Structure Learning in Bayesian Networks of Moderate Size by Efficient Sampling (Supplementary Material)

Ru He
Department of Computer Science and Department of Statistics
Iowa State University
Ames, IA 50011, USA

Jin Tian
Department of Computer Science
Iowa State University
Ames, IA 50011, USA

Huaiqing Wu
Department of Statistics
Iowa State University
Ames, IA 50011, USA

Editor:

1. Proof of Theorem 6 (iv)

Proof that if the quantity $\Delta = \sum_{G \in G} p_{\hat{\phi}}(G|D)$, then $\Delta \cdot p_{\hat{\phi}}(f|D) \leq p_{\hat{\phi}}(f|D) \leq \Delta \cdot p_{\hat{\phi}}(f|D) + 1 - \Delta$.

On one hand,

$$
\Delta \cdot p_{\hat{\phi}}(f|D)
= \sum_{G \in G} p_{\hat{\phi}}(G, D) \cdot \frac{\sum_{G \in G} f(G)p_{\hat{\phi}}(G, D)}{\sum_{G \in G} p_{\hat{\phi}}(G, D)}
= \sum_{G \in G} f(G)p_{\hat{\phi}}(G, D)
= \frac{\sum_{G} f(G)p_{\hat{\phi}}(G, D)}{p_{\hat{\phi}}(D)}
\leq \sum_{G} \frac{f(G)p_{\hat{\phi}}(G, D)}{p_{\hat{\phi}}(D)}
= p_{\hat{\phi}}(f|D).
$$
On the other hand,

\[
\Delta \cdot \hat{p}_\Delta(f|D) + 1 - \Delta \\
= \sum_{G \in \mathcal{G}} f(G)p_\Delta(G, D) + \sum_{G \in \mathcal{G}} f(G)p_\Delta(G, D) \\
\geq \sum_{G \in \mathcal{G}} f(G)p_\Delta(G, D) + \sum_{G \in \mathcal{G}} f(G)p_\Delta(G, D) \\
= \sum_{G \in \mathcal{G}} f(G)p_\Delta(G, D) \\
= \hat{p}_\Delta(f|D).
\]

Combining the two proved inequalities, the whole proof is done.

2. Supplementary Experimental Results for the DDS

As a supplement to Section 4.1 (main paper), this section shows more experimental results for the DDS by varying the sample size. With the same experimental settings as Section 4.1, we performed the experiment for the data cases Tic-Tac-Toe, Wine, Child with \(m = 500\), and German. By examining each figure from Figures 1, 3, 5, 7, and the corresponding figure from Figures 2, 4, 6, 8, we can conclude that the learning performance of the DDS with each sample size is significantly better than the one of the PO-MCMC in each data case.

3. Supplementary Experimental Results for the IW-DDS

As a supplement to Section 4.2 (main paper), this section shows more experimental results for the IW-DDS by varying the sample size. The experiment was performed for the data cases Wine, Child with \(m = 500\), and German. By examining each figure from Figures 9, 11, 13, and the corresponding figure from Figures 10, 12, 14, we can clearly see the advantage of the IW-DDS in the structure learning over the other two methods for each data case.
Figure 3: Plot of the SAD Performance of the PO-MCMC and the DDS for Wine

Figure 4: Plot of the Total Running Time of the PO-MCMC and the DDS for Wine

Figure 5: Plot of the SAD Performance of the PO-MCMC and the DDS for Child (m = 500)

Figure 6: Plot of the Total Running Time of the PO-MCMC and the DDS for Child (m = 500)

Figure 7: Plot of the SAD Performance of the PO-MCMC and the DDS for German

Figure 8: Plot of the Total Running Time of the PO-MCMC and the DDS for German
Figure 9: Plot of the SAD Performance of the DP+MCMC, the $K$-best and the IW-DDS for Wine

Figure 10: Plot of the Total Running Time of the DP+MCMC, the $K$-best and the IW-DDS for Wine

Figure 11: Plot of the SAD Performance of the DP+MCMC, the $K$-best and the IW-DDS for Child ($m = 500$)

Figure 12: Plot of the Total Running Time of the DP+MCMC, the $K$-best and the IW-DDS for Child ($m = 500$)
4. Memory-saving Strategies for the DDS and the IW-DDS with a Very Large $N_o$

In this section, we briefly describe our memory-saving strategies for the DDS and the IW-DDS if a very large number of DAG samples are required by a user for his specific requirement. As you will see, while the memory-saving strategy for the DDS is very straightforward, the memory-saving strategy for the IW-DDS is relatively more complicated since it needs to ensure that all the duplicate DAGs among the $N_o$ sampled DAGs are eliminated.

