

# 6.034 Spring 2009

Final Exam

May 19, 2009

**Please read these notes before starting the exam.**

- This exam has 22 pages. Make sure you have them all.
- The last sheet is just extra copies of diagrams. You can tear it off.
- This exam is open-note.
- Calculators are allowed, but probably not needed.
- Computers are not allowed.

Problem	Score / Max
1	/ 21
2	/ 25
3	/ 35
4	/ 22
5	/ 10
6	/ 18
7	/ 27
Total	/ 158

## Problem 1: Bayesian Networks (21 points)

### Part A (3 points).

To represent the arbitrary probabilistic relationships among a set of  $n$  discrete variables,  $\{x_1, x_2, \dots, x_n\}$ , we can list the *full joint probability table*. However, this will require a number of rows exponential in  $n$ .

If  $n = 30$  and 20 of the variables are binary, whereas the rest are ternary, give an expression for the number of rows in the full joint probability table.

### Part B (3 points).

The simplest relationship among all these variables is if they are all *independent* of each other. Give a definition of independence.

### Part C (3 points).

Give the number of probabilities you need to specify in order to fully describe the probability distribution under the assumption of total independence in part B. (You may assume  $n = 30$  and the arity of the variables given in part A, or just answer in symbolic terms.)

**Part D (3 points).**

Generally, a probabilistic model in which all the variables are mutually independent can represent only a very limited set of circumstances in the world. Give an example of one such domain.

**Part E (3 points).**

A slightly more complex probabilistic set of relations among a set of  $n$  variables is if all but the first (i.e.,  $\{x_2, \dots, x_n\}$ ) are dependent on  $x_1$ , but *conditionally independent* of each other. Schematically draw such a set of relationships as a network, and give the definition of conditional independence.

**Part F (3 points).**

Assuming that the first 20 variables  $\{x_1, \dots, x_{20}\}$  are still binary and the rest are ternary (and of course  $n > 20$ ), how many probabilities would you need to specify in order to be able to derive the joint probabilities of any combinations of values for all the variables?

**Part G (3 points).**

Now consider the case where a variable, say  $x_4$ , has three others on which it depends,  $x_1, x_2, x_3$ , and assume that each of these is binary. How many probabilities do you need to specify to completely describe an arbitrary relationship of this form? (For this, ignore any other possible variables.)

## Problem 2: Network Inference (25 points)

The network of Figure 1, which we discussed in class, forms a poly-tree relating the probability of burglary, earthquake, an alarm sounding, and two somewhat unreliable neighbors, John and Mary calling to report the alarm.

