Hansen T. Dharmawan | A91413023
Mark Darmadi | A91413023
Vania Christie Chandra | A91413023

1. $P(J = 1 \mid B = 0, E = 1)$

$= P(J = 1, A = 1 \mid B = 0, E = 1) + P\big((J = 1, A = 0 \mid B = 0, E = 1\big)$

$= P(J = 1 \mid A = 1, B = 0, E = 1) * P(A = 1 \mid B = 0, E = 1)$

$+ P(J = 1 \mid A = 0, B = 0, E = 1) * P(A = 0 \mid B = 0, E = 1)$

$= P(J = 1 \mid A = 1) * P(A = 1 \mid B = 0, E = 1) + P(J = 1 \mid A = 0) * P(A = 0 \mid B = 0, E = 1)$ → conditionally independent

$P(A = 1 \mid B = 0, E = 1) = 0.29$ → Look up CPT table

$P(A = 0 \mid B = 0, E = 1) = 1 - 0.29 = 0.71$

$P(J = 1 \mid A = 1) * P(A = 1 \mid B = 0, E = 1) + P(J = 1 \mid A = 0) * P(A = 0 \mid B = 0, E = 1)$

$= 0.90 * 0.29 + 0.05 * 0.71 = 0.2965$

Hansen T. Dharmawan | A91413023
Mark Darmadi | A91413023
Vania Christie Chandra | A91413023

2. $P(B=1|\,J=1) = P(J=1|B=1) * \frac{P(B=1)}{P(J=1)}$

$$P(J=1) = P(J=1, A=1) + P(J=1, A=0)$$
$$= P(J=1|A=1) * P(A=1) + P(J=1|A=0) * P(A=0)$$

$$P(A=1) = P(A=1|B=1, E=1) * P(B=1) * P(E=1) +$$

$$P(A=1|B=1, E=0) * P(B=1) * P(E=0) +$$

$$P(A=1|B=0, E=1) * P(B=0) * P(E=1) +$$

$P(A=1|B=0, E=0) * P(B=0) * P(E=0)$ →Look up CPT table

$$= 0.95 * 0.001 * 0.002 \;+\; 0.94 * 0.001 * (1-0.002) +\; 0.29 * (1-0.001) * 0.002 + 0.001 * (1-0.001) * (1-0.002)$$

$$= 0.00252$$

$$P(A=0) = 1 - 0.00252 = 0.9975$$

$$P(J=1) = 0.90 * 0.0025 + 0.05 * 0.9975 = 0.052$$

$$P(J=1|B=1) = P(J=1|A=1, B=1) * P(A=1|B=1) +$$

$$P(J=1|A=0, B=1) * P(A=0|B=1)$$

$$P(A=1|B=1) = P(A=1|B=1, E=1) * P(E=1) + P(A=1|B=1, E=0) * P(E=0)$$

$$= 0.95 * 0.002 + 0.94 * (1-0.002) = 0.94002$$

$$P(J=1|B=1) = P(J=1|A=1) * P(A=1|B=1) + P(J=1|A=0) * P(A=0|B=1)$$

$$= 0.90 * 0.94002 + 0.05 * (1-0.94002) = 0.8490$$

$$P(B=1|J=1) = \; P(J=1|B=1) * \frac{P(B=1)}{P(J=1)} = 0.8490 * \frac{0.001}{0.052} = 0.016$$

3.

a. **Rejection Sampling:**
Rejection sampling is an approximate inference algorithm in a Bayesian network. The idea is to sample the Bayesian network a large number of times and count only the samples that is consistent with the given evidence variables. The resulting count is then normalized to get the estimated probability
We implemented the Rejection sampling by first implementing the Prior sampling using random number generator to make random events. We

iterate through the variables in topological order, comparing the random number with the probability of that variable being True given its parents. If the random number is bigger, that variable is assigned to False, it is assigned True otherwise. From the event returned from Prior sampling, we check if all of the evidence variable's value is consistent with that event. If it is consistent, we check the value of the query variable in that event and increment the count accordingly. Finally, we normalize the count and return the result as estimated probability.

