Assignment 2

Local and Classical Search - Adversarial Search - Constraint Satisfaction Problems - Bayesian Networks and Exact Inference - Approximate Inference

Deadline: November 11, 11:55pm.
Perfect score: 100.

Assignment Instructions:

Teams: Assignments should be completed by teams of two students. No additional credit will be given for students that complete an assignment individually. Please inform the TAs as soon as possible about the members of your team so they can update the scoring spreadsheet (find the TAs' contact info under the course's website: `http://www.abdeslam.net/cs520`).

Submission Rules: Submit your reports electronically as a PDF document through Sakai (`sakai.rutgers.edu`). For programming questions, you need to also submit a compressed file via Sakai, which contains your code. Do not submit Word documents, raw text, or hardcopies etc. Make sure to generate and submit a PDF instead. Each team of students should submit only a single copy of their solutions and indicate all team members on their submission. Failure to follow these rules will result in lower grade in the assignment.

Late Submissions: No late submission is allowed. 0 points for late assignments.

Extra Credit for LaTeX: You will receive 10% extra credit points if you submit your answers as a typeset PDF (using LaTeX, in which case you should also submit electronically your source code). There will be a 5% bonus for electronically prepared answers (e.g., on MS Word, etc.) that are not typeset. If you want to submit a handwritten report, scan it and submit a PDF via Sakai. We will not accept hardcopies. If you choose to submit handwritten answers and we are not able to read them, you will not be awarded any points for the part of the solution that is unreadable.

Precision: Try to be precise. Have in mind that you are trying to convince a very skeptical reader (and computer scientists are the worst kind...) that your answers are correct.

Collusion, Plagiarism, etc.: Each team must prepare its solutions independently from other teams, i.e., without using common notes, code or worksheets with other students or trying to solve problems in collaboration with other teams. You must indicate any external sources you have used in the preparation of your solution. Do not plagiarize online sources and in general make sure you do not violate any of the academic standards of the department or the university. Failure to follow these rules may result in failure in the course.

Problem 1: Answer the following questions on informed search and heuristics.

a) Which of the following are admissible, given admissible heuristics h_1, h_2? Which of the following are consistent, given consistent heuristics h_1, h_2?

- $h(n) = min\{h_1(n), h_2(n)\}$ [3 points]
- $h(n) = wh_1(n) + (1 - w)h_2(n),$ where $0 \leq w \leq 1$ [3 points]
- $h(n) = max\{h_1(n), h_2(n)\}$ [3 points]

b) Consider an informed, best-first search algorithm in which the objective function is $f(n) = (2 - w)g(n) + wh(n)$. For what values of w is this algorithm guaranteed to be optimal when h is admissible (consider the case of `TREE-SEARCH`)? What kind of search does this perform when $w = 0$? When $w = 1$? When $w = 2$? [6 points]

Make sure you provide formal proofs to the above questions.

(15 points total)

Problem 2: Simulated annealing is an extension of hill climbing, which uses randomness to avoid getting stuck in local maxima and plateaux.

a) For what types of problems will hill climbing work better than simulated annealing? In other words, when is the random part of simulated annealing not necessary?

b) For what types of problems will randomly guessing the state work just as well as simulated annealing? In other words, when is the hill-climbing part of simulated annealing not necessary?

c) Reasoning from your answers to parts (a) and (b) above, for what types of problems is simulated annealing a useful technique? In other terms, what assumptions about the shape of the value function are implicit in the design of simulated annealing?

d) As defined in your textbook, simulated annealing returns the current state when the end of the annealing schedule is reached and if the annealing schedule is slow enough. Given that we know the value (measure of goodness) of each state we visit, is there anything smarter we could do?

(e) Simulated annealing requires a very small amount of memory, just enough to store two states: the current state and the proposed next state. Suppose we had enough memory to hold two million states. Propose a modification to simulated annealing that makes productive use of the additional memory. In particular, suggest something that will likely perform better than just running simulated annealing a million times consecutively with random restarts. [Note: There are multiple correct answers here.]

(10 points)

Problem 3: *Approximately Optimal Search.* The two objectives of finding a solution as quickly as possible during informed search and finding an optimal solution are often conflicting. In some problems, one may design two heuristic functions h_A and h_N, such that h_A is admissible and h_N is not admissible, with h_N resulting in much faster search most of the time. Then, one may try to take advantage of both functions.

