Using Dynamic Bayesian Networks to analyze genetic data

Zhen Wang\textsuperscript{a,*} and Andrew Wong\textsuperscript{b}

\textsuperscript{a}School of Computing, Queen’s University, Kingston, Canada;  
\textsuperscript{b}School of Computing, Queen’s University, Kingston, Canada

ABSTRACT
In System biology, many statistical approaches are used to analyze gene expression data and infer gene regulatory network. Dynamic Bayesian Network (DBN) is a well-known method that has produced promising results when used to analyze time-series data. Here, we adopt a Dynamic Bayesian Network algorithm with Markov Chain Monte Carlo (DBmcmc) developed by Dirk Husmeier to investigate gene expression data. We evaluated the algorithm’s performance on a synthetic yeast time-series data and then applied it to the rat CNS development temporal data\textsuperscript{[1]} to reverse engineer the gene network. The resulting network is then validated against previous studies on the same data set as well as evidence from biological databases and literature. The interactions documented in biological literature show that DBmcmc was able to correctly identify subnetworks of interacting genes that were involved in the same biological pathways.

Keywords: Dynamic Bayesian Network, Markov Chain Monte Carlo, Gene Networks, Structure Learning

1 INTRODUCTION
With the development of microarray technology, determining gene networks from microarray expression data has become a main research focus in the post-genomic era. A genetic network attempts to model the interactions between genes to discover casual relationships. There are many different methods for determining gene networks. One promising approach is Dynamic Bayesian Networks (DBNs). The statistical properties of DBNs allow it to estimate the relationships among genes objectively. Furthermore, DBN allows scientists to incorporate prior knowledge into the data analysis algorithm. Also, DBN can handle time series data and therefore can reflect causability. However, the complexity for its structure learning is exponential to the number of variables and is considered a NP-hard problem\textsuperscript{[2]}. Normally, it is too computationally expensive to use DBN for gene expression data due to the large number of genes involved. Therefore, search heuristics that reduces the number of potential graph structures must be used to generate the DBN. Many search heuristics exist for learning the structure of DBN. Here we adopt DBN with Markov Chain Monte Carlo (DBmcmc) developed by Dirk Husmeier to analyze our time-series gene expression data.

The main goal of this project, therefore, is to discover biologically relevant gene regulatory network using DBmcmc. Next section gives a brief introduction about DBmcmc. The datasets and their gene networks that are produced by DBmcmc will be presented in the Results section. Also, the learned structure will be validated using evidence from biological databases (KEGG Pathway, GO, NCBI) and literature.

2 Methods
A Bayesian Network (BN) is a directed acyclic graph, $G$. The graphical structure $G$ includes nodes which denote variables and edges between the nodes represent their conditional dependencies.

The learning of a Bayesian Network is basically searching for the graph with the highest posterior probability given data. Based on the Bayesian Formula, given the gene expression data D, the probability of $G$ is:

\[
P(G|D) = \frac{P(G,D)}{P(D)} = \frac{P(D|G)P(G)}{P(D)}. \tag{1}\]

Theoretically, with a sufficiently large set of data, the graph structure that exactly captures all dependencies in the distribution will receive a higher probability than all other graphs, as proposed by Friedman and Yakhini\textsuperscript{[3]}.  

\textsuperscript{*}Zhen Wang. E-mail: zhenw@cs.queensu.ca, Telephone: +1(613)533-6000 ext 74201
In a Dynamic Bayesian Networks model, we observe gene expression values at different time points. The assumption of Dynamic Bayesian Network is that an event at time $t$ is only influenced by event at time $t-1$, and the conditional probability is defined as

$$P(X_t | X_{t-1}) = \prod_i P(X_i^t | Pd^G(X_i^t)).$$  \hspace{1cm} (2)

Dynamic Bayesian Network is essentially a two-slice Bayesian Network that captures temporal relationships. The nodes in the first slice do not have parameters associated with them, and the ones in the second slice have an associated conditional probability distribution which is given by (2). As is shown in Figure 2, DBN can reflect recurrent networks well.

