Introduction to Artificial Intelligence

Informed Search

Jianmin Li

Department of Computer Science and Technology
Tsinghua University

Spring, 2024

Outline

- Greedy best first search
- A* search
- Heuristics

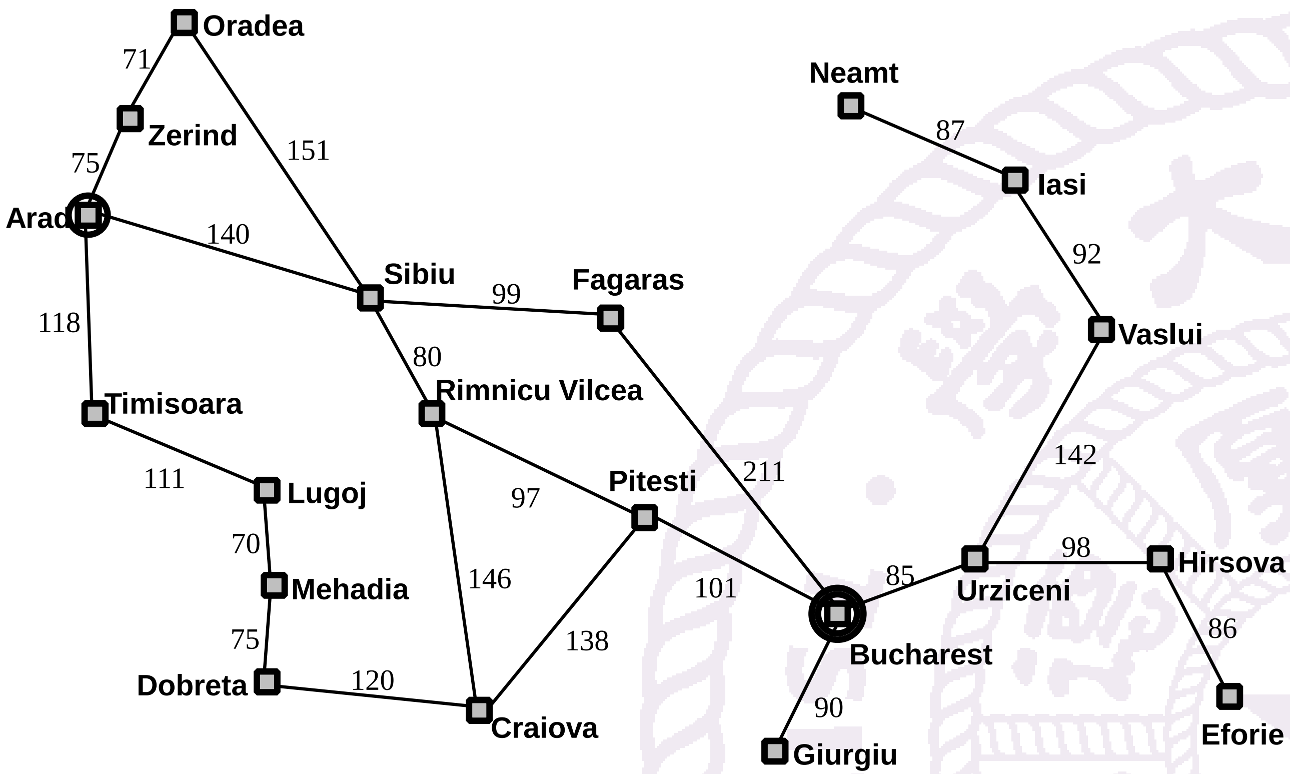
Best-first search

- Idea: use an evaluation function $f(n)$ of each node
  - estimate of "desirability"
  - Expand most desirable unexpanded node
- Implementation
  - fringe is a queue sorted in decreasing order of desirability
- Special cases
  - greedy best-first search
  - A* search

