CS188 Spring 2012 Section 2: A* Search

1 Pancake Heuristics

Here, we consider the *pancake problem*. A server is given a stack of n pancakes. Each pancake is a different size. The server can flip the top k pancakes, reversing their order. **The cost of flipping k pancakes is k.** The server's goal is to order the pancakes from smallest (top) to largest (bottom), with minimal cost. More formally, the search states are all permutations σ of $(1, 2, 3, \ldots, n)$, and the goal is $(1, 2, 3, \ldots, n)$. The successor function gives the outcome of flips, for example:

Successors$((3, 4, 1, 2, 5))$ =

action	cost	successor state
flip 2	2	(4,3,1,2,5)
flip 3	3	(1,4,3,2,5)
flip 4	4	(2,1,4,3,5)
flip 5	5	(5,2,1,4,3)

Here are three heuristics for the pancake problem:

1. H_1, The largest pancake that is out of place: largest i such that $i \neq \sigma_i$
2. H_2, The number of pancakes out of position: count of all i such that $i \neq \sigma_i$
3. H_3, One less than the size of the pancake at the top of the stack: $\sigma_1 - 1$

a) Circle all of the following heuristics that are *admissible*:

i.) H_1 ii.) H_2 iii.) H_3 iv.) $H_1 + H_2$ v.) $H_2 + H_3$ vi.) $\max(H_1, H_2, H_3)$

b) A heuristic H_A dominates a heuristic H_B if $H_A(n) \geq H_B(n)$ for every state. Circle all of the following statements that are true:

i. H_1 dominates H_2

ii. H_1 dominates H_3

iii. H_2 dominates H_1

c) Circle all of the following heuristics that are *consistent*:

i.) H_1 ii.) H_2 iii.) H_3 .

2 A*-CSCL

We saw that for A^* graph search to be guaranteed to be optimal the heuristic needs to be consistent. In this question we explore a new search procedure, A^*-graph-search-with-Cost-Sensitive-Closed-List (A^*-CSCL). In A^*-CSCL we replace the usual closed list with a cost-sensitive closed list, which stores the f-cost of an expanded node along with the last state of the partial plan associated with that node.

Whenever A^*-CSCL considers expanding a node, it first verifies whether the node's last state is in the cost-sensitive closed list and only expands it if either (a) the node's last state is not in the cost-sensitive closed list, or (b) the node's last state is in the cost-sensitive closed list with a higher f-cost than the f-cost for the node currently considered for expansion.

If A^*-CSCL expands a node per (a), its last state and f-cost get added to the cost-sensitive closed list; if it expands a node per (b), the cost associated with the node's state gets replaced by the current node's f-cost. Select all of the following statements that are true about A^*-CSCL:

i. If h is admissible, then A^*-CSCL finds an optimal solution.

ii. If h is consistent, then A^*-CSCL finds an optimal solution.

iii. If h is admissible, then A^*-CSCL will expand at most as many nodes as A^* tree search.

iv. If h is consistent, then A^*-CSCL will expand at most as many nodes as A^* tree search.

v. If h is admissible, then A^*-CSCL will expand at most as many nodes as A^* graph search.

vi. If h is consistent, then A^*-CSCL will expand at most as many nodes as A^* graph search.

3 A* with weighted heuristics

Given an admissible heuristic, we have seen that A^* tree search will expand all nodes with f-cost less than or equal to the optimal cost c^*. That is, A^* will expand all nodes for which $g + h < c^*$ before expanding the goal (and potentially some nodes with $g + h = c^*$, depending on tie-breaking process when popping from the fringe). For large state spaces, this could take too long. We may be interested in finding a goal more quickly, while accepting some degree of suboptimality. This question explores the resulting tradeoff between optimality and number of nodes expanded.

Consider scaling the consistent heuristic h by a parameter $\alpha > 1$, and using the resulting $f = g + \alpha h$ as the queueing function for A^* tree search.

(a) Is the weighted heuristic αh guaranteed to be admissible?

(b) Is the weighted heuristic αh guaranteed to be consistent, assuming the original h was consistent?

(c) Define c_w to be the cost of the solution found from running A^* tree search with the weighted heuristic. Give an expression (similar to the one above) bounding the f-costs of nodes guaranteed to be expanded before the goal.

(d) Give a bound relating the cost c_w of the solution returned using the weighted heuristic to the optimal cost c^*.