Comp3620/Comp6320 Artificial Intelligence
Tutorial 2: Search Heuristics, Game Tree Search

What we want is a 'whole part' consistent heuristics, partial won't count.

admissibility: heuristics never overestimate the real cost.

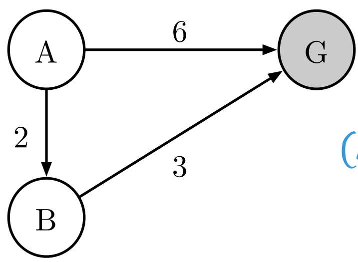
March 20-23, 2018

Exercise 1 (heuristic function properties)

Consider the search problem shown on the left. It has only three states, and three directed edges. A is the start node and G is the goal node. To the right, four different heuristic functions are defined, numbered I through IV.

$h(A) \leq 2+3$

~~ad ✓~~



$\overset{2}{(A) \leq c(A,B)} + h(B)$

	h(A)	h(B)	h(G)
I	4	1	0
II	6	3	0
III	4	3	0
IV	5	2	0

× cons ✓ad
× cons × ad
✓ cons ✓ad
× cons ✓ad

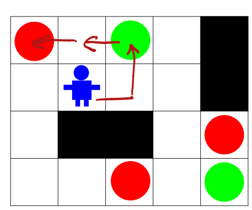
For each heuristic, state whether it is admissible and whether it is consistent for the above search problem. Compare the informativeness of heuristics III and IV (i.e., state whether one of the heuristics dominate the other) and the informativeness of heuristics I and IV.

dominance: equal or better in everywhere. I vs IV, IV is better. The best heuristics ⇒ perfect estimation.

Exercise 2 (combining heuristics and heuristics for multiple goals)

There are green and red objects on a grid. An agent must collect *exactly one object of each color* to reach the goal. The actions are moving south, north, east or west, and are only applicable when they don't result in colliding with an obstacle (black) or exiting the grid. The agent collects an object when it first reaches the cell at which this object is. The state of the problem is represented as follows. Each state is a triple (a, G, R) where a is the location of the agent on the grid, G is the set locations of yet uncollected green objects, and R is the set of locations of yet uncollected red objects. Given two locations l_1 and l_2 on the grid, $dist(l_1, l_2)$ returns the Manhattan distance between l_1 and l_2.

(4)



manhattan generally admissible in this question b/c ~~blocking~~

Which of the following heuristics are admissible at any *non-goal* state $s = (a, G, R)$ for this problem:

1. The sum of the Manhattan distances to the remaining objects?
$$\sum_{o \in G \cup R} dist(a, o)$$
× admissible (overestimates!)

$= 16 > 4$

2. The number of remaining objects? ✗
   $|G| + |R|$
3. The smallest Manhattan distance to any remaining objects? ✓
   $\min_{o \in G \cup R} dist(a, o)$
4. The maximum Manhattan distance between any two remaining objects? ✗
   $\max_{o_1 \in G \cup R, o_2 \in G \cup R} dist(o_1, o_2)$?
5. The minimum Manhattan distance between any two remaining objects of opposite colors? ✗
   $\min_{o_1 \in G, o_2 \in R} dist(o_1, o_2)$?

going to the closest does not guarantee finishing the game so no overestimation

some state s.t. heuristics overestimates: (e.g. ♀ ⇒ right-most red, then just 1 more step to finish, but heuristics reports 2.)

Some of the above heuristics, can be seen as the (often inadmissible) combination of several admissible heuristics for individual goals. This leads to the question of which combination of admissible heuristics are generally admissible. Let $h(s)$, $i(s)$ and $j(s)$ be three admissible heuristics; indicate which combinations below are also guaranteed to be admissible:

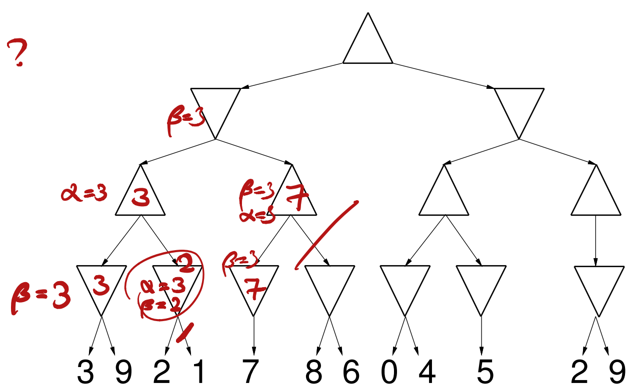
1. $\max(h(s), i(s), j(s))$ ✓
2. $\min(h(s), i(s), j(s))$ ✓
3. $\max(h(s), i(s) + j(s))$ ✗
4. $h(s) + i(s) + j(s)$ ✗
5. $h(s) * i(s) * j(s)$ ✗
6. $h(s)/3 + i(s)/3 + j(s)/3$ ✓

never going bigger than the biggest before

Exercise 3 (game tree search)

Apply Minimax without and with alpha-beta pruning to the following game tree. The root at the top of the tree is a max-node.

minimax is easy
what about α-β pruning?



In general, which of the following assertions are correct about alpha-beta pruning?

✓ 1. It can reduce computation time by pruning portions of the game tree

✗ 2. It is generally faster than minimax but loses the guarantee of optimality

✓ 3. It always returns the same value as minimax for all nodes on the leftmost edge of the tree, assuming successor nodes are expanded from left to right

✗ 4. It always returns the same value as minimax for all nodes in the tree