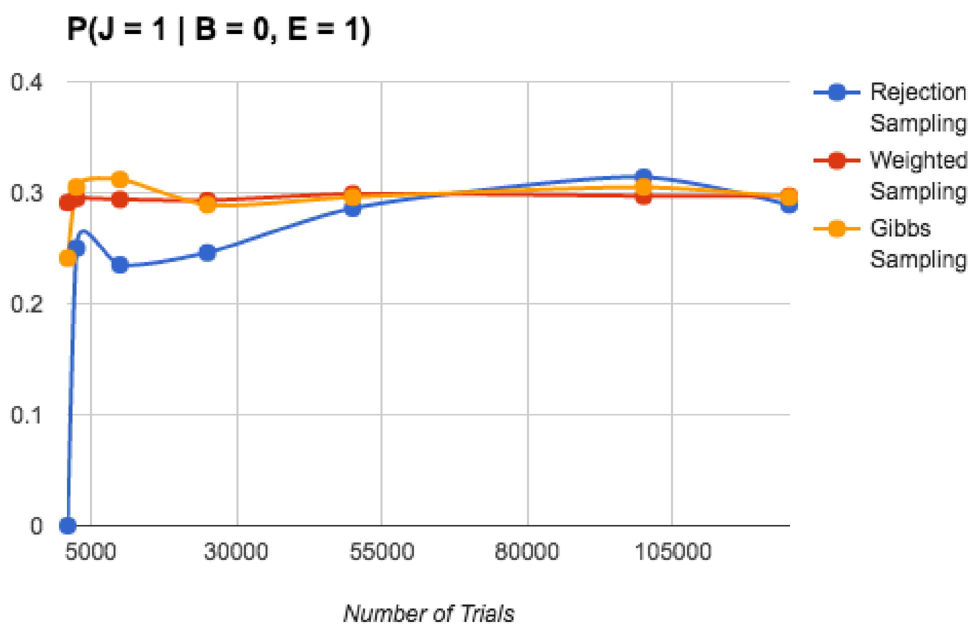
Likelihood sampling:
Likelihood sampling is another approximate inference algorithm that is more efficient than Rejection sampling. The idea is to make an weighted sample from the given evidence variables. We then use the sample to find the value of the query variable and add the weight of the sample corresponding to the value of the query variable in that sample. Finally, we normalize the total weighted sum of the query variable and return it as the estimated probability. We implemented Likelihood sampling by first implementing the Weighted-Sample that returns a weighted sample. We use the given evidence variables to make the sample, where the weight of the sample is the probability of the evidence variable's value. We then check the value of the query variable in the returned sample and add the weight of the sample to the corresponding value of query variable. The resulting sum is normalized and returned as the estimated probability.

Gibbs Sampling:
The idea of Gibbs sampling is to care only around The Markov Blanket of a node since those are the only things that affected the probability of a node. The Markov Blanket of a node includes its parents, children, and the parents of its direct children. Given that we know these information, a node is conditionally independent of every other node.
What we did is we set E, the evidence variables, to its given value while we randomized the initial value of Z, the non-evidence variables. For each iteration, we iterate the non-evidence variables and set its value based on sampling it given its Markov Blanket values. If the returned probability is the same as the one in query table, we increment the count of getting the same result. Finally, after the loop is done, we normalize the count by dividing it by the product of number of iteration and number of non-evidence variables.

b.

