Comparison of Bayesian Network Inference Methods

Robert Geraghty

February 9th, 2020

1 Introduction

In this project, I implemented three different methods of inference for Bayesian networks, namely the enumeration-ask, likelihood weighting, and MCMC algorithms. In this report, I will compare the performance and accuracy of the different algorithms in inferring the probabilities of variables in Bayesian networks.

2 Implementation

I implemented these algorithms based on the pseudo-code that can be found in the third edition of the Russel and Norvig textbook *Artificial Intelligence: A Modern Approach*. I implemented them in python and also leveraged the package "networkx" to assist in building and working with the graph structures of the Bayesian networks. I used the provided "gen-bn" code to generate the graphs for testing.

3 Methods

To compare the different algorithms, I ran a series of tests testing the impact of graph size and structure, as well as the number of samples for the approximate inference algorithms, to compare both the time and accuracy of the three methods. For the time tests, I tested the algorithms on graph sizes from 1 to 20 nodes for both DAGs and polytrees, keeping a static 1,000 samples for the approximate algorithms. Testing graphs larger than 20 nodes would become prohibitively expensive in terms of time for the exact inference algorithm, and 20 nodes provided adequate data. To test the accuracy of the approximate algorithms, I tested them on both DAGs and polytrees with 15 nodes and varied the number of samples from 1 to 100,000 samples.

4 Results

4.1 Time Tests

Here we will look at the results of the time test on both DAGs and polytrees for the different algorithms.

4.1.1 DAG Time Tests

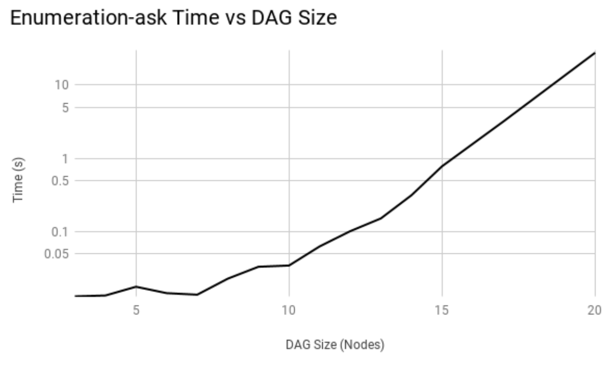

Figure 1: Comparison of exact inference time vs. number of nodes in DAG. Times are logarithmically scaled as the algorithm runs in exponential time.

First, we will look at the results of the time tests on DAGs for the exact inference, likelihood weighting, and MCMC algorithms. As we can see in Fig. 1, the exact inference algorithm takes exponential time as the number of nodes increases. When run on the 20 node DAG, the algorithm takes 27.7 seconds to complete execution. When compared with the likelihood weighting and MCMC sampling algorithms, shown in Fig. 2 and Fig. 3 respectively, we can see the immense time cost increase of the exact enumeration algorithm on larger Bayesian networks. At 20 nodes, the likelihood weighting algorithm takes 0.186 seconds and the MCMC algorithm takes 0.445 seconds. The difference between these two algorithms in terms of time can likely be attributed to the higher cost of computation involved in computing the probability based on the Markov blanket over just the conditional probability based on the parent nodes. At graphs of this size, the benefits the MCMC method provides in not having to reassign all of the nodes' values are outweighed by the computational complexity of computing probabilities. Both approximate methods perform similarly in terms of accuracy, with both achieving an error of 0.263% from the exact probability of the queried node.

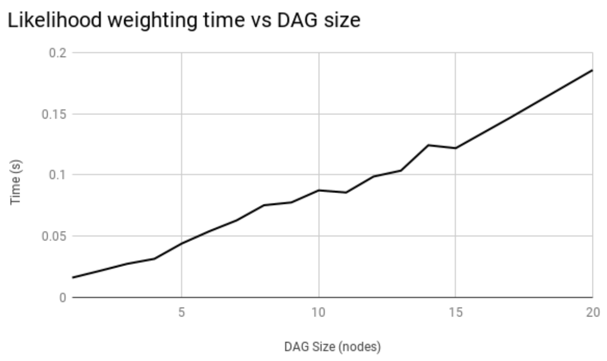

Figure 2: Comparison of likelihood weighting time vs. number of nodes in DAG.

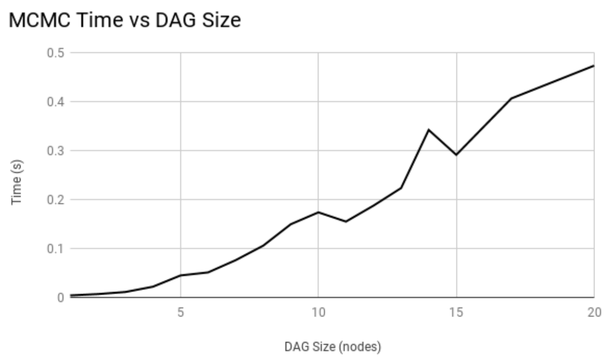

Figure 3: Comparison of MCMC time vs. number of nodes in DAG.

