Midterm Examination

CS 540: Introduction to Artificial Intelligence

March 14, 2007

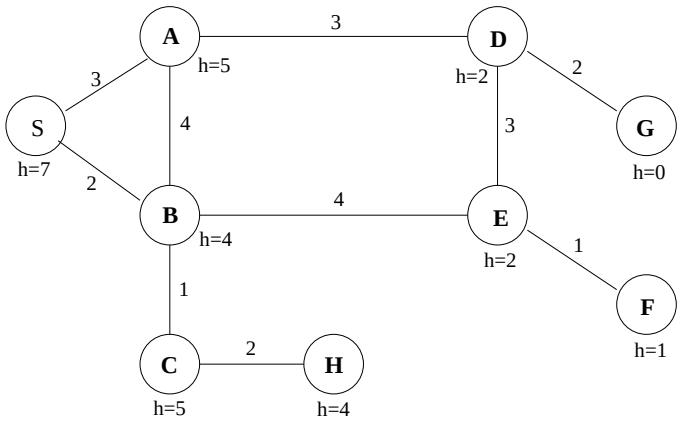
LAST (FAMILY) NAME: _____SOLUTION_____

FIRST NAME: _____

Problem	Score	Max Score
1	_____	15
2	_____	12
3	_____	10
4	_____	10
5	_____	16
6	_____	12
7	_____	15
8	_____	10
Total	_____	100

1. [15] **Search Methods**

Consider the following *state space graph* where the arcs represent the legal successors of a node, and all arcs are bi-directional, meaning the successor function can be applied from either node. The cost of moving to a successor node is given by the number on the arc. The value of a heuristic evaluation function, *h*, if computed at a state, is shown along side each node. The start state is **S** and the goal is **G**.



When a node is expanded, assume its children are put in the NODES list in alphabetical order so that the child closest to the front of the alphabet is removed before its other siblings (for all uninformed searches and for ties between siblings in informed searches). Do *not* generate a child node if that same node is an *ancestor* of the current node *in the search tree*. When selecting a node from NODES, in case of ties between non-siblings, use FIFO order to select the node that has been in NODES longest.

Give the sequence of nodes as they are **removed** from the NODES list (for expansion or before halting at the goal) for each of the following search methods.

(a) [5] Uniform-Cost search

SBACHAEDBFG

(b) [5] A* heuristic search

SBACEDFG

(c) [5] Greedy Best-First search

SBEFDG

2. [12] **Search: Short Answer Questions**

(a) [3] Under what general conditions will A* search expand the same nodes as Breadth-First search? (Do not consider issues related to ties.)

When the cost of all operators is the same and the heuristic function returns a constant value.

(b) [2] True or False: Greedy Best-First search with $h(n) = 0$ for all nodes n is guaranteed to find an optimal solution if all arc costs are 1.

False or True depending on tie-breaking.

(c) [2] True or False: In a finite search space containing no goal state, Algorithm A* will always explore all states.

True

(d) [3] Say we have a search space that is very large with a very large branching factor at most nodes, there may be infinite paths in the search space, and we have no heuristic function. What search method would be good to use in this situation and why?

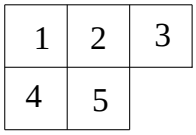
Depth-first Iterative Deepening search because it uses little space (like DFS) but guarantees finding a solution even when there may be paths of infinite length.

(e) [2] True or False: Simulated Annealing with a constant, positive temperature at all times is the same as Hill-Climbing.

False because there is always some non-negative probability of moving to a worse state in Simulated Annealing.

3. [10] **Constraint Satisfaction**

Consider the problem of assigning colors to the five squares on board below such that horizontally adjacent and vertically adjacent squares do not have the same color. Assume there are possible two colors, red (R) and black (B). Formulated as a constraint satisfaction problem, there are five variables (the squares) and two possible values (R, B) for each variable.



(a) [3] If initially every variable has both possible values and we then assign variable 1 to have value R, what is the result of the Forward Checking algorithm?

