Assignment 3

Classical Planning - Probabilistic Reasoning - Bayesian Networks and Exact Inference - Approximate Inference - Value of Information

Deadline: November 9, 11:59pm.
Available points for undergraduate students: 110. Perfect score: 100.
Available points for graduate students: 105. Perfect score: 100.

Assignment Instructions:

Teams: Assignments should be completed by teams of students - three members for undergraduates and two for graduate students. Mixed teams between undergraduates and graduates are discouraged. If formed, they will be graded as graduate teams. No additional credit will be given for students that complete an assignment with fewer members than the recommended. **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.pracsyslab.org/cs440`).

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 hard-copies 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). Resources on how to use LaTeX are available on the course's website. 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 hard-copies. 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.

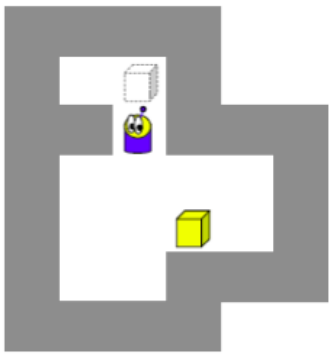
Precision: Try to be precise. Have in mind that you are trying to convince a very skeptical reader (and computer scientists are the worst kind...) that your answers are correct.

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 course, the department or the university (the standards are available through the course's website: `http://www.pracsyslab.org/cs440`). Failure to follow these rules may result in failure in the course.

Question 1: During the times of Pong, Pac-Man and Tetris, Hiroyuki Imabayashi created a complex game that tested the human abilities of planning:

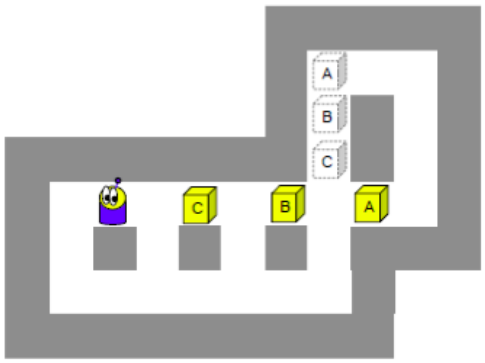
Sokoban - `http://en.wikipedia.org/wiki/Sokoban`.

Many folks are still addicted to solving Sokoban puzzles and you can join them by playing any of the versions freely distributed on the web. The goal is for the human, or robot, to push all the boxes into the desired locations. The robot can move horizontally and vertically and push one box at a time.

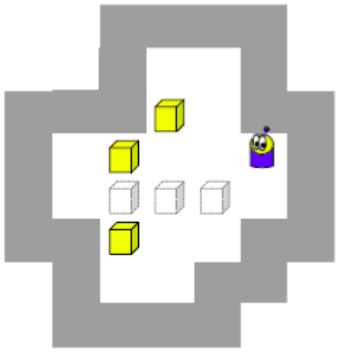


(a) Problem 2.1

(b) Problem 2.2



(c) Problem 2.3



(d) Sokoban Challenge

Your objective is to encode the above different configurations of Sokoban using the Planning domain definition language (PDDL), which we have covered in lectures. Then use the off-the-shelf planner FF to find solution plans and describe its performance.

1. For each of the problems in the above Figure, define the problem in PDDL (recommended version: PDDL2.1 level 1 (STRIPS)). You can either use the target lettering given in the picture or let the planner move any box to any target square. For the challenge problem, any box in any location is a solution. In the challenge, PDDL should NOT inform the robot which box should go to which location. In addition, you may also try other problems you invent or find on the web.

   To see example PDDL descriptions for the game of Iceblox, refer to the paper:
   `http://www.cs.rutgers.edu/~kb572/teaching/pddl_iceblox.pdf`

Additional examples of sample PDDL (including for the Wumpus World!) are available at: http://www.cs.rochester.edu/~kautz/satplan/blackbox/patrik-haslum-writing-pddl.html.

