Midterm Examination

CS 540: Introduction to Artificial Intelligence

March 26, 2003

LAST (FAMILY) NAME: _____SOLUTION_____

FIRST NAME: _____

Problem	Score	Max Score
1	———————	8
2	———————	6
3	———————	14
4	———————	16
5	———————	20
6	———————	8
7	———————	8
8	———————	20
Total	———————	100

1. [8] **Search**
Determine whether each of the following statements is true or false.

(a) [2] 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.

(b) [2] Breadth-First search is a complete algorithm even if all operators do not have the same cost.

True because breadth-first search does not use operator costs, only the depth of a node.

(c) [2] When Hill-Climbing and Greedy Best-First searches use the same admissible heuristic function, they both expand the same set of nodes in any search space.

False

(d) [2] If *h1* and *h2* are both admissible heuristics, then it is always preferable to use the heuristic *h3(n) = max(h1(n), h2(n))* over the heuristic *h4(n) = min(h1(n), h2(n))*.

True

2. [6] **Beam Search**
Beam search is a version of Best-First search that uses an evaluation function of the form $f(n)=h(n)$, but at each step only the W best nodes are kept on the OPEN list, where W is a user-defined constant that defines the "beam width."

(a) [2] Is Beam search with beam width 3 and an admissible heuristic function guaranteed to find an optimal solution? Briefly explain why or why not.

No. Beam search may throw away a node on the optimal path.

(b) [2] What is a major advantage of Beam search over Depth-First search?

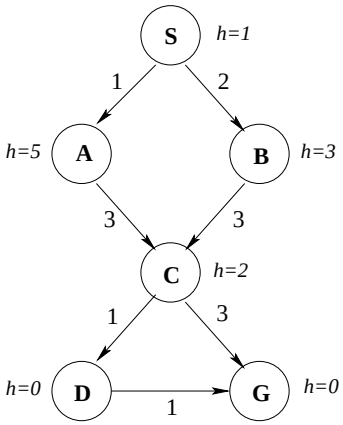
It can avoid going down a single path infinitely far, thus using less space.

(c) [2] What is a major advantage of Beam search over Hill-Climbing search?

Because it is a form of best-first search, it can backtrack and may avoid getting stuck at a local maximum.

3. [14] **Heuristic Search**

Consider the search space shown below with start state **S** and goal state **G**. The values of a heuristic function, *h*, are shown at each node.



Unfortunately, we do not know the cost of the operators that determine the costs associated with each arc in the graph. Fortunately, we do have the following trace of the OPEN list produced during the execution of the **A* algorithm**. Note: The front of the OPEN list is on the left, and repeated states are removed, keeping only the minimum *f* value.

1. {(**S**, *f*=1)}
2. {(**B**, *f*=5), (**A**, *f*=6)}
3. {(**A**, *f*=6), (**C**, *f*=7)}
4. {(**C**, *f*=6)}
5. {(**D**, *f*=5), (**G**, *f*=7)}
6. {(**G**, *f*=6)}

Based on this information, compute each of the arc costs and add them to the search graph above. Show your work.

In the A* algorithm, $f=g+h$, so $g=f-h$. The order of node expansion tells us which edge was followed at each step. At Step 2 h(B)=3, so c(B,S) = 5 - 3 = 2. Also, h(A) = 5, so C(A,S) = 6 - 5 = 1. And, g(A) = c(A,S) = 1, and g(B) = c(B,S) = 2.

At Step 3, h(C) = 2, so c(C,B) = f(C-via-B) - g(B) - h(C) = 7 - 2 - 2 = 3. And, g(C) = c(C,B) + g(B) = 3 + 2 = 5.

At Step 4, h(C) = 2, so c(C,A) = f(C-via-A) - g(A) - h(C) = 6 - 1 - 2 = 3. And now we update g(C) because we've found a cheaper path; g(C) = c(C,A) + g(A) = 3 + 1 = 4.