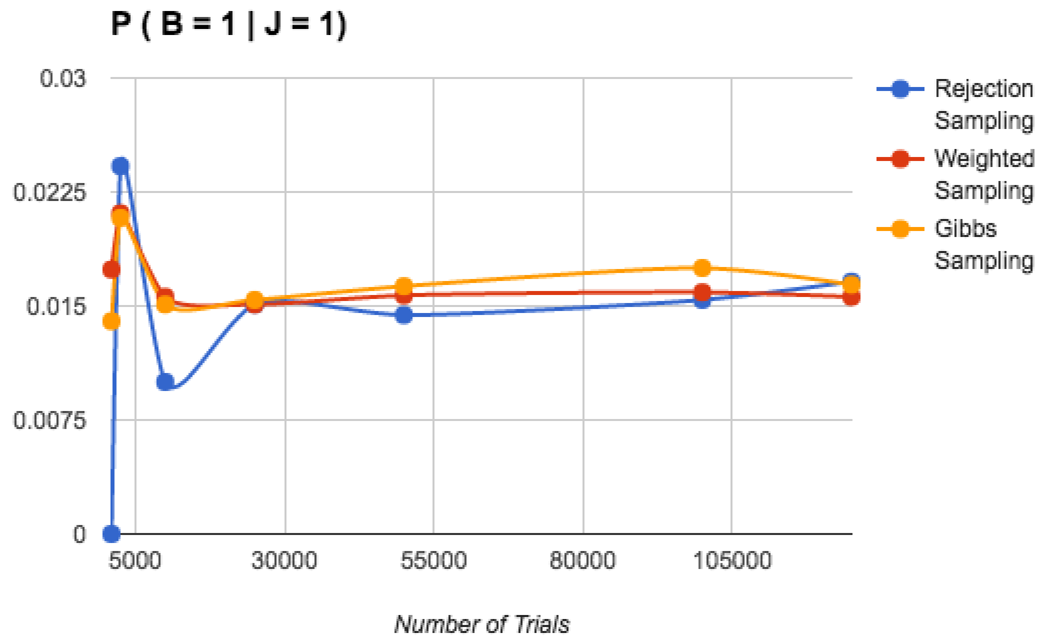
# P(J=1|B=0,E=1) = .2965			
Number of Trials	**Rejection Sampling**	**Weighted Sampling**	**Gibbs Sampling**
1000	0	0.291	0.241

2500	0.25	0.295	0.305
10000	0.235	0.294	0.312
25000	0.246	0.293	0.289
50000	0.286	0.299	0.296
100000	0.314	0.297	0.305
125000	0.289	0.297	0.296



# P(B=1|J=1) = .016			
Number of Trials	**Rejection Sampling**	**Weighted Sampling**	**Gibbs Sampling**
1000	0	0.0174	0.014
2500	0.0242	0.0211	0.0208
10000	0.01	0.0156	0.0151
25000	0.0151	0.0151	0.0154
50000	0.0144	0.0157	0.0163
100000	0.0154	0.0159	0.0175
125000	0.0166	0.0156	0.0164

Hansen T. Dharmawan | A91413023
Mark Darmadi | A91413023
Vania Christie Chandra | A91413023



c.

Rejected Sampling	
Test 1	**Test 2**
23773	100000
4774	9973
37637	32087
100000	43124
100000	34354
Average	48572.2

Weighted Sampling	
Test 1	**Test 2**
205	175
2390	6894
885	238
1327	238
345	696
Average	1339.3

Gibbs Sampling	
Test 1	**Test 2**
355	1714
272	970
1096	4283
8908	19435
629	1069
Average	3873.1

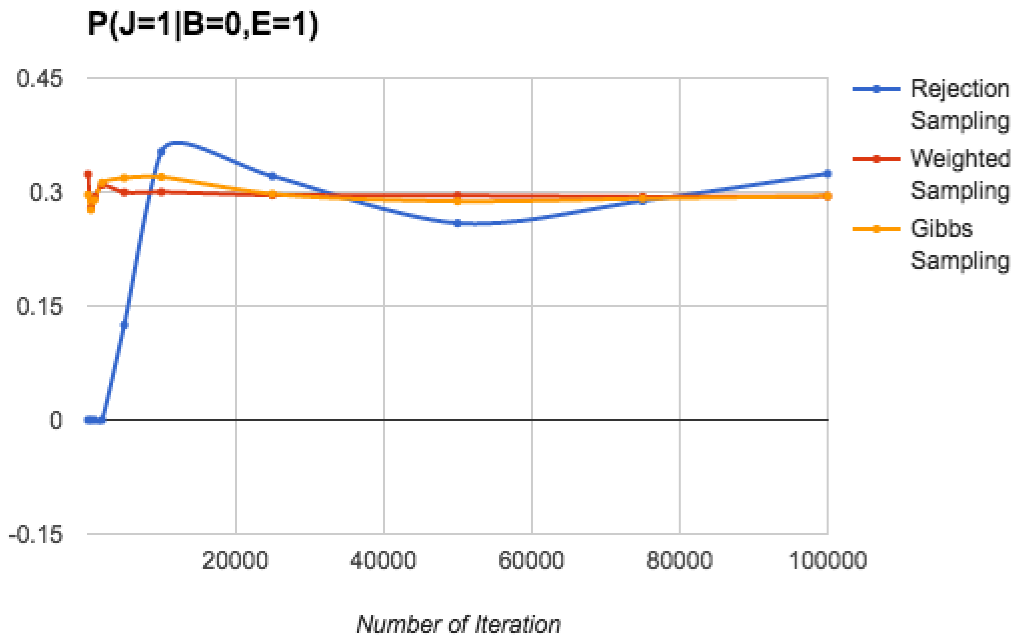
For all three algorithms, we repeated the same algorithm on the same query for 20 times to find the average time it takes for the result to converge. We take the absolute difference of the actual result and the current result and see if it is different by only 1%. If there are 10 consecutive results that are like that, we can say that it has converges.

