Project 1 Report

Mikey Ferguson

February 2025

1 Introduction

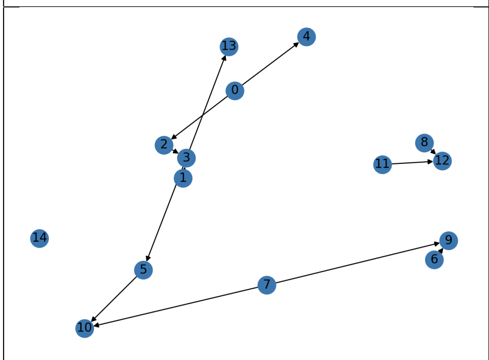
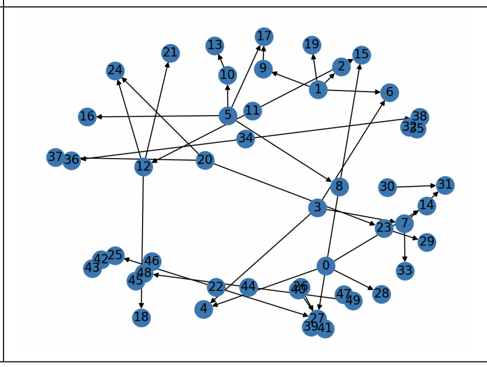
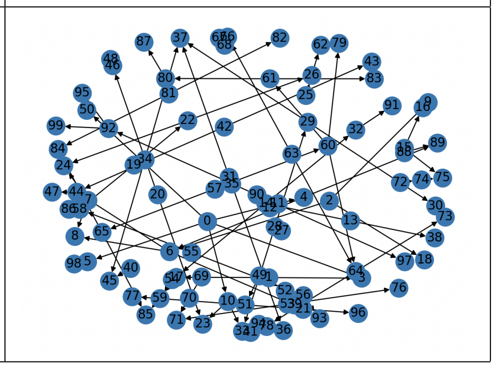
This project entailed utilizing four different algorithms to calculate the probability distribution of any generic query. The format of the outputs is the corresponding bitmask for the arranged binary query variables. For instance suppose for a Bayesian Network one particular query had two particular query variables. Then the corresponding bit masks with their probabilities would be:

- 0: p_0 (both variables false)
- 1: p_1 (first variable false, second variable true)
- 2: p_2 (first variable true, second variable false)
- 3: p_3 (first variable true, second variable true)

My algorithms spat out the vectorized probability distributions as **numpy** arrays, but my graphical results labeled each probability with its query and bit mask values, and the height on the graph is the specific bit mask's probability for that query. Therefore, when comparing different algorithms' results, the two points to compare on the separate graphs have the same query and bit mask values associated with themselves in the graph legend.
The queries were as follows:

- Query 0: $P(1, 3|2 = \textbf{False}, 5 = \textbf{True})$
- Query 1: $P(4, 5, 7|2 = \textbf{True}, 3 = \textbf{False}, 6 = \textbf{False})$
- Query 2: $P(2, 6, 10, 12|3 = \textbf{False}, 4 = \textbf{False}, 5 = \textbf{True}, 15 = \textbf{True})$