As mentioned previously, the number of possible network structures is exponential to the number of nodes. Therefore, searching through all possible structures to find the optimal network is impractical. Therefore, a search heuristic must be employed in order to generate the resulting network. In our study, we apply Husmeier’s method mentioned in [4]. The algorithm, Markov chain Monte Carlo simulation, starts with random nodes as root nodes and generates a sequence of networks [5] based on the Metropolis-Hastings acceptance criterion (MHAC) to determine the final network [6]. Given an initial network $G_{old}$, the algorithm generates a new network $G_{new}$ by either adding, removing, or reversing an edge. This network is then scored using the MHAC, which is a function based on the posterior probability of the new network given data, $P$, and the probability of the new network given the old network, $Q$. This MHAC can be expressed as follows:

$$P = \min\{1, \frac{P(G_{new}|D)}{P(G_{new}|D)} \times \frac{Q(G_{old}|G_{new})}{Q(G_{new}|G_{old})}\}.$$  \hspace{1cm} (3)

where the posterior probability of the network given data is determined using Bayesian Scoring (2). At the end of the MCMC simulation, the algorithm samples a user-defined number (e.g. 20,000) of networks from the generated sequence. The network with the highest posterior probability at the end of the simulation is then chosen as the final network. The algorithm also includes a burn-in step where an initial number (e.g. 20,000) of networks are discarded from the chain before sampling. This is because the initial networks are not stable and therefore, are not reliable.

The DBN with Markov Chain Monte Carlo algorithm is implemented in MATLAB using the DBmcmc toolbox written by Dirk Husmeier, which invokes subroutines of the BNT toolbox given by Murphy, both of which are publicly available online.

### 3 Results

#### 3.1 Synthetic Data

To evaluate the performance of this method, we first apply it to a synthetic yeast time series simulation data given by Husmeier. This simulated time series is binary in nature due to requirements of DBmcmc. The dataset is generated from a true yeast cell cycle network $G_0$ from Friedman [7] and its real network is shown in Figure 1(a). We used the dataset to produce a Dynamic Bayesian Network using DBmcmc. The result is then compared with the true network to evaluate the performance of DBmcmc.

(a) True Bayesian network $G_0$. It is of a sub network of the yeast cell cycle, taken from Friedman [7]; 38 unconnected nodes were included, 50 nodes totally. (b) Calculated network after adding noise. The solid line between ACE2 and RNR3 is false discovered and dotted lines mean the missing edges.

Figure 1. Synthetic dataset.
We first applied DBmcmc on the synthetic data. If we choose the posterior probability of 0.5 as a threshold to cut edges, two false edges are discovered. If we set the threshold as 0.8, the true network will be recovered by the algorithm. This shows that DBmcmc can effectively recover the underlying gene network from data. Furthermore, we also checked the consistency of the algorithm by adding noise to the data and re-evaluating its performance. We randomly changed 10% of the data by flipping the binary expression values and re-ran DBmcmc on the modified data. The resulting network from the noisy data, with the threshold 0.8 is shown in Figure 1(b).

This result shows that even with 10% noise in the data, the algorithm is still able to detect most of the true edges. Moreover, since noise is added randomly, the algorithm will generate different networks every time. On average, across repeated simulations (250 times), the algorithm was able to recover about 80% of the true edges.

3.2 Real rat CNS data
3.2.1 Data preprocessing
In our study, we used a subset with 65 genes from the original data generated by Wen et al. [1]. Since the expression values in this dataset are continuous, it had to be preprocessed before DBmcmc can be applied. This involved the discretization of the dataset into three categories: under-expressed, normal, and over-expressed.

First, in order to make gene expressions comparable, we normalized data and did cubic spline interpolation to have more time points. To discretize the data, we calculated the mean $\mu_i$ and standard deviation $\sigma_i$ of each gene $\text{Gene}_i$, $i = 1, 2, ..., 65$ correspondingly. For $\text{Gene}_i$, if the expression value is between $\mu_i - \alpha \sigma_i$ and $\mu_i + \alpha \sigma_i$, we set its value as 0 (normal), less than $\mu_i - \alpha \sigma_i$ is -1 (under), and the rest will be set as 1 (over). By experimenting with different thresholds, we empirically determined that $\alpha = 1.0$ gave the optimal discretization. By setting $\alpha = 1.0$, 68% of the gene expression would be classified as normally expressed and are only over-expressed or under-expressed 32% of the time.

3.2.2 Gene Network
The DBmcmc simulation outputs a $n \times n$ matrix that shows the interactions between genes in the DBN as a conditional probability. The algorithm begins with random root nodes, different simulations produce different results. Therefore, to generate our gene interaction network, we conducted 5 simulations and selected the edges that appeared consistently in the different outputs. To do this, we took the sum of the five $n \times n$ interaction matrices, and applied a threshold to remove interactions where the sum of its probability was less than two. On average, this meant that an interaction have to appear be in at least 3 networks with probability greater than 0.7 to be included. Using this strategy, the gene network is produced, 56 genes are included with 94 interactions identified.

3.2.3 Validation
Since the network as a whole is hard to analyze due to its size and connectivity, we focused on identifying and validating selected subnetworks.

