22c:145 Artificial Intelligence

Fall 2005

Informed Search and Exploration II

Cesare Tinelli

The University of Iowa

Readings

- Chap. 4 of [Russell and Norvig, 2003]

Admissible Heuristics

A* search is optimal when using an admissible heuristic function h.

How do we devise good heuristic functions for a given problem?

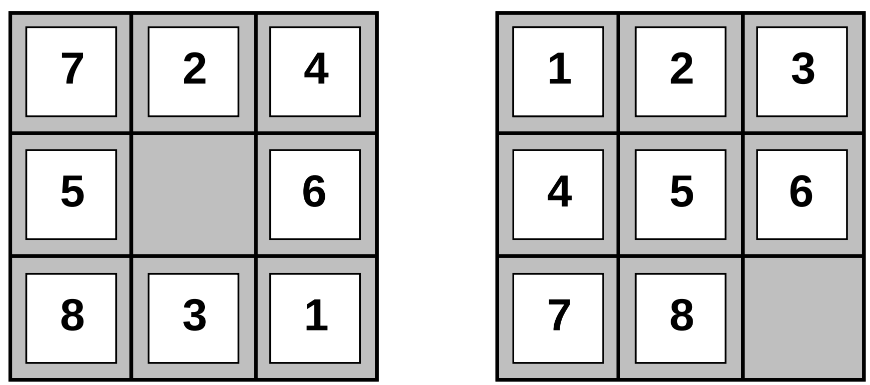
Typically, that depends on the problem domain.

However, there are some general techniques that work reasonably well across several domains.

Examples of Admissible Heuristics

Consider the 8-puzzle problem:

- $h_1(n)$ = number of tiles in the wrong position at state n
- $h_2(n)$ = sum of the Manhattan distances of each tile from its goal position.



Start State **Goal State**

- $h_1(Start)$ =
- $h_2(Start)$ =

Examples of Admissible Heuristics

Consider the 8-puzzle problem:

- $h_1(n)$ = number of tiles in the wrong position at state n
- $h_2(n)$ = sum of the Manhattan distances of each tile from its goal position.

7	2	4
5		6
8	3	1

Start State

1	2	3
4	5	6
7	8	

Goal State

- $h_1(Start)$ = 7
- $h_2(Start)$ =

Examples of Admissible Heuristics

Consider the 8-puzzle problem:

- $h_1(n)$ = number of tiles in the wrong position at state n
- $h_2(n)$ = sum of the Manhattan distances of each tile from its goal position.

7	2	4
5		6
8	3	1

Start State

1	2	3
4	5	6
7	8	

Goal State

- $h_1(Start)$ = 7
- $h_2(Start)$ = 4+0+3+3+1+0+2+1 = 14

Dominance

Definition A heuristic function h_2 dominates a heuristic function h_1 for the same problem if $h_2(n) \geq h_1(n)$ for all nodes n.

- For the 8-puzzle, h_2 = *total Manhattan distance* dominates h_1 = *number of misplaced tiles*.
- With A* search, a heuristic function h_2 is always better for search than a heuristic function h_1, if h_2 is admissible and dominates h_1.
- The reason is that A* with h_1 is guaranteed to expand at least all as many nodes as A* with h_2.
- What if neither of h_1, h_2 dominates the other? If both h_1, h_2 are admissible, use $h(n) = max(h_1(n), h_2(n))$.

Effectiveness of Heuristic Functions

Definition Let

- h be a heuristic function h for A*,
- N the total number of nodes expanded by one A* search with h,
- d the depth of the found solution.

The effective branching Factor (EBF) of h is the value b^* that solves the equation

$$x^d + x^{d-1} + \cdots + x^2 + x + 1 = N$$

(the branching factor of a uniform tree with N nodes and depth d).

A heuristics h for A* is effective in practice if its average EBF is close to 1.

Note: If h dominates some h_1, its EFB is never greater than h_1's.

Dominance and EFB: The 8-puzzle

	Search Cost			Effective Branching Factor		
d	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
6	680	20	18	2.73	1.34	1.30
8	6384	39	25	2.80	1.33	1.24
10	47127	93	39	2.79	1.38	1.22
12	364404	227	73	2.78	1.42	1.24
14	3473941	539	113	2.83	1.44	1.23
16	–	1301	211	–	1.45	1.25
18	–	3056	363	–	1.46	1.26
20	–	7276	676	–	1.47	1.27
22	–	18094	1219	–	1.48	1.28
24	–	39135	1641	–	1.48	1.26

- Average values over 1200 random instances of the problem
- Search cost = no. of expanded nodes
- IDS = iterative deepening search
- $A^*(h_1)$ = A* with h = number of misplaces tiles
- $A^*(h_2)$ = A* with h = total Manhattan distance

Devising Heuristic Functions

A relaxed problem is a search problem in which some restrictions on the applicability of the next-state operators have been lifted.

Example

- original-8-puzzle: “A tile can move from position A to position B if A is adjacent to B and B is empty.”
- relaxed-8-puzzle-1: “A tile can move from A to B if A is adjacent to B.”
- relaxed-8-puzzle-2: “A tile can move from A to B if B is empty.”
- relaxed-8-puzzle-3: “A tile can move from A to B.”

The exact solution cost of a relaxed problem is often a good (admissible) heuristics for the original problem.

Key point: the optimal solution cost of the relaxed problem is no greater than the optimal solution cost of the original problem.

Relaxed Problems: Another Example

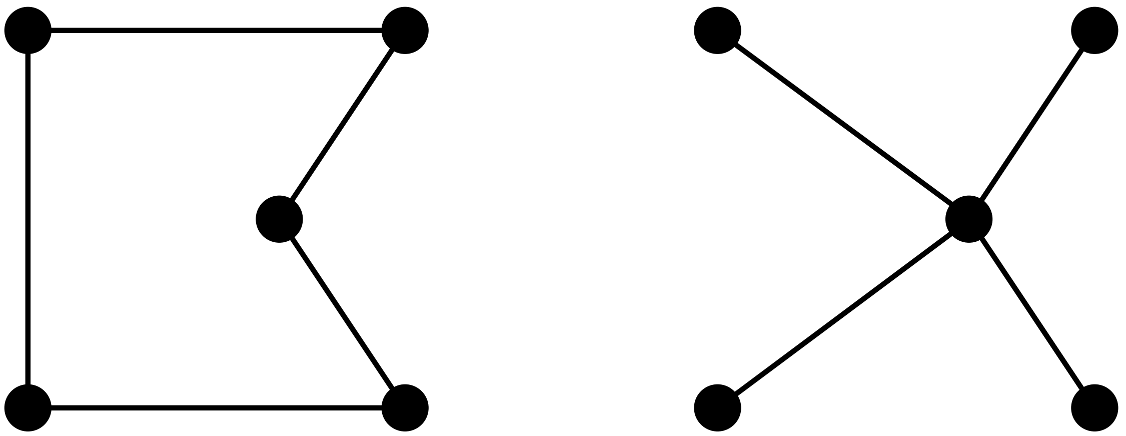
Original problem Traveling salesperson problem: Find the shortest tour visiting n cities exactly once.

Complexity: NP-complete.

Relaxed problem Minimum spanning tree: Find a tree with the smallest cost that connects the n cities.

Complexity: $O(n^2)$

Cost of tree is a lower bound on the shortest (open) tour.



Devising Heuristic Functions Automatically

- Relaxation of formally described problems.
- Pattern databases.
- Learning.