As introduced in Section 3.2 (main paper), the overall memory cost of the DDS algorithm is $O(n^2 + nN_o)$: Step 1 of the DDS has $O(n^2)$ memory cost; and Steps 2 and 3 of the DDS have $O(nN_o)$ memory cost. Since typically tens of thousands of DAG samples are sufficient for estimating $p_\prec (f|D)$, the $O(nN_o)$ space cost coming from Steps 2 and 3 does not become an issue at all. However, if a very large $N_o$ (such as being above the magnitude of $1 \times 10^6$) is needed for the estimation due to some specific requirement of a user, some memory-saving strategy needs to be used in Steps 2 and 3 to reduce their memory cost. One simple memory-saving strategy is as follow: $N_o$ can be replaced with a smaller value of $N_{bl}$ in the DDS algorithm, where $N_{bl}$ is the size of a sample block, and then Steps 2 and 3 can be repeated $\lceil N_o/N_{bl} \rceil$ times. This will not change the properties of the estimator coming from the DDS algorithm but can reduce the overall memory requirement of the DDS to $O(n^2 + nN_{bl})$. Note for the performance of our time-saving strategy for the DAG sampling step (described in Remark 5), a large $N_{bl}$ is actually preferred. Thus, $N_{bl}$ can take a value that is large but still does not lead to the memory issue for a computer. (For instance, the value of $N_{bl}$ can be set to be around several millions for a computer with 2.0 to 8.0 GB memory.) In addition, note that the estimator $\hat{p}_\prec (f|D)$ can be constructed by Eq. (5) on the fly when each DAG gets sampled so that the memory of storing all the $N_o$ sampled DAGs can be saved.

Similarly, if a very large $N_o$ (such as being above the magnitude of $1 \times 10^6$) is needed in the IW-DDS to obtain $\hat{p}_\prec (f|D) = \left( \sum_{G \in G} f(G)p_\prec (G,D) \right)/\left( \sum_{G \in G} p_\prec (G,D) \right)$ or the corresponding sound interval, the following memory-saving strategy can be used to reduce the memory requirement of the IW-DDS algorithm. $N_o$ can be replaced with a smaller value of $N_{bl}$ in the IW-DDS algorithm, where $N_{bl}$ is the size of a sample block, and then Steps 2 and 3 of the DDS as well as the bias correction step can be repeated in $\lceil N_o/N_{bl} \rceil$ iterations. (Similar to the DDS algorithm,
$N_{bl}$ can take a value that is large but still does not lead to the memory issue for a computer.) In each iteration, after the bias correction step which eliminates the duplicates among the $N_{bl}$ DAGs is done, up to $N_{bl}$ DAGs get sorted according to $p(D|G)$ and then are stored as a file on the hard disk. Finally, after these $\lceil N_o/N_{bl} \rceil$ iterations are finished, the elimination of the duplicate DAGs across the DAGs stored in $\lceil N_o/N_{bl} \rceil$ files is performed as follows: each time some score threshold $p_{thr}$ can always be found (based on $p(D|G)$ of the corresponding quantile of the sorted DAGs) such that $O(N_{bl})$ DAGs whose $p(D|G)$ is no greater than $p_{thr}$ are newly retrieved from these $\lceil N_o/N_{bl} \rceil$ files and reloaded into the memory each time. Thus, the elimination of the duplicate DAGs only needs to be performed among these DAGs in the memory so that both the denominator ($\sum_{G \in G} \hat{p}(G, D)$) and the numerator ($\sum_{G \in G} f(G)p(G, D)$) of $\hat{\mu}(f|D)$ can be updated accordingly. When all the DAGs have been retrieved from these $\lceil N_o/N_{bl} \rceil$ files, $\hat{\mu}(f|D)$ is obtained and its denominator $\sum_{G \in G} p(G, D)$ can also be used to obtain $\Delta$. Using this strategy, the expected time cost of the IW-DDS becomes $O(n^{k+1}C(m) + kn2^n + n^2N_o + n^{k+1}N_o + N_{bl}\log(N_{bl})[N_o/N_{bl}] + C_{w,r}(n)N_o + nN_o) = O(n^{k+1}C(m) + kn2^n + n^2N_o + n^{k+1}N_o + \log(N_{bl})N_o + C_{w,r}(n)N_o)$, where $C_{w,r}(n)$ is the time cost that a DAG of $n$ nodes is written to the hard disk and then is reloaded from the hard disk to the memory. The corresponding memory requirement is $O(n2^n + nN_{bl})$, with the addition of the $O(nN_o)$ space in the hard disk required to record $O(N_o)$ DAGs among the $\lceil N_o/N_{bl} \rceil$ files.

We demonstrate the performance of our memory-saving strategy for the IW-DDS based on the data case Insur19 with $m = 200$ in Figures 15 and 16. In our experiment we fixed $N_{bl} = 2 \times 10^6$ as the size of the sample block. We increased $N_o$ from $2 \times 10^6$ to $1.2 \times 10^7$ with eachincrement $2 \times 10^6$ and showed the corresponding change of $\Delta$ and the running time in Figures 15 and 16 (20 independent runs were performed for the data case to get the results.) By temporarily storing the sampled DAGs in the hard disk, our IW-DDS (equipped with our memory-saving strategy) is shown to be able to efficiently sample $N_o = 1.2^7$ DAGs so that the resulting mean of $\Delta$ can reach 95.39% with the time cost $\hat{\mu}(T_1) = 1,933.44$ seconds.

Figure 15: Plot of $\Delta$ versus $N_o$ for Insur19 ($m = 200$)

Figure 16: Plot of the Running Time versus $N_o$ for Insur19 ($m = 200$)