The GlobalMIT Toolkit

for

Learning Optimal Dynamic Bayesian Network

User Guide

Release 1.0
Maintained by Vinh Nguyen

Monash University, Victoria, Australia.

Project members:

- Vinh Nguyen, Gippsland School of Information Technology, Monash University, Victoria, Australia.
- Madhu Chetty, Gippsland School of Information Technology, Monash University, Victoria, Australia.
- Pramod Wangikar, Chemical Engineering Department, Indian Institute of Technology Bombay, India.
- Ross Coppel, Faculty of Medicine, Nursing and Health Sciences, Monash University, Victoria, Australia.

First edition: 10 Feb 2011

Short contents

Contents

One

User Manual

1.1 INTRODUCTION

GlobalMIT is a Matlab/C++ toolkit for learning dynamic Bayesian network (DBN). It implements our polynomial time algorithm for learning the globally optimal dynamic Bayesian network structure using the Mutual Information Test (MIT) criterion, as presented in [VCWC11]. The DBN model assumed by GlobalMIT is the first-order Markov stationary DBN, in which both the structure of the network and the parameters characterizing it are assumed to remain unchanged over time, such as the one exemplified in Figure 1.1a. In this model, the value of a random variable at time $t+1$ is assumed to depend only on the value of its parents at time t.

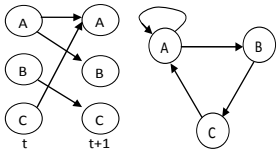

Figure 1.1: Dynamic Bayesian Network: (a) a 1st order Markov stationary DBN; (b) its equivalent folded network

1.2 INSTALLATION

The GlobalMIT Matlab toolbox is ready for use upon adding its container directory to the Matlab path environment variable (File → Set path). The toolbox contains all the functionalities.

For improved performance, the C++ search engine can be used. We provide precompiled version of the search engine (in Windows and Linux). Otherwise, the users can recompile the search engine. For example, in Linux, one may use:

```
g++ -g globalMIT.cpp -o globalMIT.exe
```

GlobalMIT can handle data that is a single time series, or a concatenation of multiple time series. Note that in the latter case, the time series need to be preprocessed for proper alignment (see Fig. 1.6).

Table 1.1: GlobalMIT Matlab toolbox main functions

File	Description
`compare_net.m`	Compare a network with a true network
`conditional_MI_DBN.m`	Calculate the conditional mutual information $I(X_i, X_j \vert \mathbf{Pa}_i)$
`conditional_MI_DBN_ab.m`	Calculate the conditional mutual information $I(X_i, X_j \vert \mathbf{Pa}_i)$ (multiple time series data)
`createDotGraphic.m`	Create dot graphic (require Graphviz)
`findLexicalIndex.m`	Find lexical order of a parent set
`globalMIT.m`	The Matlab GlobalMIT DBN search engine
`globalMIT_ab.m`	The Matlab GlobalMIT DBN search engine (multiple time series data)
`globalMIT_exe.m`	Matlab interface for the GlobalMIT C++ search engine
`globalMIT_exe_ab.m`	Matlab interface for the GlobalMIT C++ search engine (multiple time series data)
`multi_time_series_cat.m`	Concatenate multiple time series
`myDataMapping.m`	Map discrete data to a continuous value range
`myIntervalDiscretize.m`	Discretize data using equal bins
`score_MIT.m`	Get the MIT score for a DBN
`single_node_score_MIT.m`	Get the MIT score for a single node in a DBN
`writeGlobalMITfile.m`	C++ interface file
`writeGlobalMITfile_ab.m`	C++ interface file (multiple time series data)

Table 1.2: GlobalMIT C++ search engine

File	Description
`globalMIT.cpp`	global DBN search engine using the MIT criterion
`globalMIT_ab.cpp`	global DBN search engine using the MIT criterion (multiple time series data)

1.3 USAGE AND EXAMPLES

We demonstrate the use of GlobalMIT for DBN structure learning via a set of walk-through examples.

