CS 520: Introduction to Artificial Intelligence **(Due: 02/29/20)**

Assignment #2: Search Problems in AI - Probabilistic Reasoning

Instructor: Abdeslam Boularias
TA: Aravind Sivaramakrishnan

Name: Laxman Pottimuthi, *Netid:* lp642
Name: Sreeram Madineni, *Netid:* sm2323

Problem 1

Trace the operation of A* search (the tree version) applied to the problem of getting to Bucharest from Lugoj using the straight-line distance heuristic. That is, show the sequence of nodes that the algorithm will consider and the f, g, and h score for each node. You don't need to draw the graph, just right down a sequence of (city, f(city), g(city), h(city)) in the order in which the nodes are expanded.

- Consider straight line distance as h, cost from source as g and cost as f.
- Below is the sequential order in which nodes are expanded

	f	g	h
Lugoj	244	0	244
Mehadia	311	70	241
Timisoara	440	111	329
Drobeta	387	145	242
Craiova	425	265	160
Pitesti	503	403	100
Rimnicu Vilcea	604	411	193
Arad	595	229	366
Bucharest	504	504	0

Problem 2

Consider a state space where the start state is number 1 and each state k has two successors: numbers 2k and 2k + 1.

(a) Suppose the goal state is 11. List the order in which states will be visited for breadth first search, depth-limited search with limit 3, and iterative deepening search.

- States listed in a list from 1.

Algo	Order in which nodes are visited
Breadth first search	1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11
Depth First Search with limit 3	1, 2, 4, 8, 9, 5, 10, 11
Iterative deepening	1 1, 2, 3 1, 2, 4, 5, 3, 6, 7 1, 2, 4, 8, 9, 5, 10, 11

(b) How well would bidirectional search work on this problem? List the order in which states will be visited. What is the branching factor in each direction of the bidirectional search?

- In bidirectional graph, expansion from the goal nodes towards source are computed by $\lfloor g/2 \rfloor$.
- From start node, nodes expanded by BFS are nodes 2,3.
- Nodes visited from goal state is $\lfloor \frac{11}{2} \rfloor = 5$.
- In second step, nodes visited by tree from start state is 4,5.
- Here, a node in forward tree = node in backward tree and algorithm stops.
- Also, branching factor from forward is 2 and from backward is 1. Small branching factor in this example is useful to calculate the path efficiently.

Problem 3

Which of the following statements are correct and which ones are wrong?

(a) Breadth-first search is a special case of uniform-cost search.

Yes. If all path costs are equal, BFS is special case of uniform cost search.

(b) Depth-first search is a special case of best-first tree search.

Yes. With f(n) = -depth(n), depth first search is special case of best-first search.

(c) Uniform-cost search is a special case of A* search.

Yes. When h(n) = 0 in A*, it will be uniform cost search.

(d) Depth-first graph search is guaranteed to return an optimal solution.

No. It doesn't guarantee optimal solution.

(e) Breadth-first graph search is guaranteed to return an optimal solution.

No. It will guarantee optimal solution only when all the costs of edges are equal.

(f) Uniform-cost graph search is guaranteed to return an optimal solution.

Yes.It always chooses optimal node after expansion.

(g) A* graph search is guaranteed to return an optimal solution if the heuristic is consistent.

Yes. As heuristic is consistent A* gives optimal solution.

(h) A* graph search is guaranteed to expand no more nodes than depth-first graph search if the heuristic is consistent.

False. A* is not guaranteed to expand no more nodes than DFS. Counter example is DFS can sometimes give sub-optimal solution and better than A*

(i) A* graph search is guaranteed to expand no more nodes than uniform-cost graph search if the heuristic is consistent.

True. The additional heuristic term in A* allows A* to expand more nodes than uniform cost search.

Problem 4

Iterative deepening is sometimes used as an alternative to breadth first search. Give one advantage of iterative deepening over BFS, and give one disadvantage of iterative deepening as compared with BFS. Be concise and specific.

- Advantage - Iterative deepening has space complexity of O(bd) but BFS has space complexity of O(b^d). b is maximum branching factor and d is depth of shallowest goal node.
- Disadvantage - Iterative deepening takes more time compared to BFS. This is because if number of iterations are high, it has to recompute the nodes which was already computed in previous iterations.