In order to extract subnetworks that may be significant, we first grouped the genes into groups based on the functional categories highlighted by Wen et al. in 1998 [1]. Then, based on these functional categories, we looked for subnetworks where the majority of the participating genes belong to the same functional group. Based on this heuristic, 4 potential subnetworks were selected.

<table>
<thead>
<tr>
<th>SubNetwork(SN)</th>
<th>Functional Groups</th>
</tr>
</thead>
<tbody>
<tr>
<td>SN1</td>
<td>Neurotransmitter Metabolizing Enzymes (GAD–), Glutamate Receptor (mGluR–)</td>
</tr>
<tr>
<td>SN2</td>
<td>Heparin-binding Growth Factors (–GF)</td>
</tr>
<tr>
<td>SN3</td>
<td>Acetylcholine Receptors (nAChR–)</td>
</tr>
<tr>
<td>SN4</td>
<td>Serotonin Receptors(5HT–), GABA-A Receptors (GR–)</td>
</tr>
</tbody>
</table>

In SN1 shown in Figure 2(a)), there were many interactions between different forms of GAD and mGluR genes. Evidence from literature supports these interactions as GAD seems to be involved with GABA synthesis [8] and mGluR has an inhibitory effect on GABA [9]. Aside from interactions between these two genes, PDGF and IP3R were also shown as parents of mGluR and GAD. These interaction are also relevant because according to the KEGG database, both PDGF and IP3R are involved in a Calcium Signalling Pathway and Calcium release causes the release of glutamate, which binds to the receptor mGluR [10].

In the second subnetwork, SN2, the gene GAP43 is identified as the parents of many Growth Factors (GF). This interaction resonates with the fact that GAP43 is a regulator of PDGF, IGF, and FGF. [11], [12], [13] This subnetwork also show an interaction between EGF and bFGF, which are genes that are related neuron growth [14].

Our third network also identified interactions between genes that reflect the underlying biology. In SN3, BDNF is identified as a parent of nAChR, which correlates with the findings from Massey et al. [15] that BDNF increases
the amount of nAChRs in hippocampal neurons. This subnetwork also detected an interaction between PDGFR and IGF, which are genes that may contribute to cell proliferation since their inhibition causes cell death.

Our final subnetwork mainly consists of interactions between serotonin receptors (5HT) and GABA receptors (GR–). Biologically, this seems to be a well-known interaction as there are several sources that mention the inhibition of GABA receptors by serotonin receptors [16], [17].

(a) Subnetwork 1: The GAD and mGluR network.  
(b) Subnetwork 2: The Growth Factors network. 
(c) Subnetwork 3: The BDNF and nAChR network.  
(d) Subnetwork 4: The 5HT and GABA network. 

Figure 2. Synthetic dataset.

Aside from evidence from biological literature, we also compared our results with a previous study [18]. In the study conducted by Haeseleer, who used linear modeling to analyze the gene interactions, he found correlation between 5HT and GABA, as well as mGluR and GAD, which were also detected in our subnetworks, SN4 and SN1, respectively. The author also mentioned tight correlation between PDGFR, MK2, aFGF, EGF and bFGF, which are the majority of the genes that appear in our subnetwork SN3.

4 Conclusion

In conclusion, it seems that DBN using the Markov Chain Monte Carlo algorithm is effective in discovering subnetworks of gene interactions that are biologically significant. However, the global network that is produced is not as reliable as the local subnetworks because the algorithm has a tendency to introduce false edges, which is in agreement with the tests performed on the noisy synthetic data. Therefore, given expression data of unfamiliar genes, the best approach with this algorithm would be to identify local, highly connected subnetworks of genes and perform biological experiments to determine if these gene interactions are meaningful.

One of the disadvantages with the current implementation of DBN is that it detects indirect regulatory relationships. For example, if A inhibits B, which inhibits C, then an edge may exist between A and C in the final DBN. This problem may be solved with the incorporation of sequence motif information proposed by Yoshinori et al. [19].

Another way to improve the DBmcmc algorithm is to modify it so that it may be able to learn from continuous data instead of discrete data. This should produce more accurate result and reduce the number of false edges in the final network. Alternatively, it will also be interesting to be able to use DBmcmc to analyze the other two rat CNS data to see if a similar network is produced as a final result. This would give a better evaluation regarding the consistency of the algorithm. Lastly, we can incorporate the extension of Markov Chain Monte carlo learning algorithm proposed by Friedman in 2003 [20], which searches and change the order of the sequential Markov Chain to allow for exploration of different parent-child relationships within a single simulation.
ACKNOWLEDGMENTS

This paper is a course project done under the instruction and help from Dr. Parvin Mousavi.

References


