Artificial Intelligence (CSCI 446)

Homework #1: 100 points

Due September 8, 2014

All work must be completed in accordance with the *Group Work* policy, and use of the web is prohibited (unless otherwise indicated). Any external sources must be cited. All papers are due at the start of class on the due date. No hand-written papers will be accepted.

1. (20 points) R&N 2.1 (3/e): Suppose that the performance measure is concerned with just the first T time steps of the environment and ignores everything thereafter. Show that a rational agent's action may depend not just on the state of the environment but also on the time step it has reached. (To clarify this question, think of a situation where behavior depends upon extending the state description to include the time step. By "ignore," the authors mean that any rewards after time step T would not accrue to the agent.)

2. R&N 3.26 (3/e): [Reworded or amended for improved clarity] Consider an unbounded version of the regular 2D grid shown in Figure 3.9 of the text (a similar figure is shown in Figure x). The state is at the origin $(0, 0)$, and the goal state is at (x, y).

   (a) (2.5 points) What is the branching factor b in this state space?

   (b) (2.5 points) How many distinct states are there at depth k (for $k > 0$)?

   (c) (2.5 points) What is the maximum number of nodes expanded by breadth first tree search where tree search does not include a test for visiting a repeated state? (*Hint:* We are looking for a function in x and y.)

   (d) (2.5 points) What is the maximum number of nodes expanded by breadth first graph search where graph search includes a test for visiting a repeated state?

   (e) (2.5 points) Is $h = |u - x| + |v - y|$ and admissible heuristic for a state at (u, v)? Explain.

   (f) (2.5 points) How many nodes are expanded by A^* graph search using h?

   (g) (2.5 points) Is h admissible if some links are removed?

   (h) (2.5 points) Is h admissible if some links are added between non-adjacent states?

3. Consider the graph in Figure 1 representing the state space and operators for a navigation problem. Here, S is the start state and G is the goal state. The path cost is shown by the number on the links, and the heuristic evaluation is shown by the number in the box. When placing expanded child nodes on a queue, assume that the child nodes are placed in alphabetical order (i.e., if node S is expanded, the queue will be $[AB]$). Finally, assume that we never generate child nodes that appear as ancestors of the current node in the search tree.

   (a) (5 points) What is the order that breadth first search will expand the nodes?

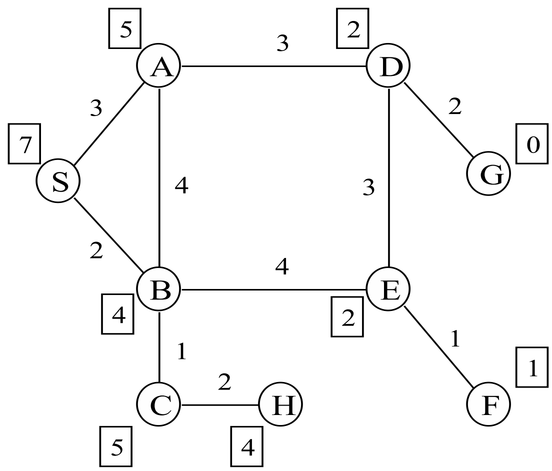
   (b) (5 points) What is the order that depth-first search will expand the nodes?

   (c) (5 points) What is the order that A^* search will expand the nodes?

4. Consider a search space where we might use A^*. We have a cost function g that evaluates the distance from the start state to a node in the search tree and an admissible heuristic h that evaluates the distance between any intermediate state s and the closest goal. Assume that the resulting function $f = g + h$ is strictly monotonic, i.e., if n' is a successor of n, then $f(n') > f(n)$.

   An alternative algorithm to A^* (Algorithm 1) is called "depth-first branch-and-bound" (BBS). This algorithm is also used to find the shortest path from the initial state to a goal state. Unlike A^*, however, BBS is a depth-first search. Thus it is only used when it is reasonable to assume that the search tree is finite, i.e., that any path terminates after a bounded number of steps. Essentially, the algorithm maintains the value of the best goal found so far. It does a depth-first search, backtracking whenever it encounters a node whose value is larger than its current upper bound.

Figure 1: Navigation Graph



Algorithm 1 Depth-First Branch and Bound

```
Initialize with upperBound = ∞
BBS(node)
{
   // pruning
   if (f(node) ≥ upperBound)
      return;
   // goal test
   if (isGoal(node))
      upperBound = min(upperBound, g(node));
   else
      for each s in Succ(node)
         BBS(s);
   return
}
```

(a) (5 points) Show that BBS is complete.

(b) (10 points) Show that BBS is optimal.

(c) (10 points) One can show that A^* never expands more nodes than BBS. However, BBS is widely used. State two reasons why BBS might be preferred to A^*.

5. A *SAT* or *Boolean satisfiability problem* is a decision problem regarding the existence of a complete assignment of truth values (*true* or *false*) to the variables in a clause that make the entire clause true. The clause in question consists of variables, parentheses, and Boolean connectives. Such a clause is *satisfiable* if there are one or more complete assignments that make the clause true and *unsatisfiable* if no such assignment exists.

   We would like to solve *SAT* using a greedy hill climbing algorithm. Each state corresponds to a complete assignment. The successor operator $Succ(s)$ generates all neighboring states of s, which we will define as all total assignments that differ by one. So, for example, given the state with total assignment $\{A \leftarrow true, B \leftarrow false\}$, the neighboring states would be $\{A \leftarrow false, B \leftarrow false\}$ and $\{A \leftarrow true, B \leftarrow true\}$. We can set the evaluation of a state to be the number of clauses that are satisfied, given the assignment of the state. This algorithm is usually called *GSAT*. Assume that ties in the evaluation function are broken randomly.

   (a) (5 points) If you have n variables, how many neighboring states does the $Succ(s)$ function produce?

   (b) (5 points) What is the total size of the search space, i.e., how many possible states are there total? Assume again that there are n variables.

   (c) (5 points) Consider the following set of clauses:

   $$\{\neg A \vee B \vee C\}, \{A \vee \neg B \vee C\}, \{A \vee B \vee \neg C\}, \{A \vee B \vee C\}.$$

   Come up with a non-goal state (i.e., a non-satisfying assignment) that is on a plateau in our hill climbing space.

   (d) (5 points) Consider the following *SAT* problem:

   $$(X \vee Y) \wedge (\neg X \vee Z) \wedge (\neg Y \vee Z) \wedge (\neg X \vee \neg Z) \wedge (X \vee U) \wedge (X \vee V)$$

   From the starting point below, present a sequence of moves that *GSAT* can execute that will result in the global maximum. For each move, give the resulting assignment and the value of the *GSAT* evaluation function for that assignment.

   $$\{X = true, Y = false, Z = true, U = false, V = false\}; val = 5.$$