Based on the result, rejection sample takes the longest to converge (average of 48,000 iterations to converge). Gibbs sampling and weighted sampling take way faster than rejection sampling. Both only takes 3000 and 1000 respectively to converge into the actual result.

d.

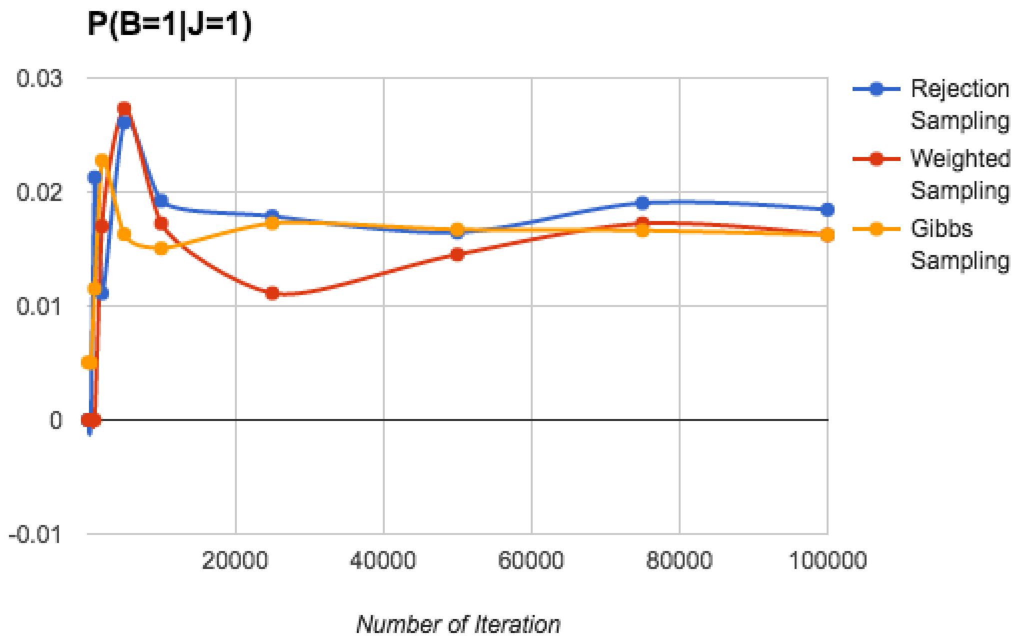
# P(J=1|B=0,E=1) = .2965			
Number of Iteration	**Rejection Sampling**	**Weighted Sampling**	**Gibbs Sampling**
100	0	0.3232323232	0.2962962963
500	0	0.2785571142	0.2765531062
1000	0	0.2932932933	0.2896229563
2000	0	0.3091545773	0.312156078
5000	0.125	0.299459892	0.3184636927
10000	0.3529411765	0.299929993	0.3197653099
25000	0.320754717	0.2958918357	0.2974518981
50000	0.2589285714	0.2953659073	0.2878457569
75000	0.2882352941	0.2936972493	0.2914172189
100000	0.3240740741	0.2944529445	0.2946429464

Hansen T. Dharmawan | A91413023
Mark Darmadi | A91413023
Vania Christie Chandra | A91413023