Problem 5

Prove that if a heuristic is consistent, it must be admissible. Construct an example of an admissible heuristic that is not consistent. (Hint: you can draw a small graph of 3 nodes and write arbitrary cost and heuristic values so that the heuristic is admissible but not consistent)

- Let h(n) be a consistent heuristic. Let n_p and n_{p+1} be two nodes where n_{p+1} is successor node of n_p. We have h(n_p) <= c(n_p,n_{p+1}) + h(n_{p+1}). c(n_p,n_{p+1}) is the cost to travel from n_p to n_{p+1}
- Assume 'a' is the shortest distance from n_p to goal state. Assume 'b' is the shortest distance from n_{p+1} to goal state
- Let h(n) be admissible for n_{p+1}. If we prove that it is admissible for n_p also then we can say by induction that h(n) is admissible.

$$h(n_{p+1}) \leq b$$
$$h(n_p) \leq c(n_p, n_{p+1}) + h(n_{p+1})$$
$$h(n_p) \leq c(n_p, n_{p+1}) + b$$

- We have $a = b + c(n_p, n_{p+1})$ as n_{p+1} is the successor node of n_p.
- Thus we get $h(n_p) \leq a$.
- Hence by induction we can say that h(n) is admissible
- Example of an admissible heuristic that is not consistent.

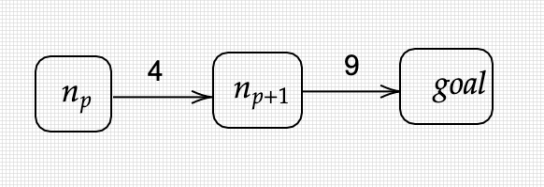

Figure 1: Admissible heuristic which is not consistent

- $h(n_{p+1}) = 6, h(n_p) = 12$
- Distance from n_p to goal = 4 + 9 = 13. Distance from n_{p+1} to goal = 9 $\geq h(n_{p+1})$
- $h(n_p) > h(n_{p+1}) + c(n_p, n_{p+1})$ (i.e 12 > 6+4)
- Thus, $h(n_p) \leq h(n_{p+1}) + c(n_p, n_{p+1})$ is false. So h(n) is not consistent

Problem 6

In a Constraint Satisfaction Problem (CSP) search, explain why it is a good heuristic to choose the variable that is most constrained but the value that is least constraining

- In CSP we are looking to find an assignment of values to variables that satisfies all the constraints. We need to find this assignment as quickly as possible or we must be able to say that we cannot find such an assignment. So we must choose the most constrained variable as it is more likely to fail. For that variable we need to assign the least constrained value. This would allow us to have more flexibility in assigning values to neighbouring variables. This would mean there is a better chance that we can find the solution quickly as we will explore more values for neighbouring variables before we explore other values of current variable.

Problem 7

Consider the following game tree, where the first move is made by the MAX player and the second move is made by the MIN player.
(a) What is the best move for the MAX player using the minimax procedure?

- Best move for A is C. Below graph is populated according to minmax procedure.

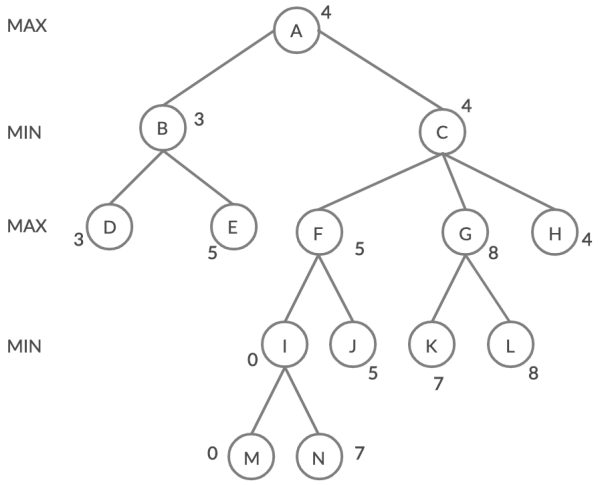

Figure 2: minmax tree

(b) Perform a left-to-right (left branch first, then right branch) alpha-beta pruning on the tree. That is, draw only the parts of the tree that are visited and don't draw branches that are cut off

- Alpha-beta pruning done on the tree from left to right using utilities as follows:

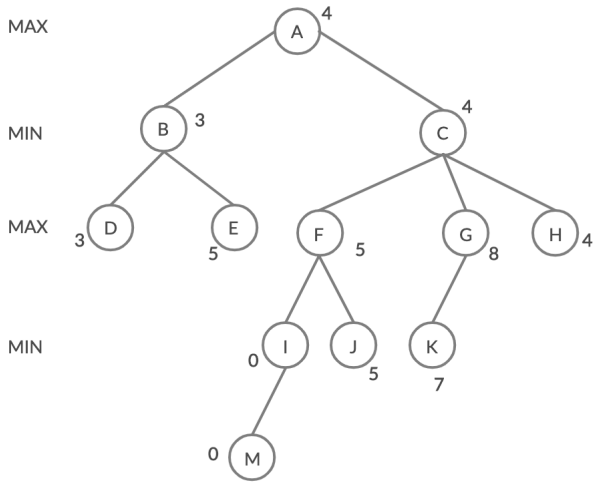

Figure 3: left to right alpha - beta pruning

(c) Do the same thing as in the previous question, but with a right-to-left ordering of the actions. Discuss why different pruning occurs.

- alpha-beta pruning done on the tree from right to left using utilities as follows:

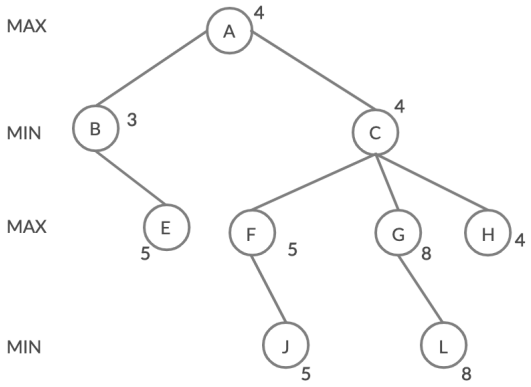

Figure 4: right to left alpha - beta pruning

- There is difference in pruning of the nodes when direction of the pruning is changed.
- This is because if α, β of the parent node satisfies the conditions of alpha-beta pruning such as $\alpha \leq \beta$, the next nodes for that parent are pruned in a order of successor nodes being visited.
- If such leaf nodes are visited first(either in either left-right or right-left pruning), the successor nodes are pruned. Therefore, the order of pruning is important

Problem 8

Which of the following are admissible, given admissible heuristics h1, h2? Which of the following are consistent, given consistent heuristics h1, h2? Justify your answer.Which one of these three new heuristics (a, b or c) would you prefer to use for A*? Justify your answer.

(a) $h(n) = min\{h1(n); h2(n)\}$

- Consider a node n_1 and its successor n_2. $h_1(n)$ is consistent, then we get
  $h_1(n_1) <= c(n_1, n_2) + h_1(n_2)$
  $h_2(n_1) <= c(n_1, n_2) + h_2(n_2)$
  Let us say $h_1(n_1)$ be the minimum at n_1 and $h_2(n_2)$ be the minimum at n_2
  We cannot say $h_1(n_1) <= c(n_1, n_2) + h_2(n_2)$ as $h_1(n_2)$ is greater than $h_2(n_2)$. So we cannot say h(n) is consistent
- As h(n) will always be one of $h_1(n_1)$ and $h_2(n_2)$, if both $h_1(n_1)$ and $h_2(n_2)$ are admissible then h(n) will also be admissible

(b) $h(n) = wh1(n) + (1 - w)h2(n)$; **where** $0 \leq w \leq 1$

- $h(n)$ = w$h_1(n)$ + (1 - w)$h_2(n)$ Consider a node n_1 and its successor n_2 we have
  $h_1(n_1) <= c(n_1, n_2) + h_1(n_2)$
  $h_2(n_1) <= c(n_1, n_2) + h_2(n_2)$
  Multiplying the first equation by w and the second equation by 1-w we get
  $w * h_1(n_1) <= w * c(n_1, n_2) + w * h_1(n_2)$
  (1-w)*$h_2(n_1)$ <=(1-w)* $c(n_1, n_2)$ +(1-w)* $h_2(n_2)$
  Adding the above two equations we get $w * h_1(n_1) + (1 - w) * h_2(n_1) <= c(n_1, n_2) + w * h_1(n_2) + (1 - w) * h_2(n_2)$. Thus the given heuristic is consistent
- h(n) = w$h_1(n)$ + (1 - w)$h_2(n)$ .Let us take a node n_1 and let 'a' be the shortest distance from n_1 to the goal
  Since $h_1(n)$ is admissible we get $h_1(n)$ <= a and $h_2(n)$ <= a. Multiplying both equations by 'w' and '1 - w' we get
  w$h_1(n_1)$ <= w*a and $(1 - w)h_{2(n_1)}$ <= (1-w)*a
  Adding the above two equations we get $w * h_1(n_1) + (1 - w) * h_2(n_1) <= a$
  Thus h(n) is admissible