Greedy search

- Evaluation function $h(n)$ (heuristic)
  - estimate of cost from n to the closest goal
  - $h_{SLD}(n)$ = straight-line distance from n to Bucharest
- Greedy search expands the node that appears to be closest to goal

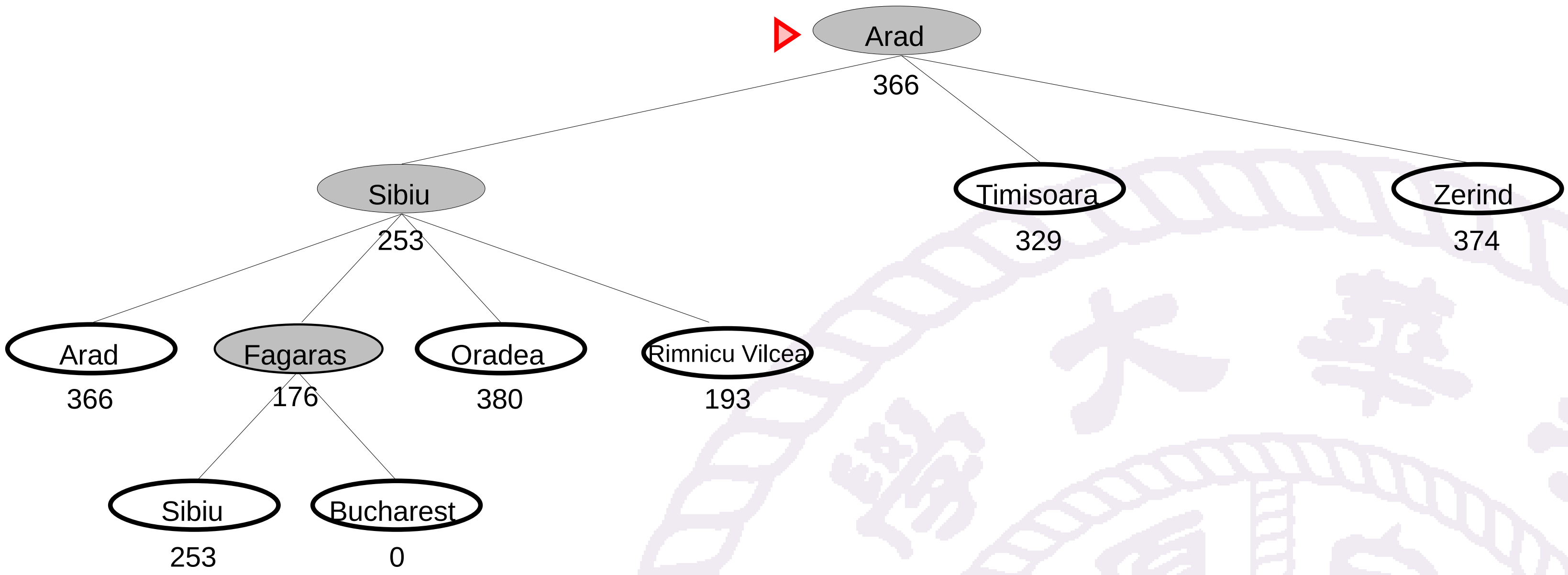
Greedy search: Romania



Straight−line distance to Bucharest

city	distance
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	98
Rimnicu Vilcea	193
Sibiu	253
Timisoara	329
Urziceni	80
Vaslui	199
Zerind	374

