MCS 380/590 Intro To Artificial Intelligence
Dr. William C. Regli
Drexel University Fall 2000
Homework Assignment Solutions #3
Artificial Intelligence (Russell / Norvig)

4.1 Greedy seach algorithm with h(n)=-g(n) means that the best nodes (the ones with the lowest score) will be those with the longest path (the higher g(n) and thus lower -g(n)). Preferring longest paths means depth-first search. If we set h(n)=g(n), we would get breadth-first search.

4.2c Path-planning: a reasonable heuristic is the straight-line distance to the goal point. This is admissile because the straight-line distance is always less than or equal to the best path. There are local maxima in the search space because a set of moves can put you really close to the goal, but maybe you will have to have to backtrack or take a longer path to reach the goal state because there is a polygon in the way. Or yet, consider a space where the start point is at the middle of a concave crescent moon shape, and the goal is just on the other side of the crescent. The heuristic will lead the search to a local maxima at the middle of the crescent, when actually the best solution is to go around the outside.

4.3 Goal: every square (except the black ones) contains a letter, and every word slot (string of consecutive of vertical squares) contains a word from the dictionary.

Initial State: a grid of all empty squares, and a dictionary with all the words.

Operators: There are many choices here. You can try one word at a time, or one letter at a time. If you do word-at-time, you can choose any word from the dictionary and place it in any word slot of the same lenght, as long as there are no contradictory letters already in place there. The resulting state has one more word slot filled in.

Path cost: One per move.

Search Strategy: A constraint-satisfaction problem. Variables are the word slots, the domain of each variable is the list of words of appropriate length in the dictionaty, and the constraint is that if two words intersect, they must have the same letter in the intersecting square.

Heuristic: We could use the most-constrained-variable heuristic, but it is expensive if the dictionary is large. We can get an estimate by choosing K words at random and seeing how many of them fit in a slot. We can do the same with the least-constraining-value: of the first k candidate words to fit a slot, choose the one that imposes the least constrains on the intersecting slots.

4.4 It is enough to do a best-first search because with the comparison method we are able to sort the nodes, and choose the node that is at the top of the sort. The property we give up is run-time efficiency: a priority queue is not as efficient when we only have a comparison operator, not a numeric ranking.

4.5 Going between Rimnicu Vilcea and Logoj is one example. The shortest path is the Southern one, though Mehadia, Dobreta and Craicova. But a greedy search using the straight-line heuristic starting in Rimnicu Vilcea will start the wrong way, heading to Sibiu. Starting at Lugoj, the heuristic will correctly lead us to Mehadia, but then a greedy search will return to Lugoj, and oscillate forever between these two cities

4.6 One solution:

A heuristic where h(n) = sum of the numbers on the squares that are not in their correct locations. In an example, we could be only one move from the goal state, and h(n) would be 8, obviously an overestimation.

4.10 Bidirectional A* can be a good idea: the forward search can use a heuristic that estimates distance to the goal, and the backward can use a heuristic that estimates distance to the start. Some problems may arise - the two searches may never meet each other, And if they meet, there is no guarantee the first meeting point is an optimal solution.