(c) $max\{h1(n); h2(n)\}$

- Consider a node n_1 and its successor n_2. $h_1(n)$ is consistent, then we get
  $h_1(n_1) <= c(n_1, n_2) + h_1(n_2)$
  $h_2(n_1) <= c(n_1, n_2) + h_2(n_2)$
  Let us say $h_1(n_1)$ be the maximum at n_1 and $h_2(n_2)$ be the maximum at n_2
  We can say $h_1(n_1) <= c(n_1, n_2) + h_2(n_2)$ since $h_2(n_2) >= h_1(n_2)$. So max($h_1(n_1)$,$h_2(n_2)$) <= $c(n_1, n_2)$ + max($h_1(n_1)$,$h_2(n_2)$)
  Thus h(n) = max($h_1(n)$,$h_2(n)$) is consistent
- As h(n) will always be one of $h_1(n_1)$ and $h_2(n_2)$, If both $h_1(n_1)$ and $h_2(n_2)$ are admissible then h(n) will also be admissible

We prefer heuristics that will make the A* algorithm to expand as few nodes as possible. For that to happen we need to use the heuristic that is greater than all other heuristics at every node n. So we can choose the heuristic max($h_1(n)$,$h_2(n)$) that will always produce the maximum value at every node n. So we choose option 'c'

Problem 9

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?

- Randomness is not required when there is no local optima/plateu/ridges or even when single global minima.

(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?

- Hill climbing part is required when we know there is global maximum and we need to achieve it using randomness. If such structure is not available then we do not need hill climbing part.

(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?

- Simulated annealing is a useful technique when the cost function curve has troubles of local minima, ridges and plateaus with global maxima. From the explanation, shape of value function should not be convex. If it is the case, hill climbing itself should solve the problem.

(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?

- Store the best solution till now of the annealing process at each stage.
- Output the result when an equilibrium state is achieved.

(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.]

- Given, we store all the 2 million states at once. Currently for the algorithm we are using the space of storing only few states. Initially steps of initialising the search is done in similarly. Initialise the search with high T.
- Move randomly for the fixed distance and using the objective function of simulated annealing, compare those states with the current state. Use same formula like simulated annealing to move to next state. In this step, store the information of the states from which the movement has been done and Also store the best solution reached from this state. In this way, relation between the states is maintained. As the relation is maintained between the states, while the T is lowered gradually, and if a state is reached by random restart, we can trace back to the states that has been visited and find the best solution which is already stored in the states. By doing this, while visiting that state a computation is not needed and a random restart is not required again. That is why consecutive random restarts of million times will be not required.

(f) Gradient ascent search is prone to local optima just like hill climbing. Describe how you might adapt randomness in simulated annealing to gradient ascent search avoid trap of local maximum.

- Gradient search is prone to local maxima when there are flat regions.
- In this state while executing the gradient ascent step the algorithm is stuck indefinitely.
- A noise/randomness is added to each gradient ascent step. It will help gain robustness to such regions and will help getting out from the stuck part.

Problem 10

Consider the two-player game described:

(1) Draw the complete game tree, using the following conventions

- Each state is represented by the value at the end of the process.
- Terminal state is given a square box and loop state with double box and a ? for those states.
- Below is the game tree using the notations given in the question.