Greedy search: Romania

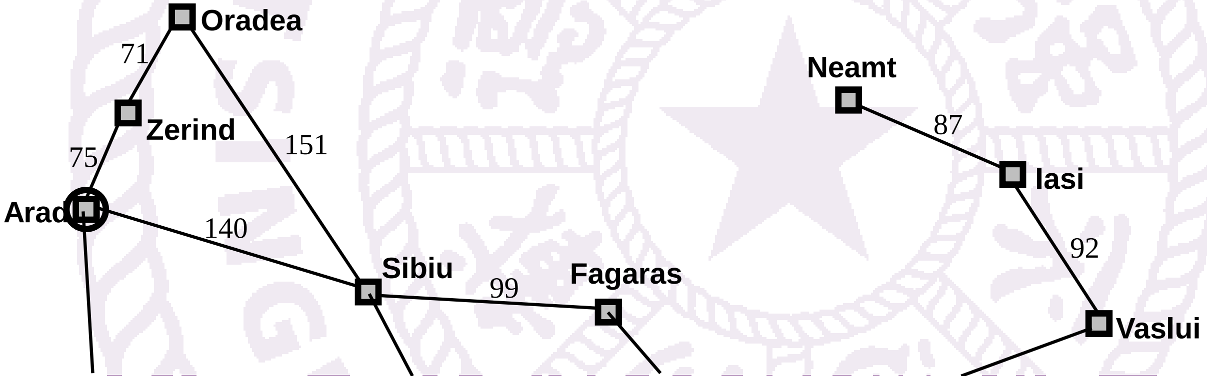


Straight−line distance to Bucharest

city	Arad	Bucharest	Craiova	Dobreta	Eforie	Fagaras	Giurgiu	Hirsova	Iasi	Lugoj
distance	366	0	160	242	161	176	77	151	226	244
city	Mehadia	Neamt	Oradea	Pitesti	Rimnicu Vilcea	Sibiu	Timisoara	Urziceni	Vaslui	Zerind
distance	241	234	380	98	193	253	329	80	199	374

Greedy search: Properties

- Complete
  - No. can get stuck in loops
    - start from Iasi, with Oradea as goal
    - Iasi -> Neamt -> Iasi -> Neamt …
  - Complete in finite space with repeated-state checking
- Time
  - $O(b^m)$, but a good heuristic can give dramatic improvement
- Space
  - $O(b^m)$, keeps all nodes in memory
- Optimal
  - No



Synergy (1)

- Breadth First / Uniform-cost search guaranteed to find optimal solution
- Greedy search often visits far fewer vertices, but may not provide optimal solution
- Can we get the best of both?

Synergy (2)

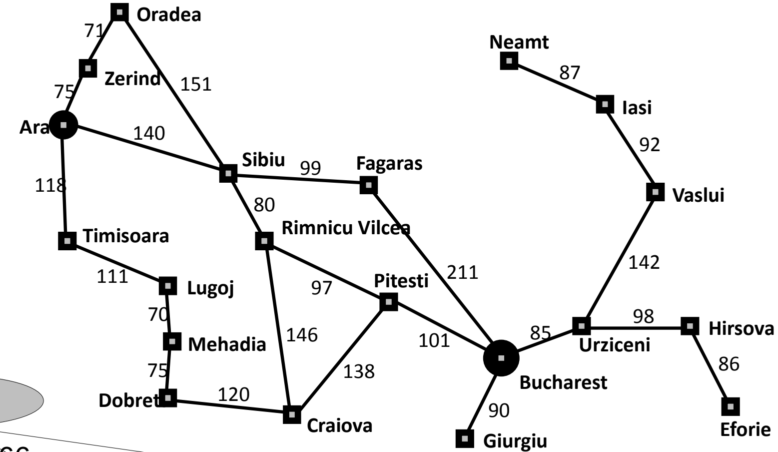
- Strategy of Breadth First / Uniform-cost search
  - Order nodes in priority queue to minimize actual distance from the start