4.1.2 Polytree Time Tests

Next, we will look at the results of the time tests on polytrees for the exact inference, likelihood weighting, and MCMC algorithms. Firstly, we can see in Fig. 4 that MCMC still suffered from exponential time complexity in the polytree tests. This should not be the case and is indicative of an error within my implementation, but unfortunately, I was unable to find what was causing the error. The graphs produced by the "gen-bn" code used to generate the networks were indeed polytrees, but I fear that the need to use the "deepcopy" method in python resulted in a lot of overhead for the algorithm in terms of time usage. Still, we should not be seeing exponential growth. The approximate inference methods still showcased linear time complexity, as can be seen in Figs. 5 and 6, and ran slightly faster than they did on the DAG tests. Likelihood weighting ran 24.5% faster and the MCMC method ran 71.9% faster on a polytree of 20 nodes when compared to a DAG of the same size. The performance increases can be attributed to the smaller number of neighbors seen in most nodes, which especially increased MCMC performance due to the major decreases in the size of the Markov blankets of each node. These tests are the first time we see the MCMC method outperform the likelihood weighting method in terms of time. This difference is not extremely significant, but as sample sizes grow and networks grow, given that the network is fairly sparse of edges to prevent the expansion of Markov blanket sizes, MCMC's

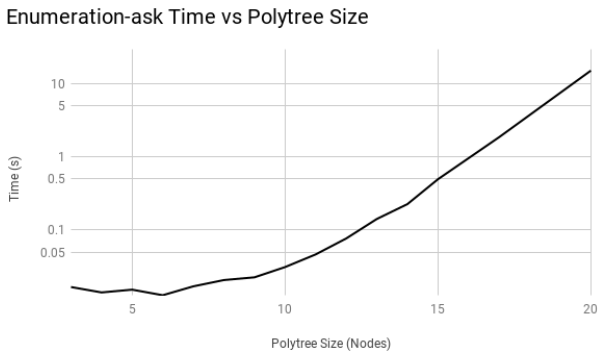

Figure 4: Comparison of exact inference time vs. number of nodes in polytree. Times are logarithmically scaled as the algorithm runs in exponential time.

technique of sample generation can prove to be faster. In terms of accuracy, both approximation methods were 3.77% off of the true probabilities. This increase in error may be attributable to random chance and/or network structure, as will be discussed in section 4.2.

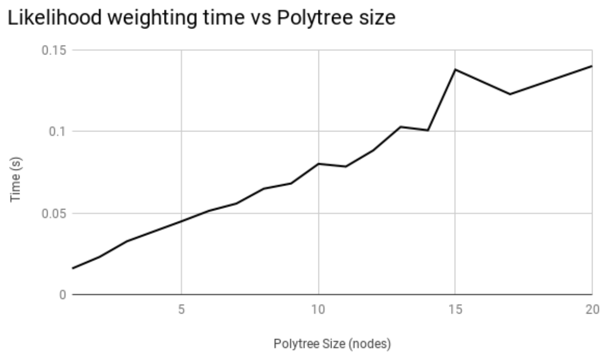

Figure 5: Comparison of likelihood weighting time vs. number of nodes in polytree.

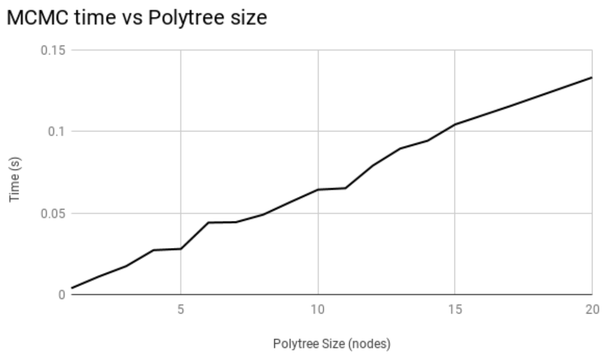

Figure 6: Comparison of MCMC time vs. number of nodes in polytree.

4.2 Approximate Accuracy Tests

In this section, we will review the accuracy of the two different approximate inference methods tested. All tests were conducted on graphs with 15 nodes, on both DAGs and polytrees, while varying the number of samples.

4.2.1 DAG Accuracy Tests

The results are in line with what is to be expected. Increasing the number of samples improves the accuracy of the approximation. The improvements are not guaranteed, however, and this most likely has to do with the structure of the graphs being tested and/or random chance with the smaller sample sizes, where the law of large numbers does not weigh in as heavily. We can see this in Figs. 7 and 8.