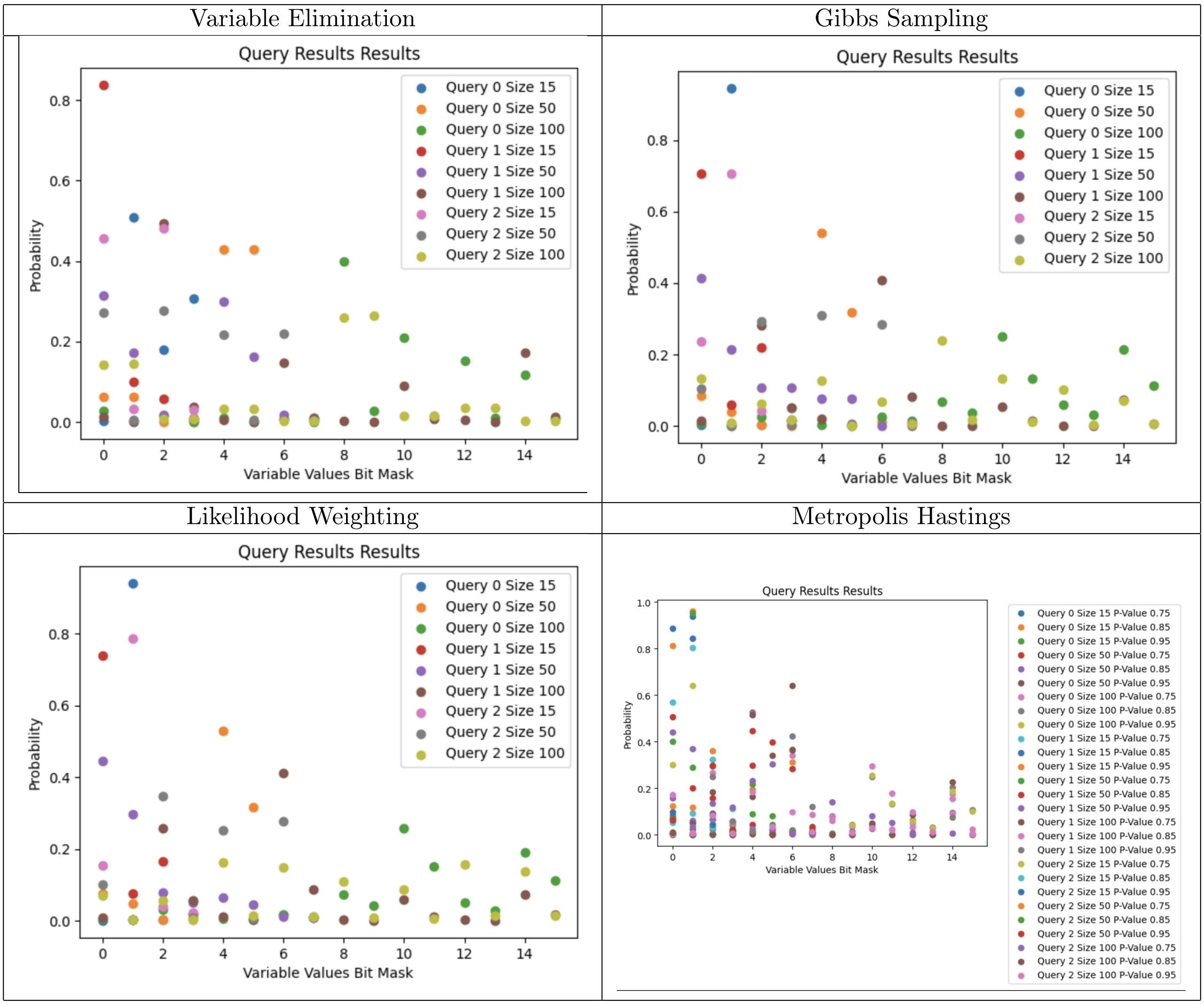
Furthermore, the Bayesian Network Polytrees of varying sizes were as follows:

Size 15	Size 50	Size 100
		

Also note that in the preceding graphics, labels such as "Query 0 Size 15" and "Query 0 Size 100" will appear. Two labels such as these corresponds to two completely different Bayesian Networks - one is size 15 while the other is 100. Therefore one should not expect such bit mask probabilities to be identical. Finally, for the Gibbs Sampling, Likelihood Weighting, and Metropolis Hastings Algorithms, I used 10000 iterations to approximate the query probability distributions.

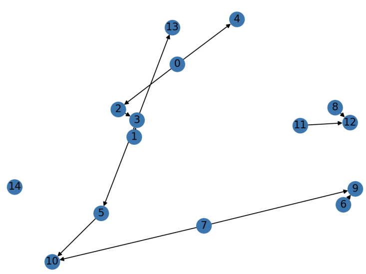
2 Performance

I broke up my testing into three different sizes for a Bayesian Network Polytree - 15, 50, and 100. Let us analyze the various algorithms' performances on each of these sizes. Before going any further, it is worth noting that the Metropolis Hastings algorithm had three times as many results for each graph compared to the other algorithms, because I needed to test out its p-values of 0.75,0.85,and 0.95. The performances of each of these p-values will be explored in the analysis.