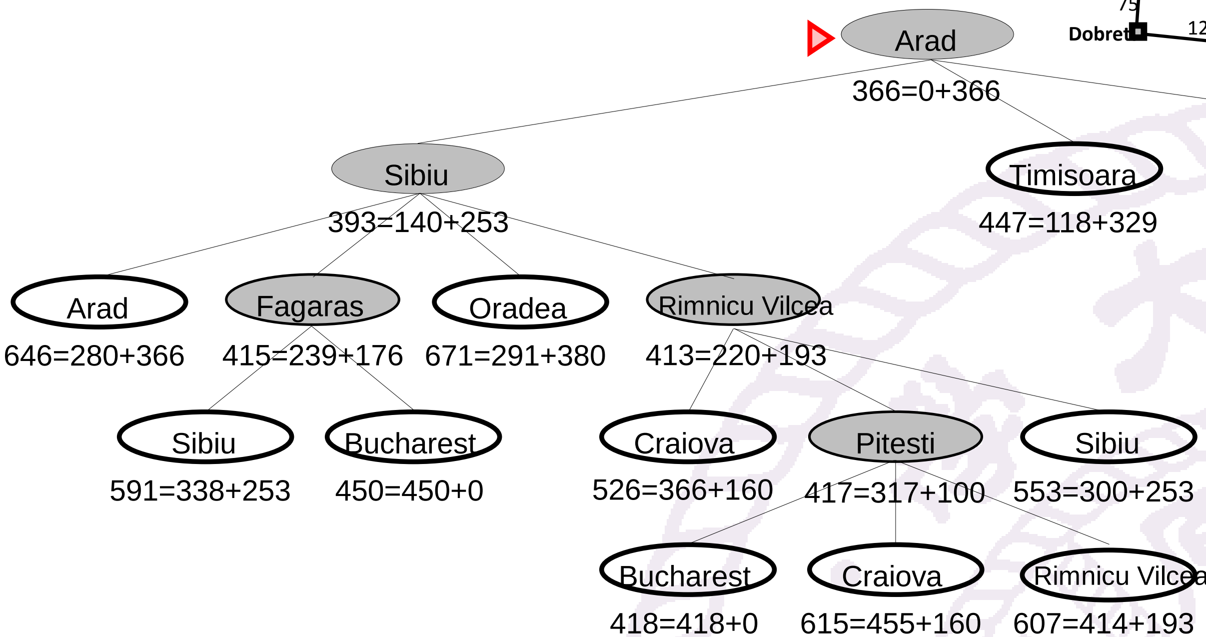
- Strategy of Greedy search
  - Order nodes in priority queue to minimize estimated distance to the goal

A* search

- Idea: avoid expanding paths that are already expensive
- Evaluation function $f(n) = g(n)+h(n)$
  - $g(n)$ = cost so far to reach n
  - $h(n)$ = estimated cost to goal from n
  - $f(n)$ = estimated total cost of path through n to goal