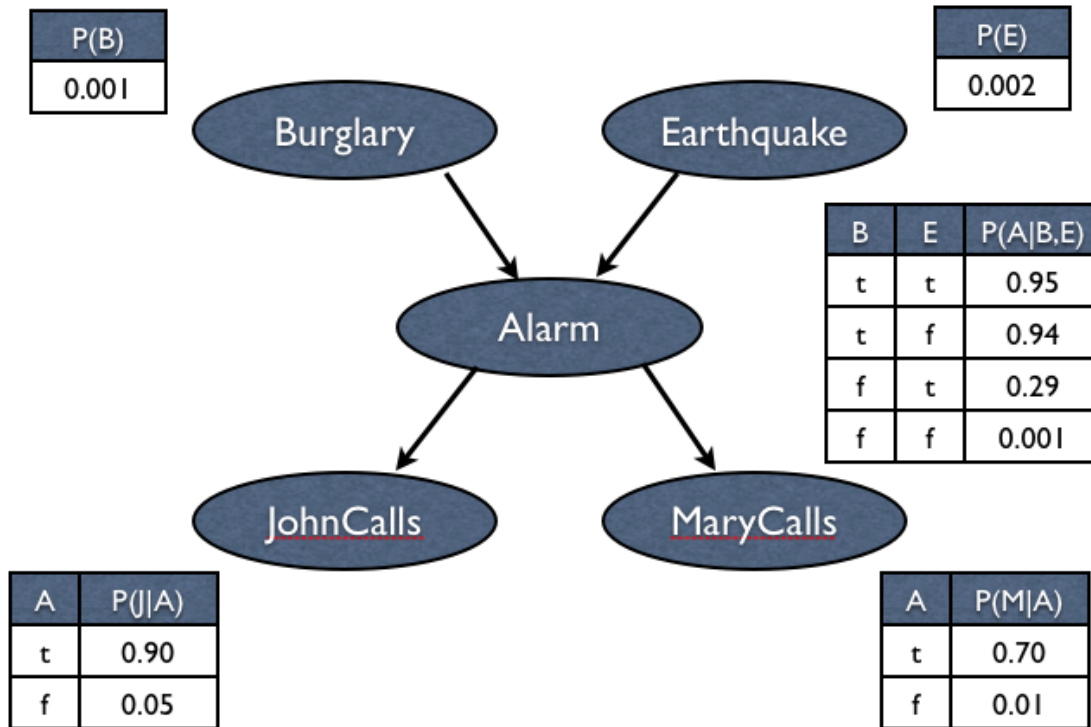


Figure 1: Pearl's example "alarm" network.

You can answer A-D with either a number or an expression that shows you could calculate that number.  
Show your work.

**Part A (5 points).**

What is the *a priori* probability that the alarm sounds, assuming we know nothing about the actual status of any of the other nodes?

**Part B (5 points).**

If Mary telephones, what is the *a posteriori* probability of the alarm actually sounding?

**Part C (5 points).**

Given her phone call, what is the *a posteriori* probability that a burglary has actually taken place, assuming that we still don't know the state of any of the other variables?

**Part D (5 points).**

Still given only the fact that Mary called, what is the probability that John calls?

**Part E (5 points).**

Change the network to reflect that fact that Mary, though fairly insensitive to your alarm going off, is actually much more likely to notice it if the shaking of an earthquake has actually disturbed her. What link(s) would you need to add to this network, and what conditional probability distribution(s) would you have to augment?

### Problem 3: Search (35 points)

Figure 2 is a graph to be searched, starting at A and ending at G. The  $h$  values are the heuristic estimates, and the numbers on the edges are the actual costs. Assume that the children of a node are ordered in alphabetical order; also use the alphabetical order to break ties, if necessary.

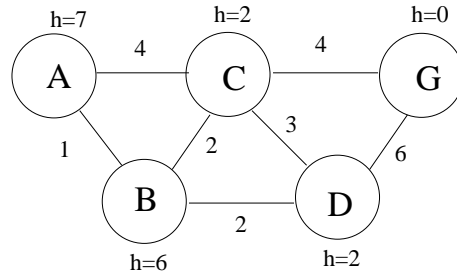


Figure 2: Search graph.

#### Part A (20 points).

A1 (15 points).

Perform an  $A^*$  search with an expanded list. Fill in the following table. Which path is returned?

Expanded Node	Partial Path leading to the node	Total estimated cost using partial path

**Path returned:**

A2 (5 points).

Did  $A^*$  search with expanded list return the optimal path? Explain why in terms of the admissibility and/or consistency of the heuristics.

**Part B (5 points).**

*B1 (4 points).*

Perform a depth-first search, without using any visited or expanded lists. Show the sequence of expanded nodes. Which path is returned?

*B2 (1 point).*

**Path returned:**

**Part C (5 points).**

*C1 (4 points).*

Perform a uniform cost search, using an expanded list. Show the sequence of expanded nodes. Which path is returned?

*C2 (1 point).*

**Path returned:**



**Part D (5 points).**

*D1 (4 points).*

Of the search algorithms covered in class, which one requires the smallest number of expansions before returning a path? Which path is returned?