2.1 Performance Analysis

One immediate point to note is that the results of the graphs do not indicate higher probabilities for lower bit mask values. While that may appear visually from the graph, that is only because some queries could not *achieve* higher bit mask values - for instance, the first query has two variables that are not hidden and not in the evidence; hence, its bit mask values can range from 0-3. However, the third query has four such variables and hence its bit mask values can range from 0-15.
Therefore, *for the same query*, the graph does not indicate a general trend in lower bit masks having higher probabilities.
The first thing to note is that all three approximate inference algorithms somewhat agreed with variable elimination in terms of which bit masks for which queries had the highest probabilities. Particularly, “Query 0 Size 15 Bit Mask 1”, “Query 1 Size 15 Bit Mask 0”, and “Query 2 Size 15 Bit Mask 1” were all high according to variable elimination. So were they for all three approximation algorithms. Interestingly, however, all three approximation algorithms rank “Query 0 Size 15” significantly higher than variable elimination. This could be indicative of the various algorithm techniques having a weakness against predicting upstream variables. Note that in “Query 1” are 1 and 3, while the evidence variables are 2 and 5. When these queries were applied to the small graph, here’s the picture:

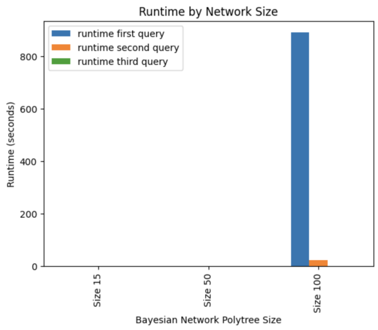
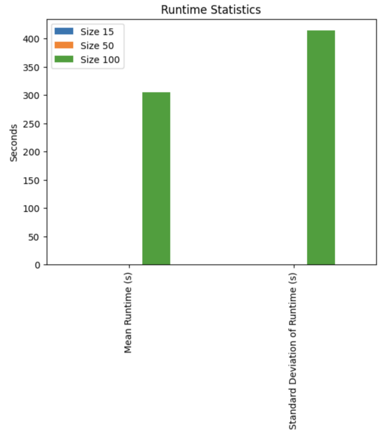
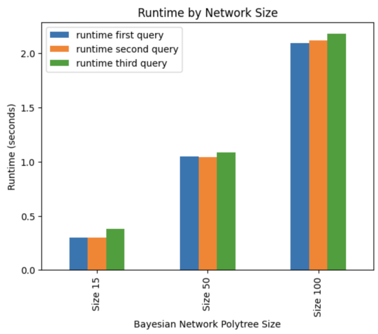
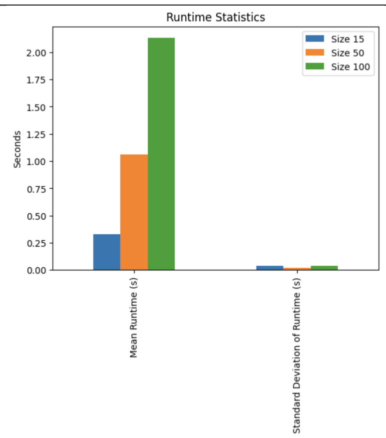
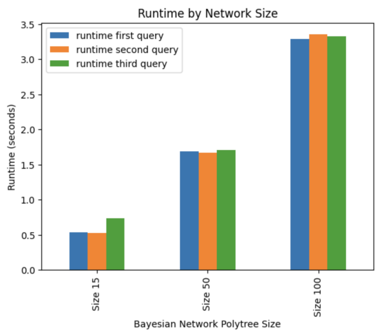
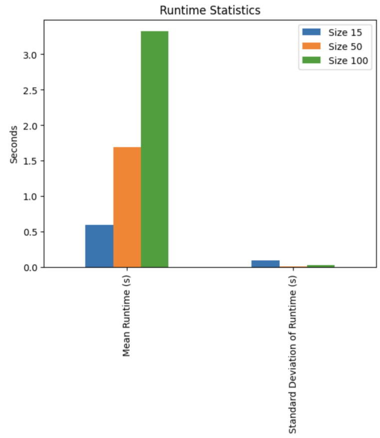
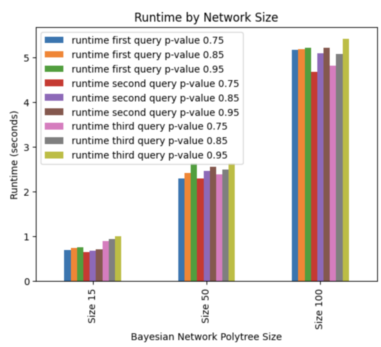
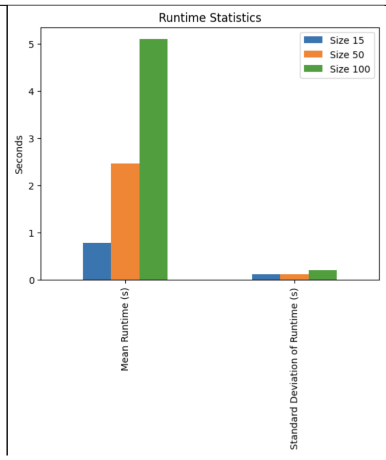