A* search: Romania



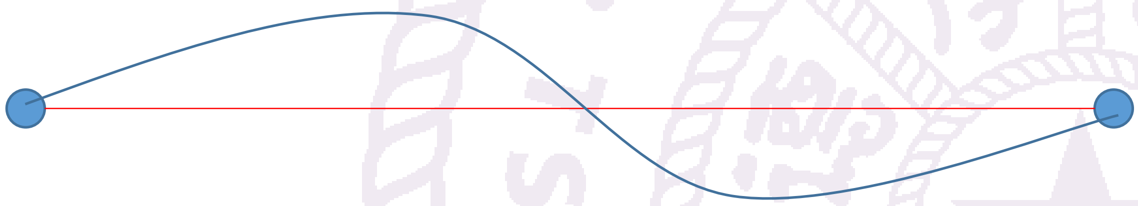


Straight−line distance to Bucharest

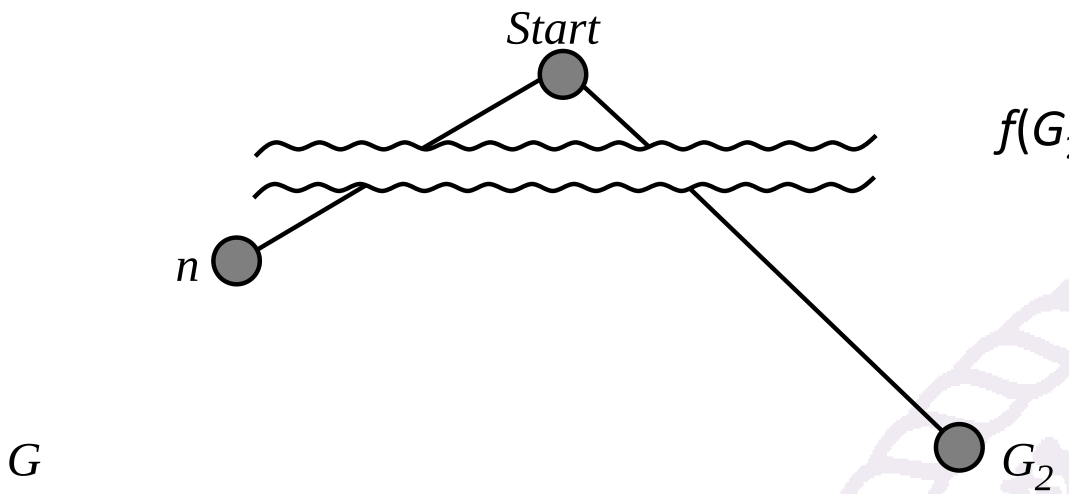
city	Arad	Bucharest	Craiova	Dobreta	Eforie	Fagaras	Giurgiu	Hirsova	Iasi	Lugoj
distance	366	0	160	242	161	176	77	151	226	244
city	Mehadia	Neamt	Oradea	Pitesti	Rimnicu Vilcea	Sibiu	Timisoara	Urziceni	Vaslui	Zerind
distance	241	234	380	98	193	253	329	80	199	374

Admissible heuristic

- Admissible heuristic
  - $h(n) <= h^*(n)$, where $h^*(n)$ is the true cost from n
  - Also require $h(n) >= 0$, so $h(G) = 0$ for any goal G
- Example
  - $h_{SLD}(n)$ never overestimates the actual road distance



Optimality of A* (1)



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

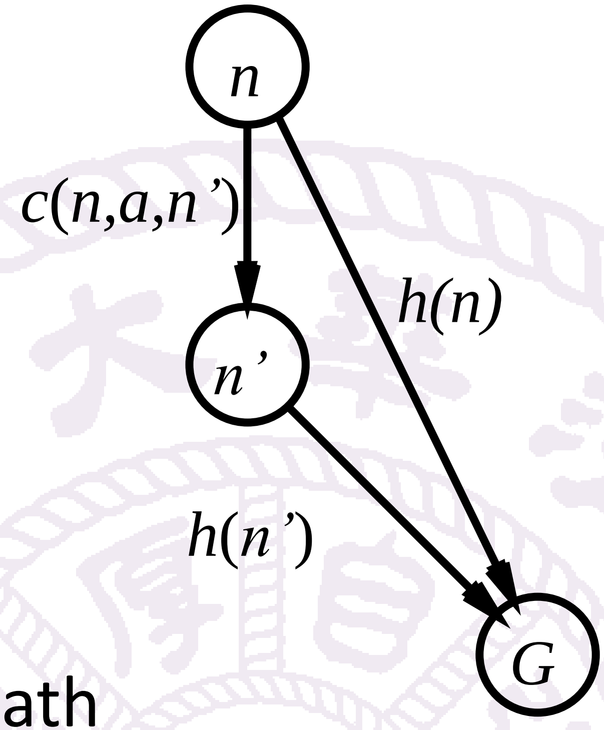
- Since $f(G_2) > f(n)$, A* will never select G_2 for expansion
- Theorem
  - If $h(n)$ is admissible, A* using TREE-SEARCH is optimal

Consistency

- A heuristic is consistent if
  - $h(n) \; <= c(n, a, n') + h(n')$
- if h is consistent, we have