# P(B=1|J=1) = .016			
Number of Iteration	**Rejection Sampling**	**Weighted Sampling**	**Gibbs Sampling**
100	0	0	0.005050505051
500	0	0	0.00501002004
1000	0.02127659574	0	0.01151151151
2000	0.01111111111	0.01699716714	0.02276138069
5000	0.02608695652	0.02731929425	0.01630326065
10000	0.01923076923	0.017239728	0.01505150515
25000	0.01787101787	0.01111969112	0.01726069043
50000	0.01644988523	0.01451359331	0.01671033421
75000	0.01902313625	0.01723390695	0.0166202216
100000	0.01844380403	0.01622739414	0.01623516235

Hansen T. Dharmawan | A91413023
Mark Darmadi | A91413023
Vania Christie Chandra | A91413023

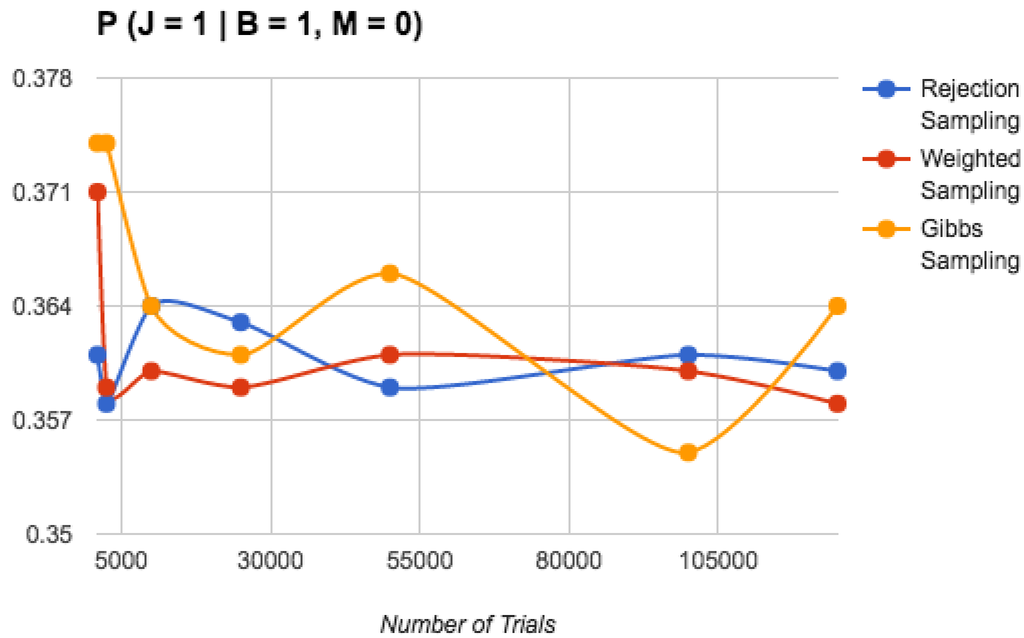


4.

a.