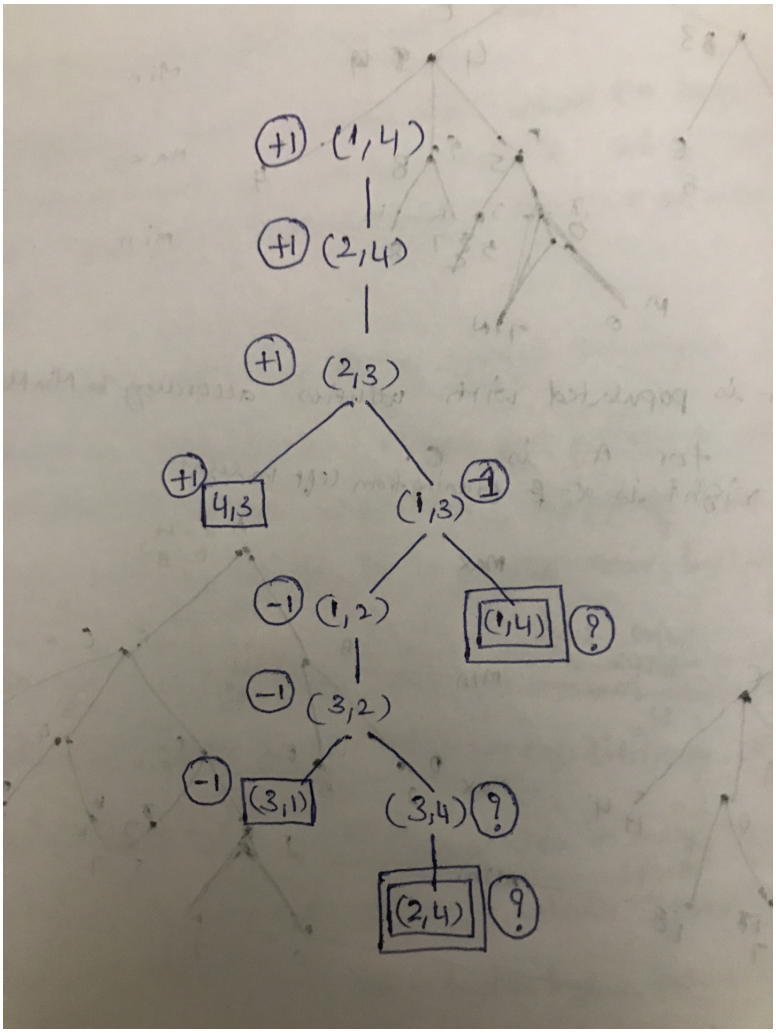

Figure 5: game tree

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

- Given that the loop states are given ? since it is not clear to handle.
- Now when a state has value of ? and if its successors states also have the same value, the all the backedup nodes should also be given the value as ?.
- While visiting states, if there is a possibility of choosing a ? or a state with value $+1/-1$, player chooses the state which does not choose ? and chooses either $+1/-1$ to win the game. For example, if there is a possibility of choosing +1 or ?, A will choose +1 to win the game and if ther is possibility of choosing -1 or ?, B will choose -1 to win the game.

Problem 11

Consider the following Bayesian network, where variables A through E are 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. The structure of the Bayesian network suggests how the joint probability distribution is decomposed to the conditional probabilities available.]

$$P(A=T,B=T,C=T,D=T,E=T) = P(\frac{D=T}{A=T,B=T})P(\frac{E=T}{B=T,C=T})P(A=T)P(B=T)P(C=T)$$

$$= 0.1 \times 0.3 \times 0.2 \times 0.5 \times 0.8$$

$$= 0.0024$$

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

$$P(A=F,B=F,C=F,D=F,E=F) = P(\frac{D=F}{A=F,B=F})P(\frac{E=F}{B=F,C=F})P(A=F)P(B=F)P(C=F)$$

$$= 0.1 \times 0.8 \times 0.8 \times 0.5 \times 0.2$$

$$= 0.0064$$

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

- Probability that A is false given that the four other variables are all known to be true

$$P(\frac{A=F}{B=T,C=T,D=T,E=T}) = \alpha P(A=F,B=T,C=T,D=T,E=T)$$

$$= \alpha P(\frac{D=T}{A=F,B=T})P(\frac{E=T}{B=T,C=T})P(A=F)P(B=T)P(C=T)$$

$$= \alpha \times 0.6 \times 0.3 \times 0.8 \times 0.5 \times 0.8$$

$$= \alpha 0.0576 \tag{11.1}$$

$$P(\frac{A=T}{B=T,C=T,D=T,E=T}) = \alpha P(A=T,B=T,C=T,D=T,E=T)$$

$$= \alpha 0.1 \times 0.3 \times 0.2 \times 0.5 \times 0.8$$

$$= \alpha 0.0024 \tag{11.2}$$

- From $11.1, 11.2$, we get $11.1 + 11.2 = 1$

$$\implies \alpha = \frac{1}{0.0576 + 0.0024} = \frac{1}{0.06} = 16.66 \tag{0.1}$$

$$\therefore P(\frac{A=F}{B=T,C=T,D=T,E=T}) = 16.66 \times 0.0576 = 0.96$$

Problem 12

Solve the following.

