Heuristics

- **A heuristic *h(n)* is admissible if for every node *n*,**

$$h(n) \leq h^{*}(n)$$

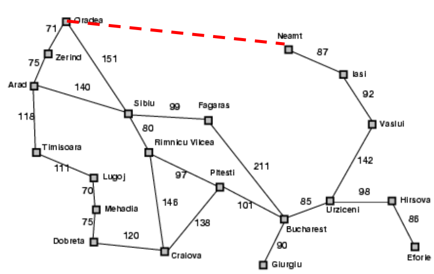
where $h^{*}(n)$ is the true cost to reach the goal state from *n*.

- **An admissible heuristic never overestimates the cost to reach the goal**

3

Admissible Heuristics

- Is the Straight Line Distance heuristic $h_{SLD}(n)$ *admissible?*
- Yes, it never overestimates the actual road distance

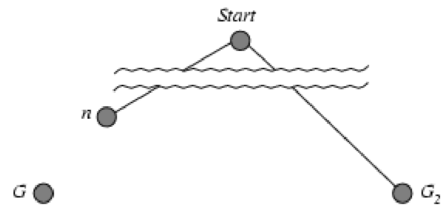


- Theorem: If *h(n)* is admissible, A^{*} using TREE-SEARCH is optimal.

4

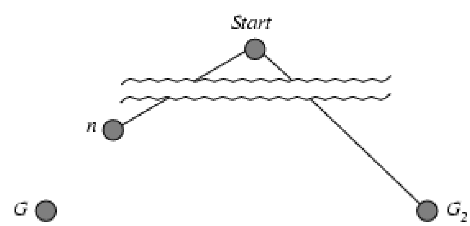
Optimality of A^* (proof)

Suppose some suboptimal goal G_2 has been generated and is in the frontier. Let n be an unexpanded node in the frontier such that *n is on a shortest path to an optimal goal G.*



$f(G) = g(G)$ since $h(G) = 0$
$f(G_2) = g(G_2)$ since $h(G_2) = 0$
$g(G) < g(G_2)$ since G_2 is suboptimal
$f(G) < f(G_2)$ from above

Optimality of A^* (cont.)

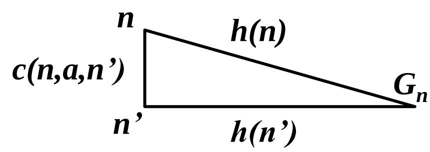


$f(G) < f(G_2)$ from prev slide
$h(n) \leq h^*(n)$ since h is admissible
$g(n) + h(n) \leq g(n) + h^*(n) = f(G)$
$f(n) \leq f(G) < f(G_2)$

Hence $f(n) < f(G_2) \Rightarrow A^*$ will select n and never G_2 for expansion.

Optimality of A* for Graph Search

- A heuristic *h(n)* is consistent if
  for every node *n* and every successor *n'* generated by an action *a*,
  $h(n) \leq c(n,a,n') + h(n')$
  (general triangle inequality)



- Theorem: If *h(n)* is consistent, A* using GRAPH-SEARCH is optimal.
  (see text for proof)

- Most admissible heuristics turn out to be consistent too
  E.g. SLD is a consistent heuristic for the route problem (prove it!)

7

Okay, enough theory… time to wake up!

8

Properties of A*

- Complete? Yes (unless there are infinitely many nodes with $f \leq f(G)$)
- Time? Exponential worst case but may be faster in many cases
- Space? Exponential: Keeps all generated nodes in memory (exponential # of nodes)
- Optimal? Yes

9

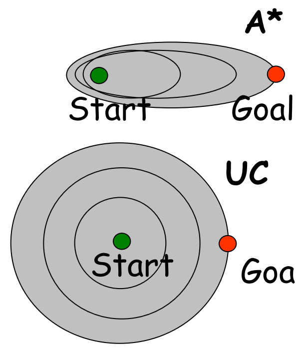
A* vs. Uniform Cost Search

- Both are optimal but differ in search strategy and time/space complexity
- A* uses $f(n) = g(n) + h(n)$ to find shortest path to a single goal
- Uniform cost search uses $f(n) = g(n)$ to find shortest path to a single goal