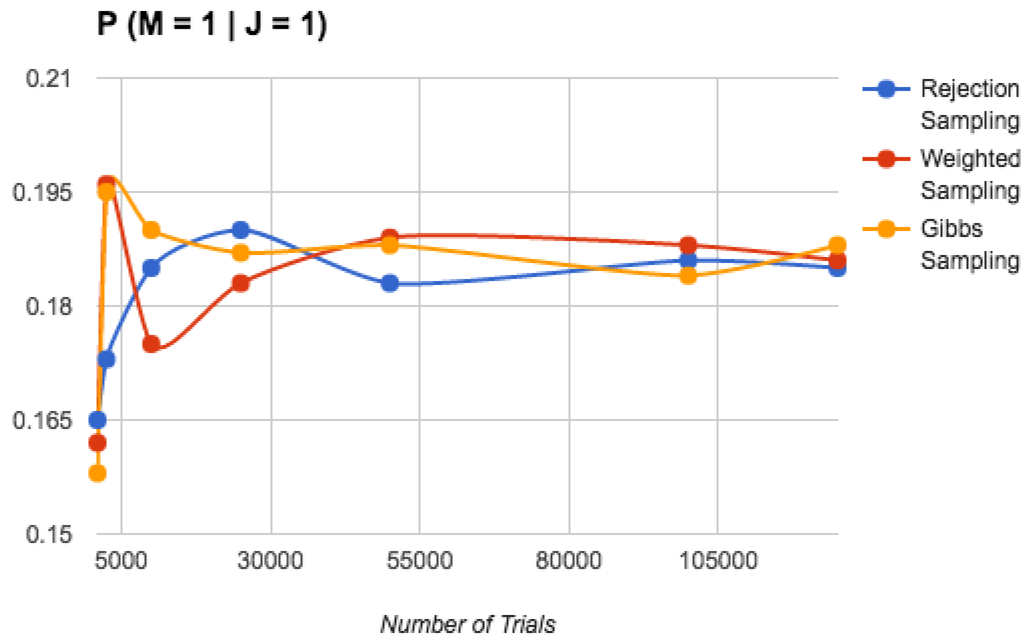
P(J=1|B=1,M=0)			
Number of Trials	**Rejection Sampling**	**Weighted Sampling**	**Gibbs Sampling**
1000	0.361	0.371	0.374
2500	0.358	0.359	0.374
10000	0.364	0.36	0.364
25000	0.363	0.359	0.361
50000	0.359	0.361	0.366
100000	0.361	0.36	0.355
125000	0.36	0.358	0.364

Hansen T. Dharmawan | A91413023
Mark Darmadi | A91413023
Vania Christie Chandra | A91413023



P(M=1|J=1)			
Number of Trials	**Rejection Sampling**	**Weighted Sampling**	**Gibbs Sampling**
1000	0.165	0.162	0.158
2500	0.173	0.196	0.195
10000	0.185	0.175	0.19
25000	0.19	0.183	0.187
50000	0.183	0.189	0.188
100000	0.186	0.188	0.184
125000	0.185	0.186	0.188

Hansen T. Dharmawan | A91413023
Mark Darmadi | A91413023
Vania Christie Chandra | A91413023



b.

Rejection Sampling	
Test 1	**Test 2**
23954	13836
9466	9452
9577	89141
100000	62052
72467	100000
Average	48994.5

Weighted Sampling	
Test 1	**Test 2**
1280	178
664	779
217	295
2391	149
8296	762
Average	1501.1

Gibbs Sampling	
Test 1	**Test 2**
2698	774
476	286
289	440
860	316
340	253
Average	673.2

For all three algorithms, we repeated the same algorithm on the same query for 20 times to find the average time it takes for the result to converge. We take the absolute difference of the actual result and the current result and see if it is different by only 1%. If there are 10 consecutive results that are like that, we can say that it has converges.
Based on the result, rejection sample takes the longest to converge (average of 48,000 iterations to converge). Gibbs sampling and weighted sampling take way faster than rejection sampling. Both only takes 600 and 1000 respectively to converge into the actual result.

5.

a. We want to find what is $P(x_i|x_1, x_2, \dots\ x_{i-1}, x_{i+1}, \dots, x_n)$ or $P(x_i|Y)$ where Y stands for every other variable in the network.

$P(x_i|Y) = \frac{P(x_i|Y)}{P(Y)}$ or after removing the abstraction,

$$P(x_i|x_1, x_2, \dots\ x_{i-1}, x_{i+1}, \dots, x_n) = \frac{P(x_1, x_2, \dots, x_n)}{P(x_1, x_2, \dots\ x_{i-1}, x_{i+1}, \dots, x_n)}$$

$$= \frac{P(x_1, x_2, \dots, x_n)}{\sum_{x_i} P(x_1, \dots, x_n)} \quad (\sim)$$

$$= \frac{\prod_{j=1}^{n} P(x_j|Parents(X_j))}{\sum_{x_i} \prod_{j=1}^{n} P(x_j|Parents(X_j))} \quad (\sim\sim)$$

$$= \frac{P(x_i|Parents(X_i)) \times \prod_{x_j \in Childrens(X_i)} P(x_j|Parents(X_j))}{\sum_{x_i} \prod_{j=1}^{n} P(x_j|Parents(X_j))}$$

$= \alpha\, P(x_i|Parents(X_i) \times \prod_{x_j \in Childrens(X_i)} P\left(x_j \middle| Parents(X_j)\right)$ (~~~)

$= P(X_i|mb(X_i))$ (~~~~) QED

(~) -> We can see that the denominator is just the numerator without xi and so we can sum it up.