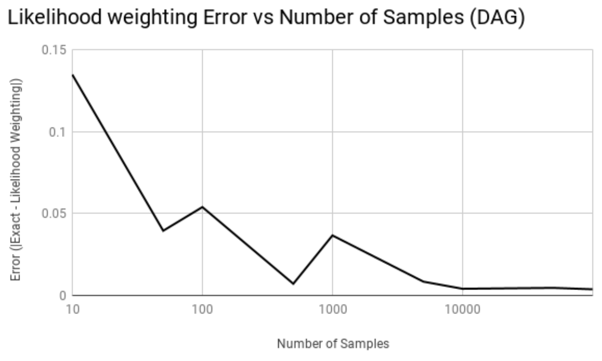

Figure 7: Comparison of likelihood weighting accuracy vs. number of samples in DAGs.

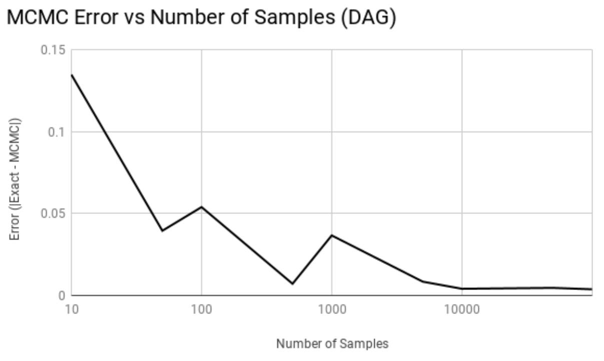

Figure 8: Comparison of MCMC accuracy vs. number of samples in DAGs.

However, what is of value is deciding how many samples are worthwhile in comparison to the costs and benefits of using the exact inference algorithm. The lowest error was of course achieved by the largest sample size for both algorithms, however, these tests took far more time than the exact inference method did. The exact inference method took ∼0.7 seconds to run on the tested networks, whereas the 100,000 sample tests took 12 and 29 seconds for the likelihood weighting and MCMC algorithms, respectively. These tests resulted in an error of 0.378% from the true probability of the queried node. When compared to using 1,000 samples, we see a runtime of 0.12 and 0.28 seconds for the likelihood and MCMC algorithms respectively, but an error of 3.67% from the true probability. This means that the decision to use approximate inference should be made with the factors of the required accuracy and size of the network in mind. On small networks, exact inference will most likely always be better, as the algorithm can run very quickly, but on larger networks, say 20 nodes, the approximate inference methods can result in fairly close approximations for far less computational effort.

4.2.2 Polytree Accuracy Tests

Much like the DAG accuracy tests, we see a clear increase in accuracy as the number of samples increases. However, when testing on polytrees the increase in accuracy provided by larger sample sizes diminishes much faster than in the DAG tests, as can be seen in Figs. 9 and 10.

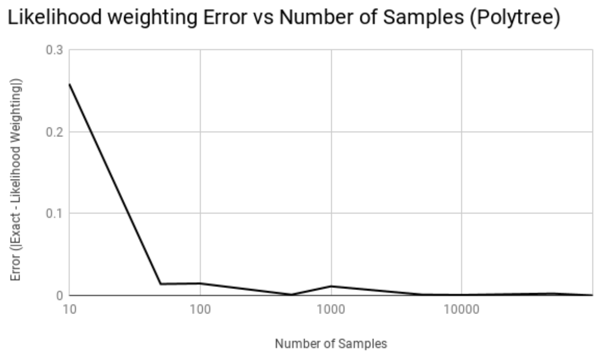

Figure 9: Comparison of likelihood weighting accuracy vs. number of samples in polytrees.

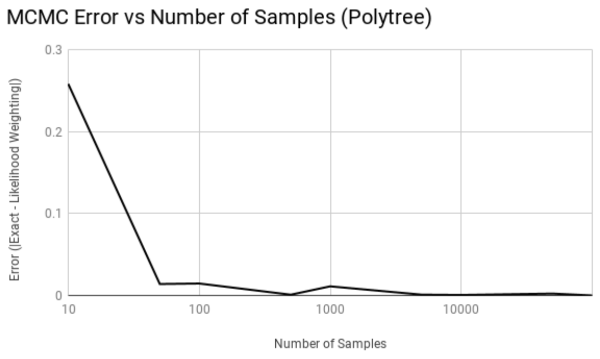

Figure 10: Comparison of MCMC accuracy vs. number of samples in polytrees.

In these tests, there is very little gain to be had from going past 500 samples. 500 samples resulted in an error of 0.094% from the approximate inference methods, which when compared with the 0.00018% error of 100,000 samples, is not that significant unless extremely accurate approximations are required.

5 Conclusion

In conclusion, we can see that the exponential nature of the exact inference method can prove to be prohibitively computationally expensive, as well as the benefits and drawbacks of approximate inference schemes. The data from the conducted tests show likelihood weighting as a better approximate method than MCMC for graphs of these sizes, especially for DAGs. With further testing, it could be seen that MCMC outperforms likelihood weighting for larger and sparse graphs with more samples, which were not tested in this project. Overall, approximation methods have proven to be valuable for larger networks where exact probabilities are not required or possible to compute and with large numbers of samples they can come very close to the true probabilities of nodes within Bayesian Networks.