(a) Calculate $P(Burglary|JohnsCalls = true, MaryCalls = true)$ and show in detail the calculations that take place. Use your book to confirm that your answer is correct

$$
\begin{aligned}
&P(Burglary = true|JohnsCalls = true, MaryCalls = true)\\
&= \alpha \sum_a \sum_e P(B = T, J = T, M = T, E, A)\\
&= \alpha \sum_a \sum_e P(\frac{J = T}{A})P(\frac{M = T}{A})P(\frac{A}{B = T, E})P(B = T)P(E)\\
&= \alpha P(B = T) \sum_e P(E) \sum_a P(\frac{J = T}{A})P(\frac{M = T}{A})P(\frac{A}{B = T, E})\\
&= \alpha P(B = T) \sum_e P(E)(P(\frac{J = T}{A = T})P(\frac{M = T}{A = T})P(\frac{A = T}{B = T, E}) + P(\frac{J = T}{A = F})P(\frac{M = T}{A = F})P(\frac{A = T}{B = T, E}))\\
&= \alpha P(B = T) \sum_e P(E)(0.90 * 0.70 * P(\frac{A = T}{B = T, E}) + 0.05 * 0.01 * P(\frac{A = F}{B = T, E}))\\
&= \alpha P(B = T)(P(E = T) * 0.90 * 0.70 * P(\frac{A = T}{B = T, E = T}) + P(E = F) * 0.90 * 0.70 * P(\frac{A = T}{B = T, E = F})\\
&+ P(E = T) * 0.05 * 0.01 * P(\frac{A = F}{B = T, E = T}) + P(E = F) * 0.05 * 0.01 * P(\frac{A = F}{B = T, E = F}))\\
&= \alpha P(B = T)(0.002 * 0.90 * 0.70 * 0.95 + 0.998 * 0.90 * 0.70 * 0.94\\
&+ 0.002 * 0.05 * 0.01 * 0.05 + 0.998 * 0.05 * 0.01 * 0.06)\\
&= \alpha P(B = T) * (0.5922)\\
&= \alpha(0.001) * (0.5922)\\
&= \alpha 0.0005922
\end{aligned}
$$

$$\therefore P(Burglary = true|JohnsCalls = true, MaryCalls = true) = \alpha 0.0005922 \tag{0.2}$$

Also,

$$
\begin{aligned}
&P(Burglary = false|JohnsCalls = true, MaryCalls = true)\\
&= \alpha \sum_a \sum_e P(B = F, J = T, M = T, E, A)\\
&= \alpha P(B = F) \sum_e P(E) \sum_a P(\frac{J = T}{A})P(\frac{M = T}{A})P(\frac{A}{B = F, E})\\
&= \alpha(0.999) \sum_e P(E)(P(\frac{J = T}{A = T})P(\frac{M = T}{A = T})P(\frac{A = T}{B = F, E}) + P(\frac{J = T}{A = F})P(\frac{M = T}{A = F})P(\frac{A = T}{B = F, E}))\\
&= \alpha 0.999 \sum_e P(E)(0.90 * 0.70 * P(\frac{A = T}{B = F, E}) + 0.05 * 0.01 * P(\frac{A = F}{B = F, E}))\\
&= \alpha 0.999(0.000354 + 0.00000029 + 0.00062874 + 0.000498501)\\
&= \alpha 0.999 * 0.0014929\\
&= \alpha(0.0014914)\\
&\therefore P(Burglary = false|JohnsCalls = true, MaryCalls = true) = \alpha(0.0014914)
\end{aligned}
\tag{0.3}
$$

From equation 0.2 and 0.3, $0.2 + 0.3 = 1$

$$\alpha = \frac{1}{0.0005922 + 0.0001491} = 480.03$$
$$\therefore P(Burglary = true/JohnsCalls = true, MaryCalls = true) = 480.03 * 0.0005922 = 0.2842$$
$$\therefore P(Burglary = false/JohnsCalls = true, MaryCalls = true) = 480.03 * 0.0014914 = 0.7158$$

(b) Suppose a Bayesian network has the from of a chain: a sequence of Boolean variables X1,. . . Xn where Parents$(X_i) = \{X_{i-1}\}$ for i = 2,. . .n. What is the complexity of computing $P(X1|Xn = true)$ using enumeration? What is the complexity with variable elimination

$$\begin{aligned}
&P(X_1|X_n = true) \\
&= \alpha \textstyle\sum_{X_2 \ldots X_{n-1}} P(X_n = true|X_{n-1})P(X_{n-1}|X_{n-2})...P(X_2|X_1)P(X_1) \\
&= \alpha P(X_1) \textstyle\sum_{X_2 \ldots X_{n-1}} P(X_n = true|X_{n-1})P(X_{n-1}|X_{n-2})...P(X_2|X_1) \\
&= \alpha P(X_1) \textstyle\sum_{X_2} \sum_{X_3} ... \sum_{X_{n-1}} P(X_n = true|X_{n-1})P(X_{n-1}|X_{n-2})...P(X_2|X_1)
\end{aligned} \tag{0.4}$$