Forward checking propagates constraints from the current assigned variable to all adjacent unassigned variables, which in this case are variables 2 and 4. This results in the following domains for the variables: 1 = {R}, 2 = {B}, 3 = {R,B}, 4 = {B}, 5 = {R,B}.

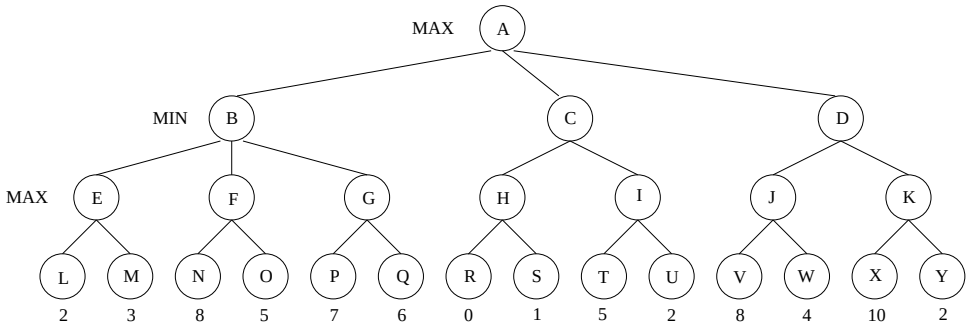
(b) [3] If initially every variable has both possible values and the arc consistency procedure is run, what are the resulting domains for each of the variables?

None of the domains change, with each variable still having possible values {R,B}.

(c) [4] If initially every variable has both possible values and we then assign variable 5 to have value B, what is the result of the Arc Consistency algorithm?

1 = {B}, 2 = {R}, 3 = {B}, 4 = {R}, 5 = {B}

4. [10] **Adversarial Search**
Consider the following game tree in which the root corresponds to a MAX node and the values of a static evaluation function, if applied, are given at the leaves.



(a) [2] What is the minimax value computed at the root node for this tree?

8

(b) [1] What move should MAX choose?

D

(c) [3] Which nodes are *not* examined when **Alpha-Beta Pruning** is performed? Assume children are visited left to right.

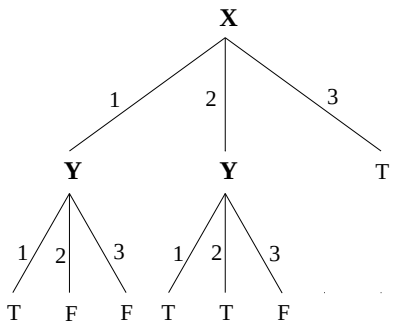
O, Q, I, T, U, Y

(d) [4] Is there a different ordering for the children of the root for which *more* pruning would result by Alpha-Beta? If so, state the order. If not, say why not.

Yes, when the children are ordered D, B, C, which is best to worst ordering of A's children.

5. [16] **Decision Trees**

(a) [5] Consider a domain in which there are two attributes, X and Y, each with three possible values, 1,2,3. The concept to be learned is X ≥ Y. (E.g., if X=3 and Y=1, then the output is true.) Draw a decision tree that represents this concept (assume your training set contains all possible combinations of values).



(b) [3] How is the problem of overfitting the training data exhibited when constructing a decision tree?

By adding nodes to the decision tree (usually due to irrelevant attributes) so that the tree correctly classifies every example in the training set but generalizes poorly for unseen examples.

(c) [8] Say we build a decision tree by adding nodes until there is one or a small number of examples at a node, which we then make a leaf. Then we prune the tree by deleting leaves until the "score" of the tree starts to get worse. The question is, how to score each possible pruning of the tree? For each of the following possible definitions of a scoring method, explain whether or not it would be a good idea to use it, and why or why not.

(i) The score is the percentage correct classification using the tree on the training set used to build the tree.

This is not a good idea because the original tree is constructed to maximize performance on the training set. So, pruning any part of the tree will reduce performance on the training set, so no pruning will be done.