10

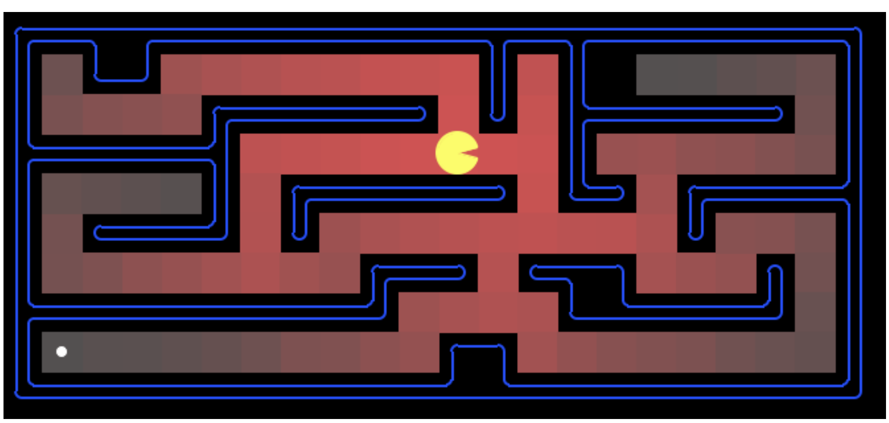
A* vs. Uniform Cost Search

- A* expands mainly toward the goal with the help of the heuristic function
- Uniform-cost expands uniformly in all directions
- A* can be more efficient (i.e., expands fewer nodes) if the heuristic is good



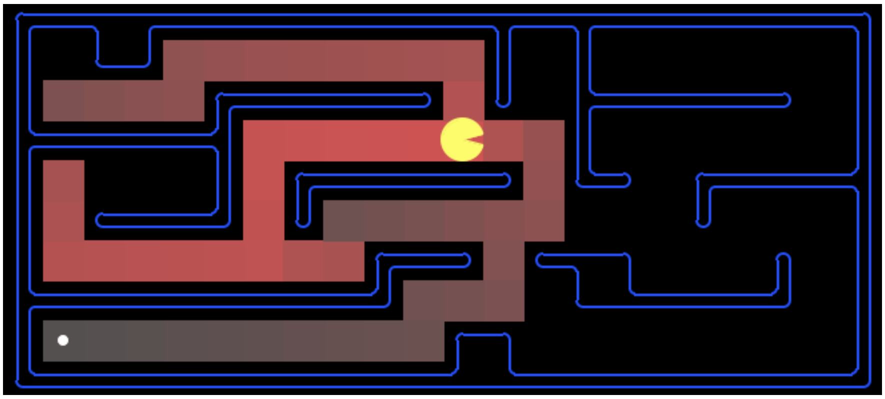
11

Uniform Cost Pac-Man



12

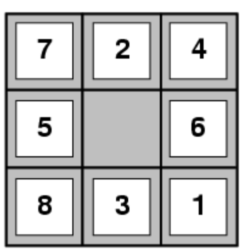
A* Pac-Man with Manhattan distance heuristic



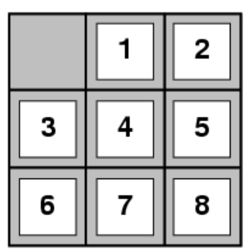
Let's explore heuristic functions

For the 8-puzzle (get to goal state with smallest # of moves), what are some heuristic functions?

- $h_1(n) = ?$
- $h_2(n) = ?$



Start State



Goal State

Example heuristic functions

Examples:
- $h_1(n)$ = number of misplaced tiles
- $h_2(n)$ = total Manhattan distance (no. of squares from desired location of each tile)

7	2	4
5		6
8	3	1

Start State

S

	1	2
3	4	5
6	7	8

Goal State

G

- $h_1(S)$ = ?
- $h_2(S)$ = ?

15

Example heuristics

Examples:
- $h_1(n)$ = number of misplaced tiles
- $h_2(n)$ = total Manhattan distance (no. of squares from desired location of each tile)

7	2	4
5		6
8	3	1

Start State

	1	2
3	4	5
6	7	8

Goal State

- $h_1(S)$ = ? 8
- $h_2(S)$ = ? 3+1+2+2+2+3+3+2 = 18

- *Are these admissible heuristics?*

16

Dominance

- If $h_2(n) \geq h_1(n)$ for all n (both admissible) then h_2 dominates h_1
- h_2 is better for search (why?)

  Getting closer to the actual cost to goal

- Does one dominate the other for:

  $h_1(n)$ = number of misplaced tiles

  $h_2(n)$ = total Manhattan distance

Dominance

- For 8-puzzle heuristics h_1 and h_2, typical search costs (average number of nodes expanded for solution depth d):