At Step 5, h(D) = 0, so c(D,C) = f(D-via-C) - g(C) - h(D) = 5 - 4 - 0 = 1. And, g(D) = c(D,C) + g(C) = 1 + 4 = 5. Similarly, h(G) = 0, so c(G,C) = f(G-via-C) - g(C) - h(G) = 7 - 4 - 0 = 3. And, g(G) = c(G,C) + g(C) = 3 + 4 = 7.

At Step 6, h(G) = 0, so c(G,D) = f(G-via-D) - g(D) - h(G) = 6 - 5 - 0 = 1. And, g(G) = c(G,D) + g(D) = 1 + 5 = 6.

4. [16] **Constraint Satisfaction**

Consider the following train scheduling problem: There are 4 trains, T1, T2, T3, and T4, and 3 locomotives, L1, L2, and L3. The following table shows the schedule for each of the trains:

Train	In Use
T1	8am - 10am
T2	9am - 1pm
T3	noon - 2pm
T4	11am - 3pm

In addition, there are the following problem assumptions:

(i) Each train must be pulled by a locomotive.
(ii) Each locomotive can pull only one train at a time.
(iii) If a locomotive is not in use, it can immediately move to any station for use by any train.
(iv) L3 is too small to pull T3.
(v) L2 and L3 are too small to pull T4.

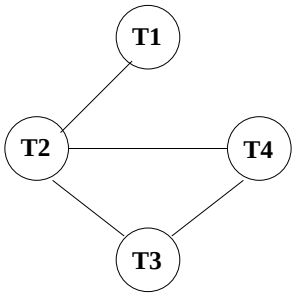
One way of formulating this problem as a constraint satisfaction problem is to let each train, T1, ..., T4, represent a variable, and each locomotive, L1, L2, L3, represent the possible values.

(a) [4] What are all the initial possible values for each variable, T1, ..., T4, given the assumptions above?

```
T1 = {L1, L2, L3}
T2 = {L1, L2, L3}
T3 = {L1, L2}
T4 = {L1}
```

(b) [4] Draw the **constraint graph** that represents this problem. For each arc you include, briefly explain what constraint is associated with it.

The constraints in this formulation are added to express the fact that a locomotive cannot pull trains that overlap in time. (Also, there would be constraints that say that two locomotives cannot pull the same train.)



(c) [8] We want to use **Backtracking search with Forward Checking** to solve this problem given the initial domains in (a), assuming the variable ordering T1 then T2 then T3 then T4, and the value ordering L1 then L2 then L3. Show the domain of each variable after each of the first TWO steps of backtracking search. In other words, fill in the following table:

	T1	T2	T3	T4
Initial Domain	L1,L2,L3	L1,L2,L3	L1,L2	L1
Domain after Step 1	L1	L2,L3	L1,L2	L1
Domain after Step 2	L1	L2	L1	L1

5. [20] **Decision Trees**

Consider collecting data where each example has three attributes, X1, X2, and X3, and the possible values of X1 are A and B, the possible values of X2 are C and D, and the possible values of X3 are E, F, and G. The goal is to construct a decision tree for a Boolean concept given the following examples:

X1	X2	X3	Class
A	C	G	+
A	D	F	+
A	C	F	–
A	D	F	+
B	D	G	–
B	D	G	–
B	C	G	+
B	C	F	–

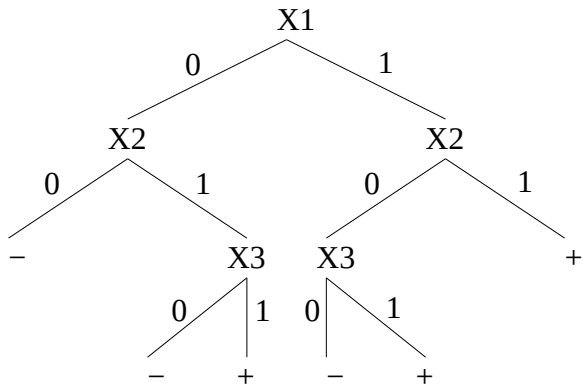
(a) [12] Compute the **information gain** for *each* of the three features, X1, X2, and X3. Show your work. For your information, I(1,0)=0, I(0.5,0.5)=1, I(0.75, 0.25)=0.81, and I(0.875, 0.125)=0.54.

```
Remainder(X1) = 4/8 I(3/4, 1/4) + 4/8 I(1/4, 3/4) = 0.81

Gain(X1) = I(4/8, 4/8) - Remainder(X1) = 1 - 0.81 = 0.19

Remainder(X2) = 4/8 I(2/4, 2/4) + 4/8 I(2/4, 2/4) = 1

Gain(X2) = I(4/8, 4/8) - Remainder(X2) = 1 - 1 = 0

Remainder(X3) = 0/8 + 4/8 I(2/4, 2/4) + 4/8 I(2/4, 2/4) = 1

Gain(X3) - I(4/8, 4/8) - Remainder(X3) = 1 - 1 = 0
```

(b) [2] Which attribute would the decision tree learning algorithm choose for the root node when using the maximum-gain criterion?

```
Gain(X1) > Gain(X2) = Gain(X3), so pick X1.
```

(c) [4] Now assume X1, X2, and X3 are all two-valued attributes, each with possible values 0 and 1. Draw a decision tree representing the function "**2 or more**." That is, the classification is true if at least 2 attributes have value 1; otherwise the classification is false.



(d) [2] Pruning nodes in a decision tree is usually performed to deal with what problem?

`Overfitting`

6. [8] **Logic**
Answer each of the following questions true or false.

(a) [2] $A <=> B \models \neg A \vee B$

`True`

(b) [2] $(A <=> B) \wedge (\neg A \vee B)$ is satisfiable.

`True`

(c) [2] $(A <=> B) <=> C$ has the same number of models as $(A <=> B)$ for any fixed set of propositional symbols that includes *A*, *B*, and *C*.

`True`

(d) [2] The sentence $(A \vee B) => (C \vee D)$ can be represented by an equivalent set of Horn clauses.

`False`

7. [8] **First-Order Logic and English Translation**
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) [4] No two people have the same social security number.

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

There are two problems here: first, this sentence uses => 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) [4] Everyone's social security number has nine digits.

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

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

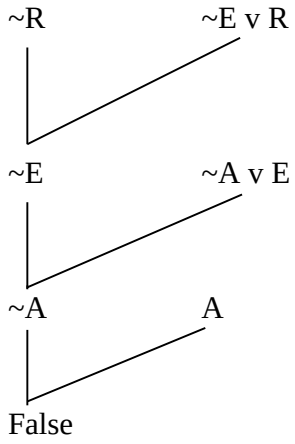
$\forall x,n \;\; (IsPerson(x) \wedge HasSS\#(x,n)) => NumDigits(n,9)$

8. [20] **Deductive Inference in Propositional Logic**
You are given the following set of true sentences in Propositional Logic:

(i) A
(ii) B
(iii) C
(iv) $(A \wedge B) \Rightarrow D$
(v) $A \Rightarrow E$
(vi) $A \Rightarrow F$
(vii) $D \Rightarrow E$
(viii) $D \Rightarrow F$
(ix) $E \Rightarrow R$
(x) $G \Rightarrow R$
(xi) $(F \wedge C) \Rightarrow R$
(xii) $(E \wedge F) \Rightarrow R$

(a) [10] Use the **Resolution Refutation** algorithm to prove sentence R is true. Give your answer as a proof tree.

After converting the above to clause form and negating the goal to be $\neg R$, we can then construct the proof tree:



(b) [10] Give a **Backward-Chaining** proof that sentence R is true by constructing an AND-OR proof tree/graph.