You may use the VAL code from the International Planning Competition (version 3.1 should be sufficient - select VAL on the left side menu):
http://www.inf.kcl.ac.uk/research/groups/PLANNING
The validator can be used to check if your plans are correct. The parser code can parse the PDDL domain and problem.

The following resources may also be helpful in understanding how to formulate PDDL domains:

- "An Introduction to PDDL" by Malte Helmert:
  http://www.cs.toronto.edu/~sheila/2542/f10/A1/introtopddl2.pdf
- "PDDL2.1: An Extension to PDDL for Expressing Temporal Planning Domains" by Fox, M. and Long, D. (2003). Journal of AI Research. The most relevant part of this paper is the appendix, which provides a BNF Specification of PDDL2.1.:
  http://www.jair.org/media/1129/live-1129-2132-jair.pdf.

You are also welcome to choose from pre-defined puzzles available at
http://webdocs.cs.ualberta.ca/~games/Sokoban/status.html.

2. Employ an existing planner to generate solutions for your test problem over a range of sizes to show the impact on performance as the problem increases in complexity. The FF planner is recommended for the solution generation process:
   http://fai.cs.uni-saarland.de/hoffmann/ff.html
   The FF planning algorithm grounds all action templates by instantiating for all possible combinations of matching values, and therefore, can be memory intensive. If you have memory issues, reduce your problem size.

   List the following elements of your solution:

   - Operators, predicates, objects.
   - The test problems (initial state, goal state).
   - A plan generated by the planner.
   - Analyze the plan and comment on its quality.
   - Time to arrive at a plan for the different problems.

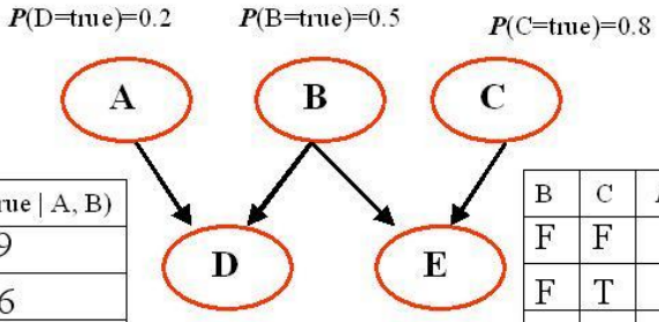
3. Is PDDL appropriate for describing such challenges? Discuss the complications in expressing geometric constraints in semantic/classical planning.

(40 points for undergraduates and 35 points for graduates)

Question 2: Consider the following Bayesian network, where variables *A* through *E* are all Boolean valued:

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



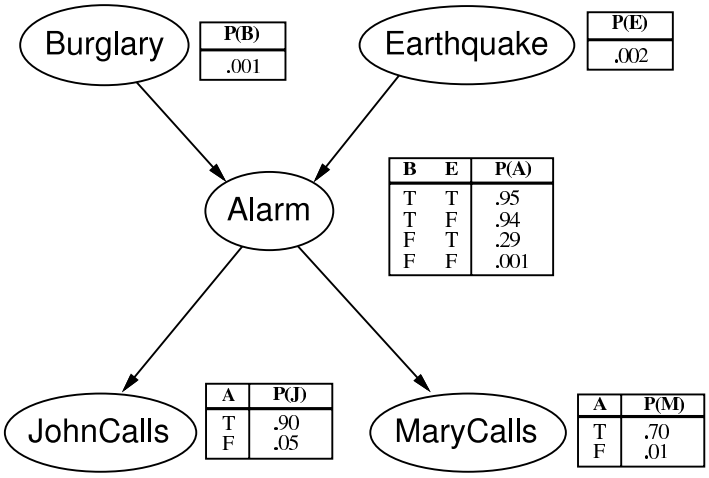
A	B	P(D=true | A, B)
F	F	0.9
F	T	0.6
T	F	0.5
T	T	0.1

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

c) What is the probability that A is false given that the four other variables are all known to be true?
[Hint: Try to use the definition of the conditional probability and the structure of the network. For probabilities that can not be computed directly from the network, remember the following normalization trick: if $P(x) = \alpha \cdot f(x)$ and $P(\neg x) = \alpha \cdot f(\neg x)$, then you can compute the normalization factor as $\alpha = \frac{1.0}{f(x)+f(\neg x)}$, since $P(x) + P(\neg x) = 1.0$.]