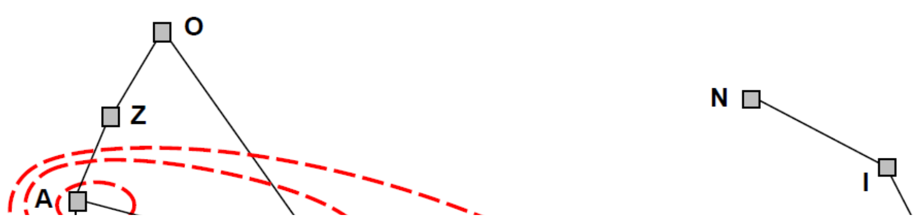
$$
\begin{aligned}
f(n') &= g(n') + h(n') \\
&= g(n) + c(n, a, n') + h(n') \\
&\geq g(n) + h(n) \\
&= f(n)
\end{aligned}
$$

- $f(n)$ is nondecreasing along any path
- Theorem
  - If $h(n)$ is consistent, A* using GRAPH-SEARCH is optimal



Optimality of A* (2)

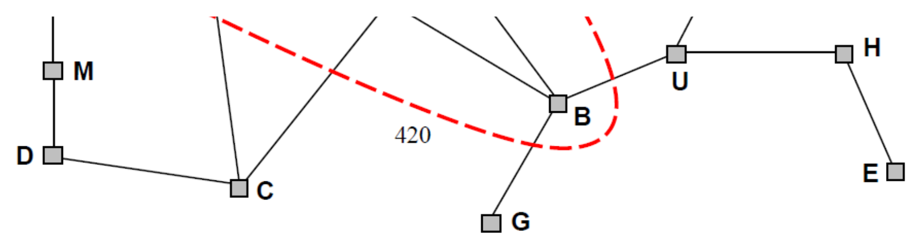
- A* expands nodes in order of increasing *f* value
  - Gradually adds "*f*-contours" of
  - Contour *i* has all nodes with $f = f_i$, where $f_i < f_{i+1}$



A* expands all nodes with $f(n) < C$

A* expands some nodes with $f(n) = C$

A* expands no nodes with $f(n) > C$

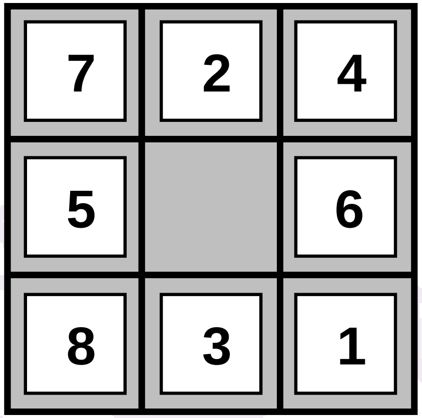


A* search: Properties

- Complete
  - Yes, unless there are infinitely many nodes with f <= $f(G)$
- Time
  - is optimally efficient for any given heuristic function
    - no other optimal algorithm is guaranteed to expand fewer nodes than A*
  - Exponential in [relative error in h x length of soln.]
- Space
  - keeps all nodes in memory
- Optimal
  - Yes. can not expand f_{i+1} until f_i is finished