1.3.1 Single time series data

We first load the yeast data set [Hus03]:

```
load husmeier_yeast_100;
```

The data file contains a true network, and synthetic data generated from this network as described in [Hus03]. The synthetic data set is a single time series of 100 observations. We first get the MIT score of this true network, and visualize it using Graphviz, an open

source graph visualization software from AT&T Research[1]. The correct path to Graphviz execution file needs to be supplied in `createDotGraphic.m`. The resulting graph is displayed in Fig. 1.2.

```
s_true=score_MIT(a,true_net);
fprintf('Score of the true network: %f \n',s_true);
createDotGraphic(true_net,'True DBN');
```

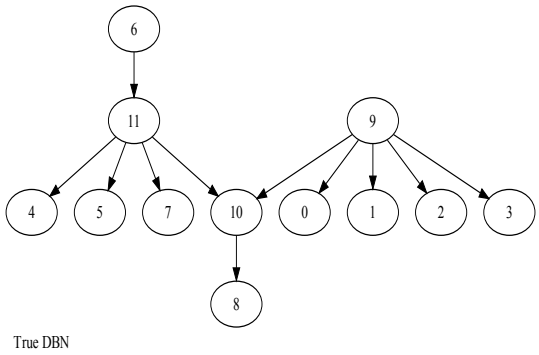

Figure 1.2: True yeast network.

This data set is binary, so we do not need to discretize the data. We use the Matlab GlobalMIT search engine to find the optimal DBN:

```
alpha=0.999;
allowSelfLoop=1;
[best_net]=globalMIT(data,alpha,allowSelfLoop);
createDotGraphic(best_net,'GlobalMIT DBN');
```

This function takes three parameters:

- `data`: a single time series data, with each column being a variable, and each row being an observation.
- `alpha`: the significance level for the mutual information test of independence.
- `allowSelfLoop`: allow the self regulated link (a link from a node to it-self) or not

It can be seen in Figure 1.3 that GlobalMIT finds the correct DBN as in Fig. 1.2.

On our Core 2 Duo PC, this operation takes:

```
Elapsed time is 16.539414 seconds.
```

[1] www.graphviz.org

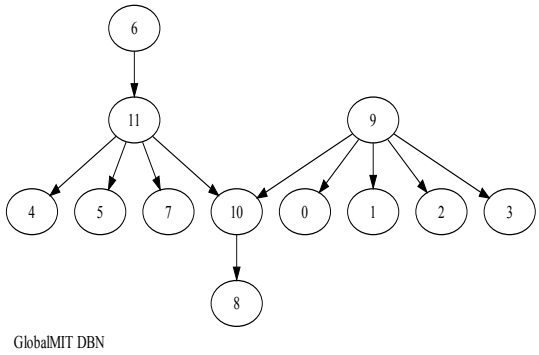

Figure 1.3: DBN found by globalMIT.

We now test the C++ version of globalMIT:

```
[best_net_exe,score,time]=globalMIT_exe(a,alpha,0);
createDotGraphic(best_net_exe,'GlobalMIT C++ DBN');
```

It can be verified that globalMIT C++ finds the network identical to globaMIT Matlab. On the same computer, globalMIT C++ takes only:

```
time =

    0.6262
```

second to complete the task.

1.3.2 Multiple time series data

This section illustrates the use of GlobalMIT on data composed of multiple time series. We first load the following data set:

```
clear clc;
load Yu_net_5;
n_state=3;
```

which is generated from Yu's net No. 1 [YSW+04], as demonstrated in Fig. 1.4. This network consists of 20 nodes, and operates according to the following linear dynamical system:

$$X_{t+1} - X_t = A(X_t - T) + \epsilon \tag{1.1}$$

with X denotes the expression profiles, A describes the strength of gene-gene regulations, T is the constitutive expression values, and ϵ simulates a uniform biological noise. The detailed parameters can be found in [YSW+04].