Observe that 1 is an upstream variable of 5 (which can be confirmed from the project’s .JSON file), a possible explanation for the approximation algorithms’ bias away from the variable elimination result for this query.
Another query worth observing is “Query 1”, with query variables 4, 5, and 7. Notably in the evidence, 3 is false. Gibbs Sampling and Likelihood Weighting approximate inference techniques marked bit mask value 0 with a lower probability than variable elimination. Meanwhile, Metropolis Hastings did not. Note that from the .JSON file, the probability of query variable 5 being true given that 3 is false is quite low at around 2.8%. While variable elimination calculates exact probability and Metropolis Hastings favors more likely states, Gibbs Sampling and Likelihood Weighting have a stochastic nature that could cause them to overestimate small probabilities - like 5 being true if 3 is false - if they run across their occurrences by chance. This could be why Likelihood Weighting and Gibbs Sampling underestimate the probability of a bit mask of 0 for “Query 1” - where 5 is false - to be less likely when it should be more likely since 3 is false.

3 Runtime

Runtime scaled predictably and reasonably for all approximation algorithms, but not so for variable elimination - as is to be expected. In fact, the runtime for Variable Elimination when the network size reached 100 nodes completely dwarfed all other run-times, which goes to show the exponential nature of variable elimination in the worst case. However, this was only really the case for the first query, which comes as no surprise as this query had the fewest number of evidence variables, eliminating fewer entries in each probability distribution.

Meanwhile, the run-times for all three approximation algorithms scaled linearly with respect to graph size, which also comes as no surprise. Interestingly, however, while average runtime increased as graph sized increased, the standard deviation of the runtime only increased for Metropolis Hastings as graph size increased. While variation in p-value had no significant effect on results or runtime, it varied runtime enough in increasing graph sizes to increased runtime standard deviations.

	Runtime	Runtime Statistics
Variable Elimination		
Likelihood Weighting		
Gibbs Sampling		
Metropolis Hastings		

4 Conclusions

All three approximation algorithms performed consistently with each other when predicting the same query for the same graph, and their deviations from variable elimination results were also uniform. Said deviations are possibly the result of poor approximation algorithm performance on upstream query variables. They could also be the result of small probabilities of query variables given their parent values. Metropolis Hastings seemed to offer reasonable performance for dealing with small probabilities, although it still struggled with upstream query variables just like Gibbs Sampling and Likelihood Weighting.