- *d=12* IDS = 3,644,035 nodes

  $A^*(h_1)$ = 227 nodes

  $A^*(h_2)$ = 73 nodes
- *d=24* IDS = too many nodes to fit in memory

  $A^*(h_1)$ = 39,135 nodes

  $A^*(h_2)$ = 1,641 nodes

For many problems, A* can still require too much memory

19

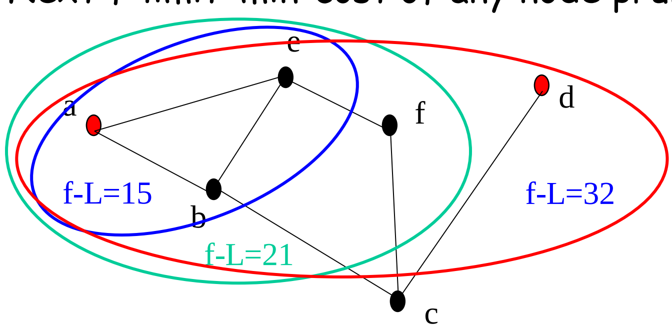
Iterative-Deepening A* (IDA*)

- Less memory required compared to A*
- Like iterative-deepening search, but…
- Depth bound modified to be an **f-limit**

  Start with limit = h(start)

  Prune any node if f(node) > f-limit

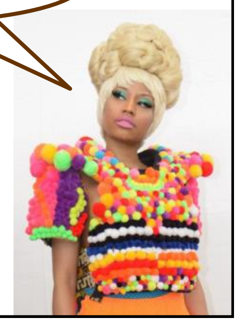
  Next f-limit=min-cost of any node pruned



20

That's cool yo but howdya derive 'em heuristic functions?

Just relax, bro!



Relaxed Problems

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

For route planning, what is a relaxed problem?

Relax requirement that car has to stay on road
→ Straight Line Distance becomes optimal cost

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

22

Heuristics for eight puzzle

7	2	3
5	1	6
8	4	

→

1	2	3
4	5	6
7	8	

start **goal**

- What can we relax?

23

Heuristics for eight puzzle

7	2	3
5	1	6
8	4	

→

1	2	3
4	5	6
7	8	

Original: Tile can move from location A to B if A is horizontally or vertically next to B *and* B is blank

Relaxed 1: Tile can move from any loc A to any loc B
Cost = h_1 = number of misplaced tiles

Relaxed 2: Tile can move from loc A to loc B if A is horizontally or vertically next to B
Cost = h_2 = total Manhattan distance

24

Need for Better Heuristics

Performance of h_2 (Manhattan Distance Heuristic)

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

Can we do better?

Adapted from Richard Korf presentation

Creating New Heuristics

- Given admissible heuristics h_1, h_2, …, h_m, none of them dominating any other, how to choose the best?
- Answer: No need to choose only one! Use:

  $h(n) = \max \{h_1(n), h_2(n), \ldots, h_n(n)\}$

  - h is admissible (prove it!)
  - h dominates each individual h_i (by construction)

Pattern Databases [Culberson & Schaeffer 1996]

- Idea: Use solution cost of a subproblem as heuristic.
- For 8-puzzle: pick any subset of tiles
  - E.g., 3 tiles
- Precompute a table

  Compute optimal cost of solving just these tiles
  - This is a lower bound on actual cost with all tiles

  Search backwards from goal and record cost of each new pattern encountered
  - State = position of just these tiles & blank
- Admissible heuristic h_{DB} for complete state = cost of corresponding sub-problem state in database

*	*	3
2	1	*
*	*	■

1	2	3
*	*	*
*	*	■

Combining Multiple Databases

- Repeat for another subset of tiles

  Precompute multiple tables
- How to combine table values?

  Use the *max* trick!

- E.g. Optimal solutions to Rubik's cube

  First found w/ IDA* using pattern DB heuristics

  Multiple DBs were used (diff subsets of cubies)

  Most problems solved optimally in 1 day

  Compare with ***574,000 years*** for IDS

Drawbacks of Standard Pattern DBs

- Since we can only take ***max***

  Diminishing returns on additional DBs

- Would like to be able to ***add*** values
  - But not exceed the actual solution cost (admissible)
  - How?

Adapted from Richard Korf presentation

Disjoint Pattern DBs

- Partition tiles into disjoint sets

  For each set, precompute table

  Don't count moves of tiles not in set
  - This makes sure costs are disjoint
  - Can be added without overestimating!
  - E.g. 8 tile DB has 519 million entries
  - And 7 tile DB has 58 million

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

- During search

  Look up costs for each set in DB

  Add values to get heuristic function value

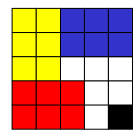
  Manhattan distance is a special case of this idea where each set is a single tile

Adapted from Richard Korf presentation

Performance of Disjoint PDBs

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

- 24 Puzzle: 12 millionx speedup *vs* Manhattan
  - IDA* can solve random instances in 2 days
  - Uses DBs for 4 disjoint sets as shown
  - Each DB has 128 million entries
  - Without PDBs: 65,000 years



Adapted from Richard Korf presentation

31

Next Time

- *Local search*
- *Gaming search and searching for Games*
- To do: Project #1, Read Sec. 4.1, Chap. 5

32