CSE 3402 Miterm Test Sample Questions

February 20, 2007

1 Short Answer

1. Is A*'s search behavior necessarily "exponentially explosive"?. That is, does its search time always grow at least exponentially with the length of the optimal solution.

2. It would seem that iterative deepening search should have a higher asymptotic time complexity than breadth-first search because every time the depth-bound is increased it must start its search from scratch. However this is not true. Why?

3. If $h()$ is admissible and s is the start node, how is $h(s)$ related to the cost of the solution found by A* search?

4. It can be shown that when $h(n) = h^*(n)$, A* only expands nodes that lie on an optimal path to a goal. When this condition on $h(n)$ holds does it imply that A* will always take time linear in the solution length to find an optimal solution? Explain your answer.

5. Consider solving the Sudoku problem using A* search. The start state has some number of cells filled in, and the successors of each state are all legal ways an additional cell can be filled in. At any time we know exactly the number of cells that remain to filled. Therefore, if K cells remain to be filled at node n, we know that $h^*(n) = K$. Thus if we set our heuristic function $h(n)$ to be $h^*(n) = K$, we will have perfect heuristic knowledge in A^*.

   Explain the mistake in the above statement.

6. What happens if we use a heuristic $h()$ in A* search that does not have the guarantee that $h(n) \leq h^*(n)$ for all states n.

7. How do depth-first, breadth-first and depth-first with iterative deeping search compare in terms of their asymptotic time complexity? In terms of their asymptotic space complexity?

2 Search Space Representation

1. Consider the problem of transporting N people across a river, where each person has a certain weight. Everyone starts off on the right-hand side of the river and is to be transported to the left-hand side. Suppose also that only one boat exists, which is able to support a maximum weight of W_B.

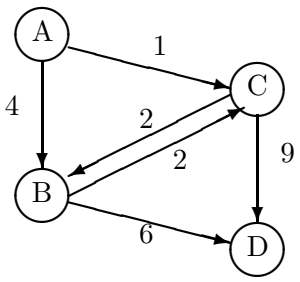
(a) Design a state space representation for this problem. Specify clearly the meaning of each component of the state representation.

(b) Design a complete set of operators in this domain based on the above representation. Note that these operators should also work for transporting any allowed subgroup of people in either direction. That is, they should not be overly specific to the actual problem we want to solve, but instead should allow one to traverse the search space in a general manner.

3 Algorithm Execution

1. Trace the execution of A* for the graph shown below.

   - Show the successive configurations of the frontier where the elements on the frontier are paths. That is, the path $n_1 \rightarrow n_2 \rightarrow n_3$ would be written $[n_1, n_2, n_3]$. Under each element of the frontier, indicate the f, g and h values of the final node in the path (e.g., if $g(m) = 5$ and $h(m) = 7$ write "12=5+7" underneath m, **Remember to get the order right,** $g(m)$ **followed by** $h(m)$**.**
   - Indicate the path that is expanded at each stage.
   - Finally show the path to the goal found by A^*.
   - A is the start state and D is the goal state.



h(A) = 8
h(B) = 3
h(C) = 7
h(D) = 0

   (a) Indicate how the search would change if cycle checking in its simplest form was to be used. In particular, once a node n is expanded it is remembered, all other paths to n on the frontier are removed, and a path to n is never subsequently put on the frontier.

   (b) What is wrong with the heuristic that caused cycle checking search to find a non-optimal path?

2. Consider the game tree shown in Figure 1.

   (a) Circle the nodes that will be visited during a minmax depth-first search that uses *alpha-beta* pruning. You should also mark any terminal node that is visited. Draw a line across each edge that leads to a subtree pruned by an alpha or beta bound.

   (b) In the first move will player MAX move to node (D) or (H)?