*D2 (1 points).*

**Path returned:**

## Problem 4: Game Search (22 points)

### Part A (6 points).

You are watching two people play a game called Mini-Four, which is just like Connect Four, but played on a smaller (4x4) board. In this game, the black player and white player alternate dropping a piece into one of the four columns; the piece falls into the lowest row not already occupied. If a player can get four pieces of the same color in a row, he wins.

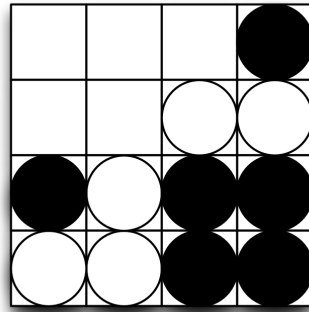


Figure 3: The in-progress game.

*A1 (1 points).*

The game has already progressed to the point seen in Figure 3. Black went first, and so it is now white's turn to move. You notice that white has a move that guarantees a win. Determine and circle the move white should make to guarantee victory.

Column 1   Column 2   Column 3   Column 4

*A2 (5 points).*

Mini-Four is a very small game, and so easy to exhaustively search. As the size of the board gets bigger, even just to the standard 6x7, this becomes tough. For Minimax or  $\alpha$ - $\beta$  to work on these big games, we must stop early and use a static evaluator on non-final board states, as you just did in the last part.

We want you to design a static evaluator for these Mini-Four-like games. That is, you should give a function or algorithm that takes in a board state (the location and color of the pieces) and gives back a number. The number should be large when the board is good for the current player, and small when it's bad. If it makes it easier for you to write, you may assume the current player is white.

Please remember that the static evaluator should only consider the current state. You don't need to look into the future; that's what Minimax does for you!

**Part B (16 points).**

You write two programs to play Mini-Four, one using Minimax and one using  $\alpha$ - $\beta$  from left to right. We provide some arbitrary game search tree they're both trying to evaluate in Figures 4 and 5. The leaves are labeled with their static evaluation values.

*B1 (7 points).*

Use Minimax to evaluate the game tree as a maximizer. Fill in the blanks in Figure 4 with each node's Minimax value.

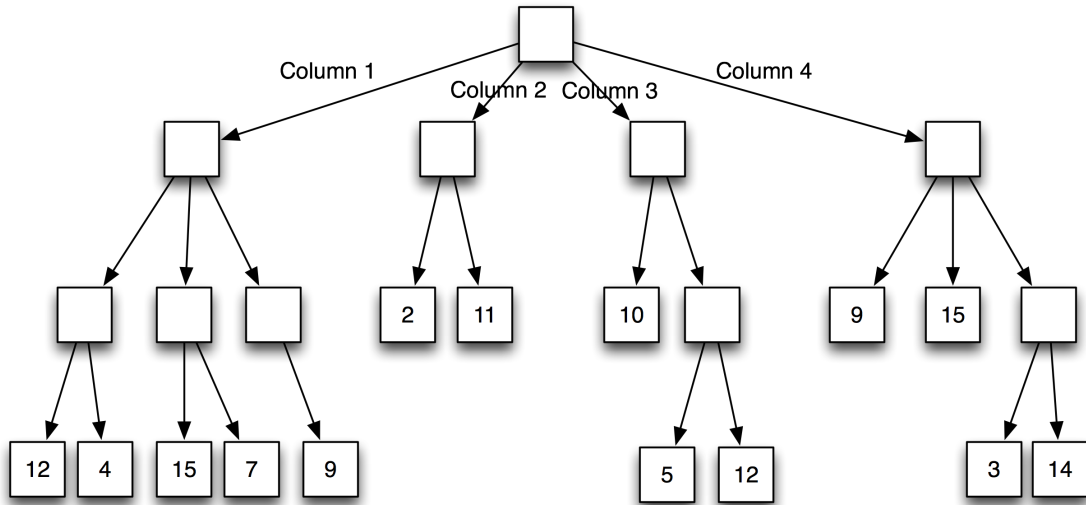


Figure 4: The Minimax search tree.

*B2 (1 points).*

What move does Minimax say the maximizer should make?