(~~) -> From Lecture 18/19, slide 17, we can see that a node X is conditionally independent of its non ancestors given its parents.

(~~~) -> We can see that the denominator is just a normalization constant and so we can simplify it to alpha

(~~~~) -> Using the equation from part b, we can work backwards to reach the final equation.

b.

$$P(Z_i|mb(Z_i)) = P(Z_i|\ Parents(Z_i), Children(Z_i), Spouses(Z_i))$$

$$= \frac{P(Children(Z_i)|Z_i, Parents(Z_i), Spouses(Z_i))P(Z_i|Parents(Z_i), Spouses(Z_i))}{P(Children(Z_i)|Parents(Z_i), Spouses(Z_i))}$$

$$= \frac{P(Children(Z_i)|Z_i, Spouses(Z_i))P(Z_i|Parents(Z_i))}{P(Children(Z_i)|Spouses(Z_i))} \quad (\sim)$$

$$= \prod_{Y_j \in Children(Z_i)} P(y_j|Parents(y_j)) \times \frac{P(Z_i|Parents(Z_i))}{P(Children(Z_i)|Spouses(Z_i))} \quad (\sim\sim)$$

$$= \prod_{Y_j \in Children(Z_i)} P(y_j|Parents(y_j)) \times P\left(Z_i \middle| Parents(Z_i)\right) \times \alpha \quad (\sim\sim\sim)$$

$$= \alpha P\left(Z_i \middle| Parents(Z_i)\right) \times \prod_{Y_j \in Children(Z_i)} P(y_j|Parents(y_j)) \ (QED)$$

Parents(Zi) -> Parents of Zi

Children(Zi) -> Childrens of Zi

Spouses(Zi) -> All Children(Zi)'s parents except Zi

(~) -> See that we can get rid some of the conditions because there are conditionally independent. For example, the children of Zi is independent of its parents and so we can cross them out.

(~~) -> We can see that the probability of the children of Zi given Zi and its spouses is simply just all the parents of the children of Zi and so we can simplify it.

(~~~) -> We can see that 1/P(Children(Zi)|Spouses(Zi)) is just the normalizing constant (Using Exact Inference in Bayes Net, lecture 18/19 slide 36) and so we can replace it with alpha.

6.

Mark Darmadi:

I was in charge of implementing peformRejectionSample and peformWeightedSample for this program. Together with debugging and testing for all the algorithm. I also worked on part 5 partially. I learnt about the different algorithms that will give similar results even though some of them take longer to converge. I also learn to familiarize myself with API faster and learn alot about the markov blanket when debugging gibbs sampling. Doing number 5 taught me about the various manipulations of bayes rules, conditionalized bayes rules and marginalization.

Hansen Dharmawan:

I was in charge of debugging performRejectionSample and performWeightedSample as well as Gibbs sampling for this program. I also worked on number 5 of the report. I learnt about how to implement approximate inference in Bayesian network. In doing this program, I learnt that python always use pass by reference when copying object instead of pass by value i.e it does not actually create a new object when being copied. I also learnt a lot about how Bayesian network can be used to estimate the probability of a random variable without actually doing exact calculation.

Vania Chandra:

Hansen T. Dharmawan | A91413023
Mark Darmadi | A91413023
Vania Christie Chandra | A91413023

I was in charge of calculating the query for number 1 and 2 manually, debugging rejection sampling algorithm, doing the Gibbs sampling, and doing the testing on number 3 for both alarm network and the guilty network. I really learn a great deal about probability from doing the calculation on number 1 & 2 because it helps me understand Bayes rule and normalization. The calculation is long but when done step by step with care, we will eventually get into the result. By implementing and debugging all three methods of collecting probability, I also get to learn the disadvantage and strength of the different methods and how each works. Each methods help me to understand the material in class better.