The data set here contains 3 time series, `a1, a2, a3`, each of length 33, generated by (1.1). We discretize each series into 3 states:

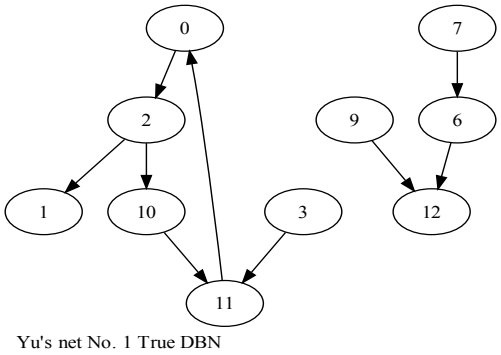

Figure 1.4: Yu's net 1.

```
a1= myIntervalDiscretize(a1,n_state);
a2= myIntervalDiscretize(a2,n_state);
a3= myIntervalDiscretize(a3,n_state);
```

We first take a single time series for analysis:

```
alpha=0.95;
tic;[best_net]=globalMIT(a1,alpha,1);toc
createDotGraphic(best_net,'GlobalMIT DBN');
compare_net(best_net,true_net,1)
```

Note that since the data is short, we lower `alpha` to 0.95. The network recovered is presented in Fig. 1.5. Note that there are lots of self regulated links in the discovered network. This is to be expected, since biological time series data are often smooth and have a high degree of auto-correlation and auto-mutual information at short lag. While these links might or might not be present in the ground-truth network, they are generally not very informative, as for most natural biological processes, the current state often dictates the state in the near future (unless the process evolves in a random-walk manner).

The quality metrics of the reconstructed network are (without taking the self-links into account):

```
Recall: 0.111111 ; Imprecision= 0.888889
```

which is low, since the data is too short. In reality, if multiple time-series are available, they can be concatenated to make a single time series. The data need to be preprocessed for proper alignment as illustrated in Fig 1.6.

```
[b,c]=multi_time_series_cat(a1,a2,a3);
alpha=0.95;
[best_net_ab]=globalMIT_ab(b,c,alpha,1);
```

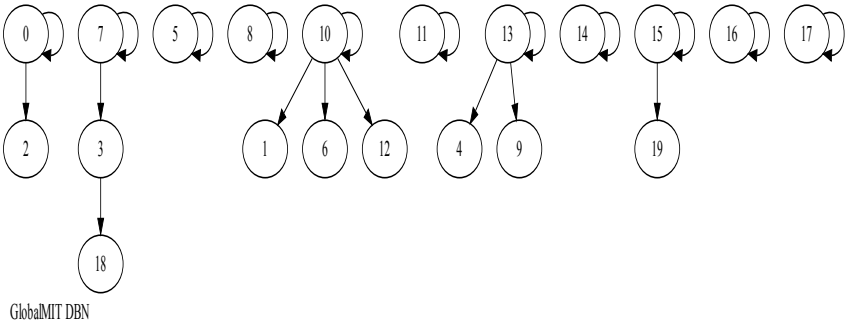

Figure 1.5: GlobalMIT on Yu's net 1, short data

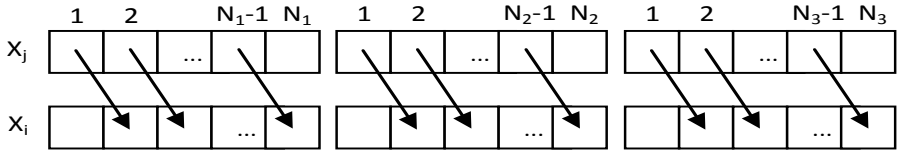

Figure 1.6: Multiple time series data alignment

```
createDotGraphic(best_net_ab,'GlobalMIT DBN')
compare_net(best_net_ab,true_net,1)
```

The newly discovered network is presented in Fig. 1.7.

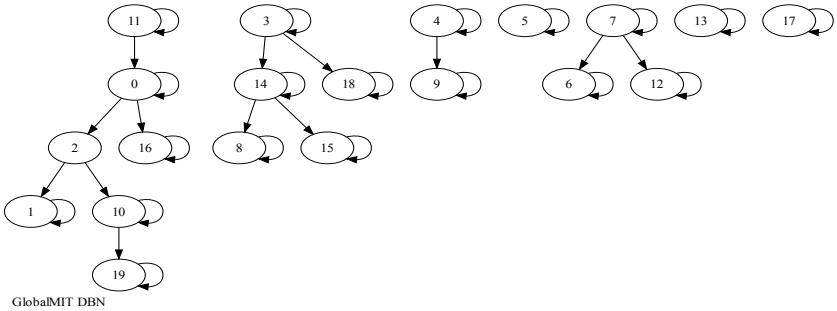

Figure 1.7: GlobalMIT on Yu's net 1, long data

With more data available, the network quality improves, with sensitivity (recall) increases and imprecision decreases:

```
Recall: 0.555556 ; Imprecision= 0.615385
```