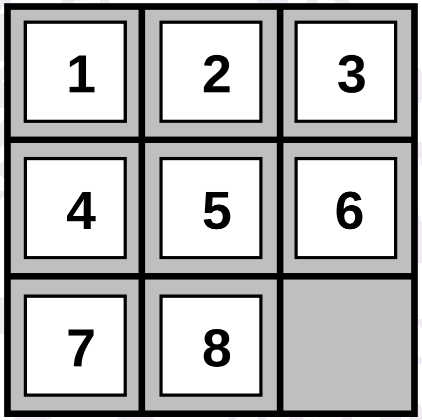
Admissible heuristics: 8-puzzle

- $h_1(n)$ = number of misplaced tiles



Start State

- $h_2(n)$ = total Manhattan distance



Goal State

Question 1

- $h_1(n)$ = number of misplaced tiles

A. 5
B. 6
C. 7
D. 8

7	2	4
5		6
8	3	1

Start State

1	2	3
4	5	6
7	8	

Goal State

Question 2

- $h_2(n)$ = total Manhattan distance

A. 11
B. 12
C. 14
D. 15

7	2	4
5		6
8	3	1

Start State

1	2	3
4	5	6
7	8	

Goal State

The effect of heuristic accuracy on performance

- Effective Branching Factor b^*

$$N+1 = 1+b+(b^*)^2+(b^*)^3+ \ldots +(b^*)^d$$

d	Search Cost (nodes generated)			Effective Branching Factor		
	IDS	A*(h_1)	A*(h_2)	IDS	A*(h_1)	A*(h_2)
2	10	6	6	2.45	1.79	1.79
4	112	13	12	2.87	1.48	1.45
8	6384	39	25	2.80	1.33	1.24
12	3644035	227	73	2.78	1.42	1.24
16		1301	211		1.45	1.25
20		7276	676		1.47	1.27
24		39135	1641		1.48	1.26

Dominance

- If $h_2(n)$ >= $h_1(n)$ for all n (both admissible) then h_2 dominates h_1
- h_2 is better for search
  - d = 12, IDS = 3,644,035 nodes, A*(h_1) = 227 nodes, A*(h_2) = 73 nodes
  - d = 24, IDS 54,000,000,000 nodes, A*(h_1) = 39,135 nodes, A*(h_2) = 1,641 nodes

Inventing Heuristic Functions

- Composite heuristics
  - $h(n) = \max(h_1(n), \ldots, h_m(n))$
  - Admissible?
  - Consistent?
- Learn the coefficients for features of a state
  - $h(n) = c_1 x_1(n) + \ldots + c_k x_k(n)$
  - Admissible?
  - Consistent?
- Good heuristics should be efficiently computable

Relaxed Problems

- A problem with fewer restrictions (on the actions) is called a relaxed problem
  - original problem: A if B and C
  - relaxed problem 1: A if B
  - relaxed problem 2: A if C
  - relaxed problem 3: A
- The cost of an optimal solution to a relaxed problem is an admissible heuristic for the original problem
  - the optimal solution cost of a relaxed problem is no greater than the optimal solution cost of the real problem

Relax Problems: 8-puzzle (1)

- 8-puzzle: move a tile from cell A to cell B
- Original conditions:
  - cond1: there is a tile on A
  - cond2: B is empty
  - cond3: A and B are adjacent (horizontally or vertically)
- Relaxed problems:
  - Remove cond2: Move from A to B, if A is adjacent to B.
    - => Manhattan distance

Relax Problems: 8-puzzle (2)

- 8-puzzle: move a tile from cell A to cell B
- Original conditions:
  - cond1: there is a tile on A
  - cond2: B is empty
  - cond3: A and B are adjacent (horizontally or vertically)
- Relaxed problems:
  - Remove cond3: Move from A to B, if B is empty.
    - => Gaschnig’s heuristic (1979)

Relax Problems: 8-puzzle (3)

- 8-puzzle: move a tile from cell A to cell B
- Original conditions:
  - cond1: there is a tile on A
  - cond2: B is empty
  - cond3: A and B are adjacent (horizontally or vertically)
- Relaxed problems:
  - Remove cond2 and cond3: Move from A to B. i.e. a tile can be moved to anywhere
    - => Misplaced tiles

Summary

- Heuristic functions *h*(*n*) estimate costs of shortest paths from *n*
- Good heuristics can dramatically reduce search cost
- Greedy best-first search expands lowest *h*
  - incomplete and not always optimal
- A* search expands lowest *g*+*h*
  - complete and optimal
  - optimally efficient
- Admissible heuristics can be derived from exact solution of relaxed problems

谢谢！