There are (n - 2) summations and each term has (n - 1) terms in it. So the total number of computations is

$$\begin{aligned}
&O((n-2) \times 2^{n-2}) \\
&= O(n2^{n-2} - 2^{n-1}) \\
&= O(n2^{n-2}) \\
&= O(n2^n/4) \\
&= O(n2^n)
\end{aligned} \tag{0.5}$$

In variable elimination algorithm we sum out each variable from right to left and keep producing new factors. In the below equation we will get two terms for each summation. So for (n - 2) summations we get (n - 2)2 terms. So the runtime is $O((n-2)2) = O(n)$

$$\therefore P(X_1|X_n = true) = \alpha \sum_{X_{n-1}} P(X_n = true|X_{n-1}) \sum_{X_{n-2}} P(X_{n-1}|X_{n-2}) ... \sum_{X_3} P(X_4|X_3) \sum_{X_2} P(X_3|X_2) \tag{0.6}$$

Problem 13

(a) Construct a Bayes Network to identify fraudulent transactions.

- Bayes Network to identify fraudulent transactions

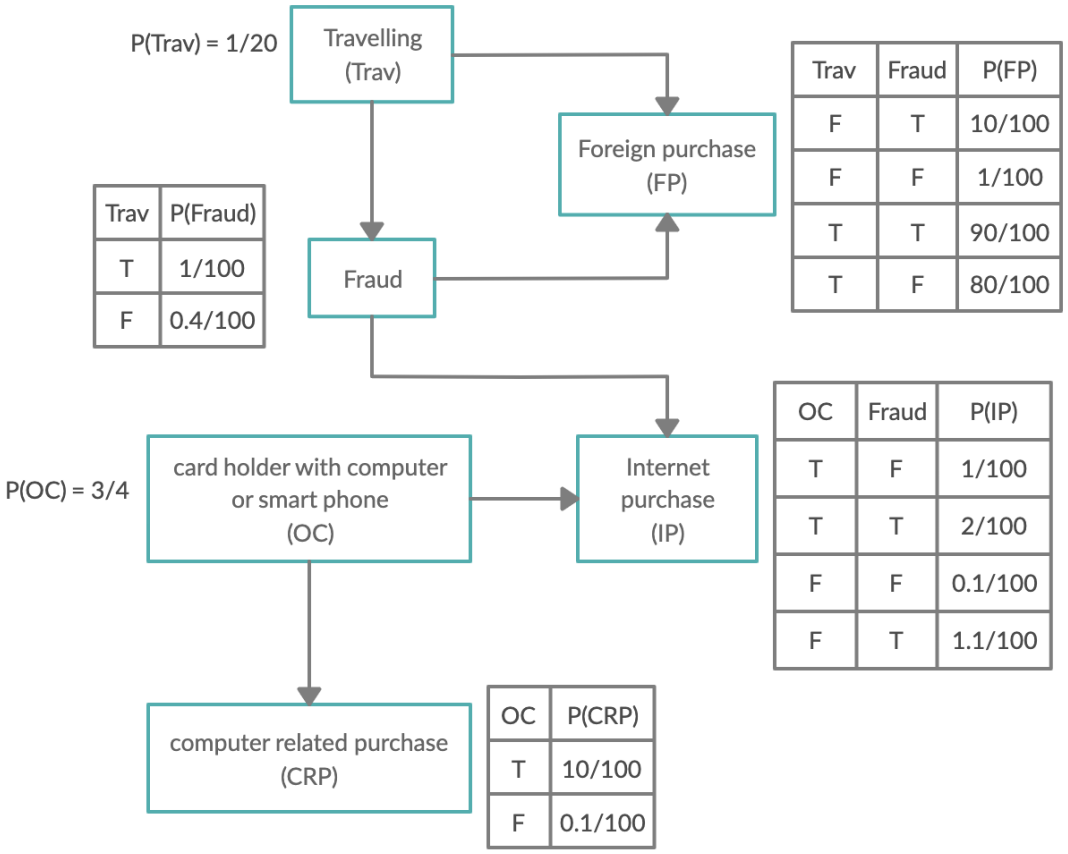

Figure 6: Bayes Network to identify fraudulent transactions

(b) What is the prior probability (i.e., before we search for previous computer related purchases and before we verify whether it is a foreign and/or an internet purchase) that the current transaction is a fraud? What is the probability that the current transaction is a fraud once we have verified that it is a foreign transaction, but not an internet purchase and that the card holder purchased computer related accessories in the past week?