1.3.3 Large data set

The worst case complexity of GlobalMIT is dependant upon the number of variables and the number of observations. Let us take a large data set, generated from Yu's network in the example above, with 20 variables and 2000 observations.

```
alpha=0.9999;
data= myIntervalDiscretize(data,n_state);
[best_net_exe,score,time]=globalMIT_exe(data,alpha,1);
createDotGraphic(best_net_exe,'GlobalMIT C++ DBN');
compare_net(best_net_exe,true_net,1)
```

Since the number of observations is large, we raise the significance level `alpha` to 0.9999. The Matlab version of globalMIT would take more than a day to analyze this data set. The C++ implementation takes only more than an hour.

The newly discovered network is presented in Fig. 1.8. The quality metrics are:

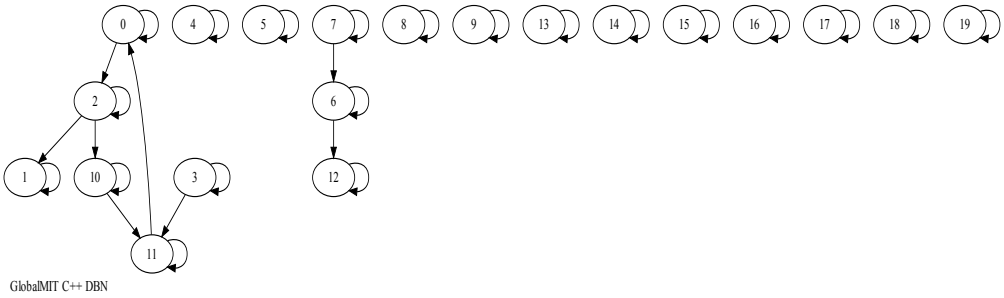

Figure 1.8: GlobalMIT on Yu's net 1, large data

```
Recall: 0.888889 ; Imprecision= 0.000000
```

1.3.4 Using GlobalMIT without Matlab

It is noted that GlobalMIT does not require Matlab, a non-free software, to carry out its core functionality, i.e., searching for the globally optimal DBN under MIT, thanks to the GlobalMIT C++ implementation. Note however that GlobalMIT C++ requires access to the inverse Chi square function. This function is available in the GNU scientific library. However, to keep portability, the current version of GlobalMIT C++ requires this information to be pre-provided in a file (`myChiValue.txt`, currently created by the

`writeGlobalMITfile.m` interface module). The user can use Octave, a freely available, Matlab-like software, which also provides the inverse Chi square function, to generate this file.

Two

Supplementary Material

In this section, we review the MIT score for learning BN, then adapts it to the DBN case. For a more complete treatment of the subject, readers are referred to [VCWC11].

2.1 The GlobalMIT Algorithm for Learning the Globally Optimal Dynamic Bayesian Network Structure

Briefly speaking, under MIT the goodness-of-fit of a network is measured by the total mutual information shared between each node and its parents, penalized by a term which quantifies the degree of statistical significance of this shared information. Let $\mathbf{X} = \{X_1, \ldots, X_n\}$ denote the set of n variables with corresponding $\{r_1, \ldots, r_n\}$ discrete states, D denote our data set of N observations, G be a DAG, and $\mathbf{Pa}_i = \{X_{i1}, \ldots, X_{is_i}\}$ be the set of parents of X_i in G with corresponding $\{r_{i1}, \ldots, r_{is_i}\}$ discrete states, $s_i = |\mathbf{Pa}_i|$, then the MIT score is defined as:

$$SS_{MIT}(G : D) = \sum_{\substack{i=1 \\ \mathbf{Pa}_i \neq \emptyset}}^{n} \{2N.I(X_i, \mathbf{Pa}_i) - \sum_{j=1}^{s_i} \chi_{\alpha, l_i \sigma_i(j)}\}$$