(a) A best-first search algorithm called A^*_ϵ uses the evaluation function $f(N) = g(N) + h_A(N)$. During each iteration, A^*_ϵ expands a node N such that $f(N) \leq (1 + \epsilon) \times min_{N \in FRINGE}\{f(N)\}$, where ϵ is any strictly positive number. Formally argue about the cost of the solution returned by A^*_ϵ. [10 points]

(b) Explain briefly how A^*_ϵ can use the second heuristic function h_N to reduce the time of the search. What tradeoff is being made in choosing ϵ? [5 points]

(15 points)

Problem 4: Consider the two-player game described in Figure 1.

a. Draw the complete game tree, using the following conventions: [3 points]

- Write each state as (s_A, s_B) where s_A and s_B denote the token locations.
- Put each terminal state in a square box and write its game value in a circle.
- Put *loop states* (states that already appear on the path to the root) in double square boxes. Since it is not clear how to assign values to loop states, annotate each with a "?" in a circle.

b. Now mark each node with its backed-up minimax value (also in circle). Explain how you handled the "?" values and why. [3 points]

c. Explain why the standard minimax algorithm would fail on this game tree and briefly sketch how you might fix it, drawing on your answer to (b). Does your modified algorithm give optimal decisions for all games with loops? [3 points]

d. This 4-square game can be generalized to n squares for any $n > 2$. Prove that A wins if n is even and loses if n is odd. [1 point]

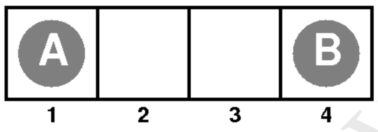


Figure 1: The start position of a simple game. Player A moves first. The two players take turns moving, and each player must move his token to an open adjacent space in *either direction*. If the opponent occupies an adjacent space, than a player may jump over the opponent to the next open space if any. (For example, if A is on 3 and B is on 2, then A may move back to 1.) The game ends when one player reaches the opposite end of the board. If player A reaches space 4 first, then the value of the game to A is +1; if player B reaches space 1 first, then the value of the game to A is -1.

(10 points)

Problem 5: Consider the problem of constructing (not solving) crossword puzzles: fitting words into a rectangular grid. The grid, which is given as part of the problem, specifies which squares are blank and which are shaded. Assume that a list of words (i.e., a dictionary) is provided and that the task is to fill in the blank squares using any subset of the list. Formulate the problem in two ways: [Hint: There might be multiple correct answers here.].

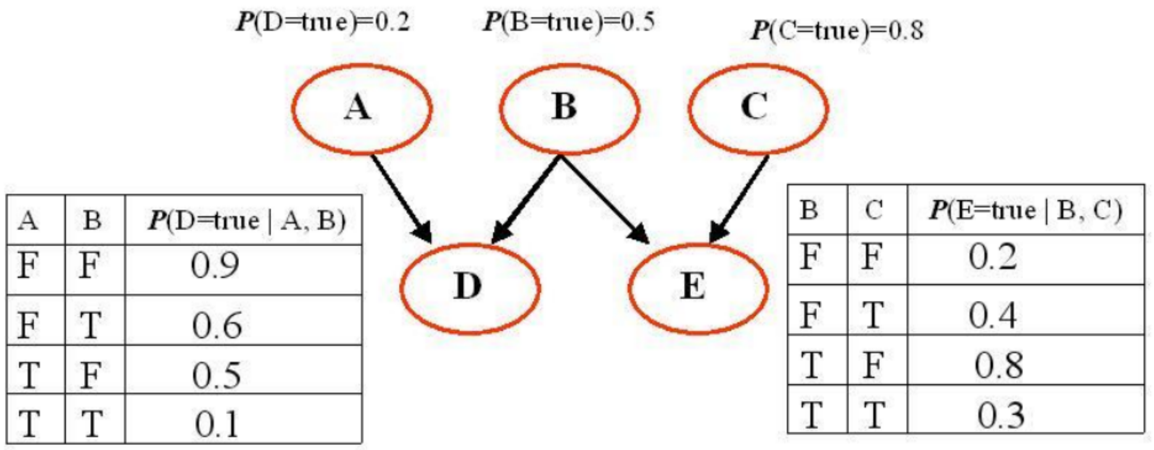
a) As a general search problem. Choose an appropriate algorithm. Is it better to fill in blanks one letter or one word at a time?

b) As a constraint satisfaction problem. Should the variables be words or letters?

Which formulation do you think will be better? Why?

(5 points)

Problem 6: Consider the following Bayesian network, where variables A through E are all Boolean valued:



A	B	P(D=true | A, B)
F	F	0.9
F	T	0.6
T	F	0.5
T	T	0.1

B	C	P(E=true | B, C)
F	F	0.2
F	T	0.4
T	F	0.8
T	T	0.3

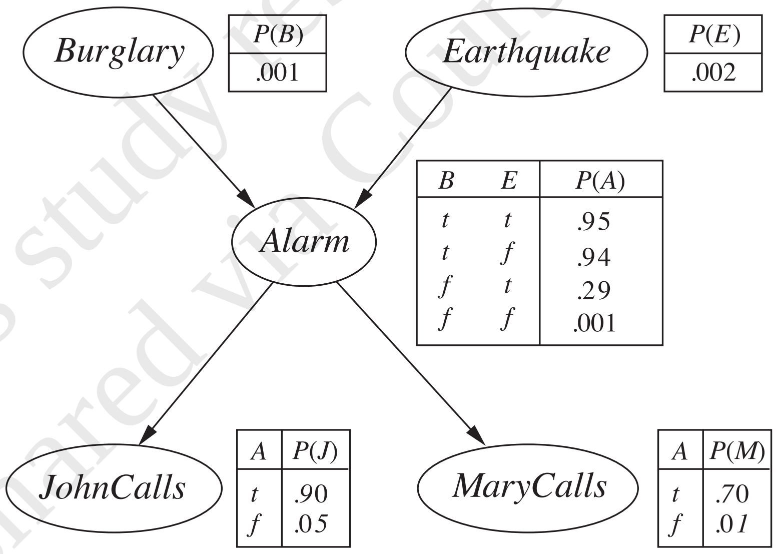
a) What is the probability that all five of these Boolean variables are simultaneously true?
[Hint: You have to compute the joint probability distribution. The structure of the Bayesian network suggests how the joint probability distribution is decomposed to the conditional probabilities available.]