Column 1   Column 2   Column 3   Column 4

B3 (7 points).

Use  $\alpha$ - $\beta$  to evaluate the game tree as a maximizer. Work left to right. Cross out nodes that  $\alpha$ - $\beta$  does not evaluate (both the blank ones and the ones we've given values for). Fill in the blanks in Figure 4 with each node's Minimax value.

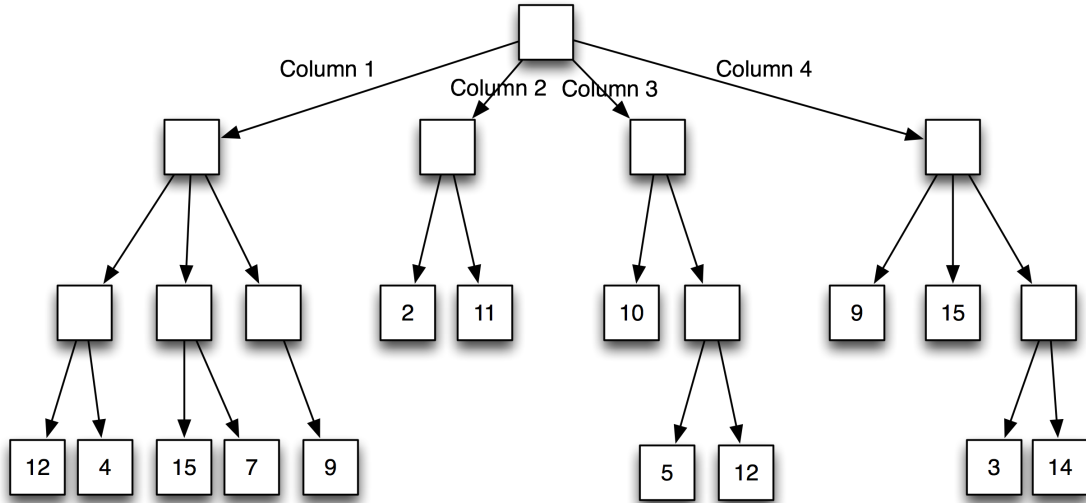


Figure 5: The  $\alpha$ - $\beta$  search tree.

B4 (1 points).

What move does  $\alpha$ - $\beta$  say the maximizer should make?

Column 1   Column 2   Column 3   Column 4

## Problem 5: Decision Trees (10 points)

You have three horses you can run in the Belmont Stakes, “USS Kelvin”, “Five Dollar Milkshake”, and “Totes Jelly”. Once you select a horse, then you need to select a jockey. You have two jockeys you can use, named “T-Dum” and “T-Dee”. T-Dum is the better jockey and will win the race 80% of the time on any horse; T-Dee is not as good, winning 60% of the time.

Based on the betting, your utility with T-Dum is as follows:

Horse	Winning Worth	Losing Worth
USS Kelvin	100	10
Five Dollar Milkshake	50	10
Totes Jelly	100	20

Using T-Dee, your utility function is as follows:

Horse	Winning Worth	Losing Worth
USS Kelvin	50	50
Five Dollar Milkshake	50	50
Totes Jelly	100	80

**Part A (8 points).**

Draw the entire decision tree for this problem, including the values for each of the decision and chance nodes.

**Part B (2 points).**

Which horse and which jockey do you choose to run the race for you?

## Problem 6: Constraint Satisfaction (18 points)

Misters White, Orange, Pink, and Brown are planning a crime while sitting in a small diner. 4 chairs labeled 1,2,3 and 4 fit around the diner's table.

You will model where each person sits as a constraint satisfaction problem. The variables will be the diners, {(W)hite, (O)range, (P)ink, (B)row}. The domain for each is {1,2,3,4}. Chair pairs 1 and 2, 2 and 3, 3 and 4, and 4 and 1 are adjacent to each other. Chairs 1 and 3 are across from each other, as are chairs 2 and 4.