where $I(X_i, \mathbf{Pa}_i)$ is the mutual information between X_i and its parents as estimated from D. $\chi_{\alpha, l_{ij}}$ is the value such that $p(\chi^2(l_{ij}) \leq \chi_{\alpha, l_{ij}}) = \alpha$ (the Chi-square distribution at significance level $1 - \alpha$), and the term $l_i \sigma_i(j)$ is defined as:

$$l_i \sigma_i(j) = \begin{cases} (r_i - 1)(r_{i\sigma_i(j)} - 1) \prod_{k=1}^{j-1} r_{i\sigma_i(k)}, & j = 2 \ldots, s_i \\ (r_i - 1)(r_{i\sigma_i(j)} - 1), & j = 1 \end{cases}$$

where $\sigma_i = \{\sigma_i(1), \ldots, \sigma_i(s_i)\}$ is any permutation of the index set $\{1 \ldots s_i\}$ of $\mathbf{Pa}_i$, with the first variable having the greatest number of states, the second variable having the second largest number of states, and so on.

To make sense of this criterion, let us first point out that maximizing the first term in the score, $\sum_{\substack{i=1 \\ \mathbf{Pa}_i \neq \emptyset}}^{n} 2N.I(X_i, \mathbf{Pa}_i)$, can be shown to be equivalent to maximizing the log-likelihood criterion. Learning BN by using the maximum likelihood principle suffers from overfitting however, as the fully-connected network will always have the maximum likelihood. Likewise, for the MIT criterion, since the mutual information can always be increased by including additional variables to the parent set, i.e., $I(X_i, \mathbf{Pa}_i \cup X_j) \geq I(X_i, \mathbf{Pa}_i)$, the complete network will have the maximum total mutual information.

Thus, there is a need to penalize the complexity of the learned network. Penalizing the log-likelihood criterion with $-\frac{1}{2} C(G) \log(N)$ gives us the BIC/MDL criteria, while $-C(G)$

gives us the AIC criterion (where $C(G) = \sum_{i=1}^{n}(r_i - 1)\prod_{j=1}^{s_i} r_{ij}$ measures the network complexity). As for the MIT criterion, while the mutual information always increases when including additional variables to the parent set, the degree of statistical significance of this increment might become negligible as more and more variables are added. This significance degree can be quantified based on a classical result in information theory by [Kul68], which, in this context, can be stated as follows: under the hypothesis that X_i and X_j are conditionally independent given $\mathbf{Pa}_i$ is true, the statistics $2N.I(X_i, X_j|\mathbf{Pa}_i)$ approximates to a $\chi^2(l)$ distribution, with $l = (r_i - 1)(r_j - 1)q_i$ degree of freedom, and $q_i = 1$ if $\mathbf{Pa}_i = \emptyset$, otherwise q_i is total the number of state of $\mathbf{Pa}_i$, i.e., $q_i = \prod_{k=1}^{s_i} r_{ik}$. Now we can see that the second term in the MIT score penalizes the addition of more variables to the parent set. Roughly speaking, only variables that have the conditional mutual information shared with X_i given all the other variables in $\mathbf{Pa}_i$ that is higher than 100α percent of the MI values under the null hypothesis of independence can increase the score. For detailed motivations and derivation of this scoring metric as well as an extensive comparison with BIC/MDL and BD, we refer readers to [dC06].

Adapting MIT for DBN learning is rather straightforward. One just has to pay attention to the fact that the mutual information is now calculated between a parent set and its child, which should be 1-unit shifted in time, as required by the first-order Markov assumption, denoted by $X_i^{\overrightarrow{1}} = \{X_{i2}, X_{i3}, \ldots, X_{iN}\}$. As such, the number of "effective" observations, denoted by N_e, for DBN is now only $N - 1$. This is demonstrated in Figure 2.1. The MIT score for DBN should be calculated as:

$$S'_{MIT}(G : D) = \sum_{\substack{i=1 \\ \mathbf{Pa}_i \neq \emptyset}}^{n} \{2N_e.I(X_i^{\overrightarrow{1}}, \mathbf{Pa}_i) - \sum_{j=1}^{s_i} \chi_{\alpha, l_i \sigma_i(j)}\}$$

Similarly, when the data is composed of N_t separate time-series, the number of effective observations is only $N_e = N - N_t$.