If no evidence at all is provided, then we can marginalize over the parents of the inquiry variable.

$$P(Fraud = True) = P(Fraud = T, Parent(Fraud) = T) + P(Fraud = T, Parent(Fraud) = F)$$
$$= \text{P(Fraud = True,Trav = True)} + \text{P(Fraud = True,Trav = False)}$$
$$= \text{P(Fraud=T} \mid Trav = T)P(Trav = T) + P(Fraud = T \mid Trav = F)P(Trav = F)$$
$$= (1/100)(0.4/100) + (1/20)(19/20) = 0.0043$$
$$\therefore P(Fraud = True) = 0.0043 \ (0.7)$$

Also for the second part,

$$
\begin{aligned}
&P(Fraud = T \mid FP = T, IP = F, CRP = T) \\
&= \alpha \sum_{Trav} \sum_{OC} P(Fraud = T, FP = T, IP = T, CRP = T, OC, Trav) \\
&= \alpha \sum_{Trav} \sum_{OC} P(CRP = T \mid OC) P(IP = F \mid OC, Fraud = T) P(FP = T \mid Fraud = T, Trav) P(Fraud = T \mid Trav) P(Trav) \\
&= \alpha \sum_{Trav} P(Trav) P(Fraud = T \mid Trav) P(FP = T \mid Fraud = T, Trav) \sum_{OC} P(CRP = T \mid OC) P(IP = F \mid OC, Fraud = T) \\
&= \alpha \sum_{Trav} P(Trav) P(Fraud = T \mid Trav) P(FP = T \mid Fraud = T, Trav)(0.0735 + 0.00024725) \\
&= \alpha (P(Trav = T) P(Fraud = F \mid Trav = T) P(FP = T \mid Fraud = F, Trav = T) \\
&+ P(Trav = F) P(Fraud = F \mid Trav = F) P(FP = T \mid Fraud = F, Trav = F)) * (0.0737) \\
&= \alpha (0.00045 + 0.00038)(0.0737) \\
&= \alpha 0.000061171 \\
&\therefore P(Fraud = T \mid FP = T, IP = F, CRP = T) = \alpha 0.000061171
\end{aligned}
\tag{0.8}
$$

$$
\begin{aligned}
&P(Fraud = F \mid FP = T, IP = F, CRP = T) \\
&= \alpha \sum_{Trav} \sum_{OC} P(Fraud = F, FP = T, IP = T, CRP = T, OC, Trav) \\
&= \alpha \sum_{Trav} \sum_{OC} P(CRP = T \mid OC) P(IP = F \mid OC, Fraud = F) P(FP = T \mid Fraud = F, Trav) P(Fraud = F \mid Trav) P(Trav) \\
&= \alpha \sum_{Trav} P(Trav) P(Fraud = F \mid Trav) P(FP = T \mid Fraud = F, Trav) \sum_{OC} P(CRP = T \mid OC) P(IP = F \mid OC, Fraud = F) \\
&= \alpha \sum_{Trav} P(Trav) P(Fraud = F \mid Trav) P(FP = T \mid Fraud = F, Trav)(0.07425 + 0.00024975) \\
&= \alpha (P(Trav = T) P(Fraud = T \mid Trav = T) P(FP = T \mid Fraud = T, Trav = T) + P(Trav = F) P(Fraud = T \mid Trav = F) \\
&= \alpha (0.04455 + 0.009462)(0.074499) \\
&= \alpha 0.00402 \\
&\therefore P(Fraud = F \mid FP = T, IP = F, CRP = T) = \alpha 0.00402
\end{aligned}
\tag{0.9}
$$

- From 0.8,0.9, we have $0.8 + 0.9 = 1$
  $\therefore \alpha = 1/(0.00402 + 0.000061171) = 245.02$

  $\therefore P(Fraud = T \mid FP = T, IP = F, CRP = T) = 245.02 * 0.000061171 = 0.0149$