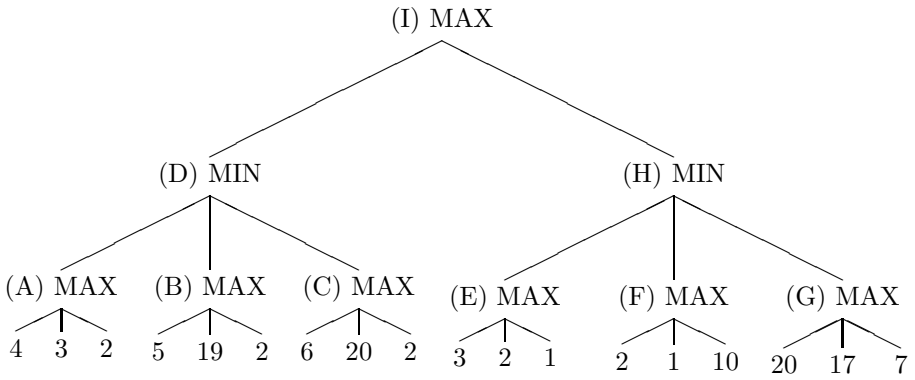


Figure 1: Game Tree

(c) Suppose that MIN does not follow a minmax strategy in playing the game. Instead suppose that at node (D) MIN would choose to move to node (C), and at node (H) MIN would choose to move to node (G). Under this strategy for MIN what would MAX's best first move be? (I.e., should MAX move to D or to H first).

3. Consider the following CSP with 3 variables X, Y and Z:

   $Dom(X) = \{1, \cdots, 10\}$, $Dom(Y) = \{5, \cdots, 15\}$, and $Dom(Z) = \{5, \cdots, 20\}$,

   and binary constraints: $C(X, Y) : X > Y$, $C(Y, Z) : Y + Z = 12$, and $C(X, Z) : X + Z = 16$.

   a) Draw the constraint graph.

   b) Are the constraints *arc consistent*? If no, apply arc consistency method repeatedly so they become arc consistent. What is the updated domain of each variable?

4. Consider the N-Queens problem. That is, the problem of placing N Queens on an $N \times N$ chessboard so that no two Queens can attack each other.

   Find the *first solution* to the 5-Queens problem by using the *Forward Checking* algorithm with *dynamic variable reordering* using the heuristic that we always instantiate next that variable with smallest remaining number of elements in its domain, breaking ties in favor of the lowest numbered variable.

   (Note, use the same CSP formulation as that used in class. That is, we have 5 variables, $Q_1, \ldots, Q_5$ each with domain $[1, 2, 3, 4, 5]$. Each variable Q_i represents the queen in the i-th row, and the assignment $Q_i = j$ means that the queen in the i-th row has been placed in the j-th column.

   Draw the search tree explored by this algorithm. **At each node** indicate

   (a) The variable being instantiated, and the value it is being assigned.

   (b) A list of the variables that have had at least one of their values pruned by the new assignment, and for such variable a list of its remaining legal values. *Note, you must follow the forward checking algorithm precisely: only prune values that would be pruned by the algorithm.*

   (c) Mark any node where a deadend occurs because of a domain wipe out (use the symbol DWO).

4 Simple Proofs

1. Let $h^*(n)$ denote the cost of an *optimal* path from n to a goal node, let $g^*(n)$ denote the cost of an *optimal* path from the start node s to n. Finally let C^* be equal to the cost on an optimal solution (optimal path from s to a goal node).

   Suppose that the heuristic used by A^* satisfies the following property

   $$h^*(n) - \epsilon \leq h(n) \leq h^*(n)$$

   for all nodes n. That is, $h(n)$ is admissible, and never underestimates by more than ϵ.

   Assuming that a path to a goal exists, prove that whenever A^* expands a node n (any node) it has found a path to n that is within ϵ of the optimal; i.e.

   $$g(n) \leq g^*(n) + \epsilon$$

   Hint, you may utilize the fact, proved in class, the f value of every node expanded by A^* using an admissible heuristic is never greater than C^*.

2. Prove that if $h(n) = h^*(n)$ for all n, then whenever A^* expands a node n, n must lie on an optimal path to a goal.