b) What is the probability that all five of these Boolean variables are simultaneously false?
[Hint: Answer similarly to above.]

c) What is the probability that A is false given that the four other variables are all known to be true?

(15 points)

Problem 7: For this problem, check the Variable Elimination algorithm in your book. Also consider the Bayesian network from the "burglary" example.



P(B)
.001

P(E)
.002

B	E	P(A)
t	t	.95
t	f	.94
f	t	.29
f	f	.001

A	P(J)
t	.90
f	.05

A	P(M)
t	.70
f	.01

a) Apply variable elimination to the query:

$$P(Burglary|JohnsCalls = true, MaryCalls = true)$$

and show in detail the calculations that take place. Use your book to confirm that your answer is correct.

b) Count the number of arithmetic operations performed (additions, multiplications, divisions), and compare it against the number of operations performed by the tree enumeration algorithm.

c) Suppose a Bayesian network has the from of a *chain*: a sequence of Boolean variables $X_1, \ldots X_n$ where $Parents(X_i) = \{X_{i-1}\}$ for $i = 2, \ldots, n$. What is the complexity of computing $P(X_1|X_n = true)$ using enumeration? What is the complexity with variable elimination?

(15 points)

Problem 8: One method for approximate inference in Bayesian Networks is the Markov Chain Monte Carlo (MCMC) approach. This method depends on the important property that a variable in a Bayesian network is independent from any other variable in the network given its Markov Blanket.

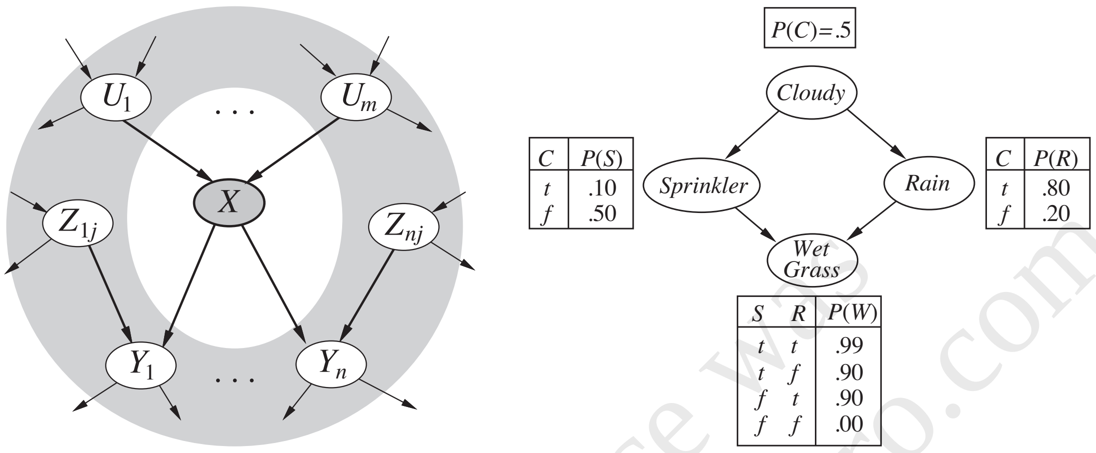


Figure 2: (left) The Markov Blanket of variable X (right) The Rain/Sprinkler network.

a) Prove that

$$P(X|MB(X)) = \alpha\, P(X|Parents(X)) \prod_{Y_i} P(Y_i|Parents(Y_i))$$

where $MB(X)$ is the Markov Blanket of variable X.

b) Consider the query

$$P(Rain|Sprinkler = true, WetGrass = true)$$

in the Rain/Sprinkler network and how MCMC would answer it. How many possible states are there for the approach to consider given the network and the available evidence variables?

c) Calculate the transition matrix Q that stores the probabilities $P(y \rightarrow y')$ for all the states y, y'. If the Markov Chain has n states, then the transition matrix has size $n \times n$ and you should compute n^2 probabilities.
[Hint: At each step, an unbiased coin $(0.5, 0.5)$ is tossed in order to select a variable to sample. Entries on the diagonal of the matrix correspond to self-loops, i.e., remaining in the same state. Such transitions can occur by sampling either variable. Entries where one variable is different between y and y', can occur only by sampling that one variable. Entries where two or more variables change cannot occur, since in MCMC only one variable is allowed to change at each transition.]

(15 points)