(15 points)

Question 3: For this problem, check the Variable Elimination algorithm in your book. Also consider the Bayesian network from the "burglary" example.



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)

Question 4: 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.

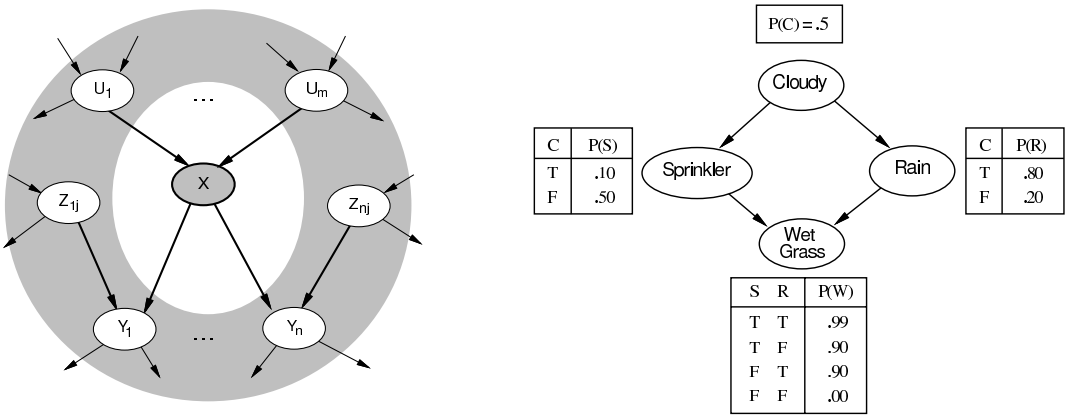


Figure 1: (left) The Markov Blanket of variable X (right) The Rain/Sprinkler network.

a) Prove that

$$P(X|MB(X)) = \alpha\, P(X|U_1, \ldots, U_m) \prod_{Y_i} P(Y_i|Z_{i1} \ldots)$$

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: 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', must sample 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)

Question 5: Assume you are interested in buying a used vehicle C_1. You are also considering of taking it to a qualified mechanic and then decide whether to buy it or not. The cost of taking it to the mechanic is \$100. C_1 can be in good shape (quality q^+) or bad one (quality q^-). The mechanic might help to indicate what shape the vehicle is in. C_1 costs \$3,000 to buy and its market value is \$4,000 if in good shape; if not, \$1,400 in repairs will be needed to make it in good shape. Your estimate is that C_1 has a 70% chance of being in good shape. Assume that the utility function depends linearly on the vehicle's monetary value.

a. Calculate the expected net gain from buying C_1, given no test.

b. We also have the following information about whether the vehicle will pass the mechanic's test:

$$P(pass(c_1)|q^+(c_1)) = 0.8$$

$$P(pass(c_1)|q^-(c_1)) = 0.35$$

Use Bayes' theorem to calculate the probability that the car will pass/fail the test and hence the probability that it is in good/ bad shape given what the mechanic will tell you.
[Hint: Compute the four probabilities: $P(q^+|Pass)$, $P(q^-|Pass)$, $P(q^+|\neg Pass)$, $P(q^-|\neg Pass)$]

c. What is the best decision given either a pass or a fail? What is the expected utility in each case?
[Hint: Use the probabilities from the previous question.]

d. What is the value of optimal information for the mechanic's test? Will you take C_1 to the mechanic or not?
[Hint: You can easily answer this based on the answers from questions a) and c).]

(25 points)