The criminal masterminds are petty and finicky, and this puts constraints on where they can sit.

1. No two people can sit in the same chair.
2. Mr. Pink will only sit in chair 3.
3. Mr. White will not sit across from Mr. Pink.
4. Mr. Orange will sit adjacent to Mr. Pink.
5. Mr. Brown will only sit across from Mr. White.

### Part A (4 points).

Using the constraints, perform full constraint propagation on this seating problem starting with domains of {1,2,3,4} for each variable B, P, W, and O. List the domains of each variable after full constraint propagation.

- B =
- P =
- W =
- O =

### Part B (2 points).

Does full constraint propagation find a solution to this problem? Is there a solution to this problem? Explain.

### Part C (4 points).

Mr. Brown now will only sit adjacent to Mr. White instead of across from him. After full constraint propagation, now what are the domains for each person?

- B =
- P =
- W =
- O =



**Part D (4 points).**

Using the variable order B, P, W, O, and starting with the domains from the above full constraint propagation, show the sequence of variable assignments during a pure backtracking search. Assign chair values in numerical order.

**Part E (4 points).**

Do the same as the previous problem, except this time show the sequence of variable assignments during backtracking *with forward checking*.

## Problem 7: Search and CSP (27 points)

### Part A (15 points).

Consider the following search problem formulation with finitely many states:

**States:** there are  $d + 2$  states:  $\{s_s, s_g\} \cup \{s_1, \dots, s_d\}$

**Initial state:**  $s_s$

**Successor function:**  $\text{Succ}(s)$  generates at most  $b$  successors

**Goal test:**  $s_g$  is the only goal state

**Step cost:** each step has a cost of 1

*A1 (2 points).*

Suppose an optimal solution has cost  $n$ . If the goal is reachable, what is the upper bound on  $n$ ?

*A2 (2 points).*

Suppose we must solve this search problem using Breadth-First Search, but with limited memory. Specifically, assume we can only store  $k$  states during search. Give a bound on  $n$  for which the search will fit in the available memory.

*A3 (2 points).*

Would any other search procedure allow problems with substantially deeper solutions to be solved? Either argue why not, or give a method along with an improved bound on  $n$ .

*A4 (5 points).*

If we knew the exact value of  $n$ , we could formulate a CSP whose complete assignment specifies an optimal solution path  $(X_0, X_1, \dots, X_n)$  for this search problem. State binary and unary constraints which guarantee that a satisfying assignment is a valid solution:

**Variables:**  $X_0, X_1, \dots, X_n$

**Domains:**  $Dom(X_i) = \{s_s, s_g\} \cup \{s_1, \dots, s_d\}$

**Constraints:**

*A5 (4 points).*

After reducing the domains of any variables with unary constraints, suppose we then make all arcs  $X_i \rightarrow X_{i-1}$  consistent, processed in order from  $i = 1$  to  $n$ . Next, we try to assign variables in reverse order, from  $X_n$  to  $X_0$ , using backtracking DFS. Why is this a particularly good variable ordering?

**Part B (12 points).**

Each problem is worth 2 points. Provide a short explanation that justifies your answer. (Without such explanation, you will not get credit for your answers.)

*B1 (2 points).*

If a search method is guaranteed to find an optimal solution on trees, then that method is also guaranteed find an optimal solution when applied to general graphs.

*B2 (2 points).*

An optimal solution path for a search problem with positive costs will never have repeated states.

*B3 (2 points).*

Admissible heuristics are always more effective than inadmissible ones.

*B4 (2 points).*

If one search heuristic  $h_1(s)$  is admissible and another one  $h_2(s)$  is inadmissible. then  $h_3(s) = \min(h_1(s), h_2(s))$  will be admissible.

*B5 (2 points).*

Alpha-beta pruning can alter the computed minimax value of the root of a game tree.

*B6 (2 points).*

When doing alpha-beta pruning on a game tree which is traversed from left to right, the leftmost branch will never be pruned.

THE END!

# Extra Diagrams