(ii) The score is the percentage correct classification using the tree on a separate set of examples from the training set used to build the tree.

This is a good idea because the separate set of examples will provide an independent check on whether pruning a node is likely to increase or decrease performance on unseen data.

6. [12] **Propositional Logic**

(a) [3] Define **soundness** in terms of Propositional Logic (PL) sentences α and β, and |= and |−.

$$\alpha \mid\!- \beta \Rightarrow \alpha \mid\!= \beta$$

which means that if β is derivable from α, then β logically follows from α.

(b) [3] Is the PL sentence $((P \Rightarrow Q) \wedge Q) \Rightarrow P$ valid, unsatisfiable or satisfiable? Justify (i.e., prove) your answer.

Satisfiable. When P=T, Q=T, the sentence is T, but when P=F, Q=T, the sentence is F.

(c) [6] Given two arbitrary sentences α and β in PL, α |= β if and only if

(i) [2] the sentence $\alpha \Rightarrow \beta$ is _____valid_____

(ii) [2] the sentence $\alpha \wedge \neg\beta$ is _____unsatisfiable_____

(iii) [2] $\alpha \wedge \neg\beta$ |− _____false_____

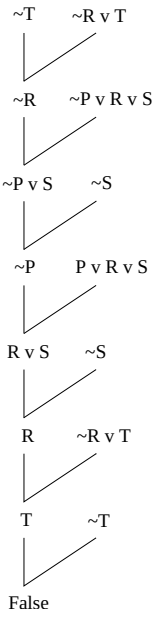
7. [15] **Deductive Inference in Propositional Logic**
Consider the following 4 premise sentences in PL:

$P \Rightarrow (R \vee S), \neg P \Rightarrow (R \vee S), \neg S, (R \vee U) \Rightarrow T$

(a) [4] Convert the premise sentences into conjunctive normal form (CNF) and show the result as a set of **clauses**.

$\neg P \vee R \vee S, P \vee R \vee S, \neg S, \neg R \vee T, \neg U \vee T$

(b) [8] Prove the sentence T is true given the premises using the **Resolution Refutation algorithm**. Show your result as a proof tree.



(c) [3] Can the Resolution Refutation algorithm be used to show that a given sentence in PL is *not* true given a set of premise sentences in PL? If yes, explain how. If not, explain why not.

Yes. There are two possible ways to do this in this case. First, since everything is finite in PL, there are only a finite number of resolvents that can be generated using the resolution rule of inference. After generating all of them based on the premises and the negation of the query sentence, if the empty clause has not been derived, the query sentence is not true given the premises. A second way is to negate the given sentence and then prove it that way (which involves negating again when using Resolution Refutation). Note that the second method might fail to prove it, but the first way still can.

8. [10] **Representation in First-Order Logic**
For each of the following sentences in English, is the accompanying sentence in First-Order Logic a good translation? If yes, answer "yes." If no, explain why not and then give a correct answer.

(a) [5] No two people have the same social security number.

$\neg\, \exists x, y, n \;\; (IsPerson(x) \wedge IsPerson(y)) \Rightarrow (HasSS\#(x, n) \wedge HasSS\#(y, n))$

There are two problems here: first, this sentence uses $\Rightarrow$ when it should use $\wedge$ because of $\exists$. Second, this sentence currently says that no person has a social security number because it doesn't restrict itself to cases where x and y are not equal. A corrected version is:

$\neg\, \exists x, y, n \;\; IsPerson(x) \wedge IsPerson(y) \wedge \neg(x = y) \wedge HasSS\#(x, n) \wedge HasSS\#(y, n)$

(b) [5] Everyone's social security number has nine digits.

$\forall x, n \;\; IsPerson(x) \Rightarrow (HasSS\#(x, n) \wedge NumDigits(n, 9))$

This is not correct because it says that everyone has every social security number. A corrected version is:

$\forall x, n \;\; (IsPerson(x) \wedge HasSS\#(x, n)) \Rightarrow NumDigits(n, 9)$