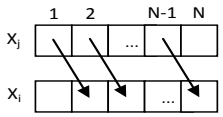

Figure 2.1: Data alignment for dynamic Bayesian network with an edge $X_j \rightarrow X_i$. The "effective" number of observations is now only $N - 1$.

2.2 Optimal Dynamic Bayesian Network Structure Learning in Polynomial Time with MIT

In this section, we show that learning the globally optimal DBN with MIT can be achieved in polynomial time. Our development is based on a recent result by [Doj06], which states that under several mild assumptions, there exists a polynomial worst-case time complexity algorithm for learning the optimal DBN with the MDL and BDe scoring metrics. Specifically, the 4 assumptions that Dojer considered are:

Assumption 1. *(acyclicity) There is no need to examine the acyclicity of the graph.*

Assumption 2. *(additivity)* $S(G:D) = \sum_{i=1}^{n} s(X_i, \boldsymbol{Pa}_i : D|_{X_i \cup \boldsymbol{Pa}_i})$ *where* $D|_{X_i \cup \boldsymbol{Pa}_i}$ *denotes the restriction of* D *to the values of the members of* $X_i \cup \boldsymbol{Pa}_i$.

To simplify notation, we write $s(\mathbf{Pa}_i)$ for $s(X_i, \mathbf{Pa}_i : D|_{X_i \cup \mathbf{Pa}_i})$.

Assumption 3. *(splitting)* $s(\boldsymbol{Pa}_i) = g(\boldsymbol{Pa}_i) + d(\boldsymbol{Pa}_i)$ *for some non-negative functions g,d satisfying* $\boldsymbol{Pa}_i \subseteq \boldsymbol{Pa}'_i \Rightarrow g(\boldsymbol{Pa}_i) \leq g(\boldsymbol{Pa}'_i)$

Assumption 4. *(uniformity)* $|\boldsymbol{Pa}_i| = |\boldsymbol{Pa}'_i| \Rightarrow g(\boldsymbol{Pa}_i) = g(\boldsymbol{Pa}'_i)$

Assumption 1 is valid for DBN in general. For the first-order Markov DBN that we are considering in this paper, since the graph is bipartite, with edges directing only forward in time (Fig. 1.1a), acyclicity is automatically satisfied. Assumption 2 simply states that the scoring function decomposes over the variables, which is satisfied by most scoring metrics such as BIC/MDL, BD and also clearly by MIT. Together with assumption 1, this assumption allows us to compute the parents set of each variable independently. Assumption 3 requires the scoring function to decompose into two components: d evaluating the accuracy of representing the distribution underlying the data by the network, and g measuring its complexity. Furthermore, g is required to be a monotonically non-decreasing function in the cardinality of $\mathbf{Pa}_i$ (assumption 4), i.e., the network gets more complex as more variables are added to the parent sets.

We note that unlike MIT in its original form that we have considered above, where better networks have higher scores, for the score considered by Dojer, lower scored networks are better. And thus the corresponding optimization must be cast as a score minimization problem. We now consider a variant of MIT as follows:

$$S_{MIT}(G:D) = \sum_{i=1}^{n} 2N_e.I(X_i^{\overrightarrow{1}}, \mathbf{X}) - S'_{MIT}(G:D) \tag{2.1}$$

which admits the following decomposition over each variable (with the convention of $I(X_i, \emptyset) = 0$):

$$\begin{aligned} s_{MIT}(\mathbf{Pa}_i) &= d_{MIT}(\mathbf{Pa}_i) + g_{MIT}(\mathbf{Pa}_i) \\ d_{MIT}(\mathbf{Pa}_i) &= 2N_e.I(X_i^{\overrightarrow{1}}, \mathbf{X}) - 2N_e.I(X_i^{\overrightarrow{1}}, \mathbf{Pa}_i) \\ g_{MIT}(\mathbf{Pa}_i) &= \sum_{j=1}^{s_i} \chi_{\alpha, l_i \sigma_i(j)} \end{aligned}$$

Roughly speaking, d_{MIT} measures the "error" of representing the joint distribution underlying D by G, while g_{MIT} measures the complexity of this representation. We state the following results:

Proposition 1. *The problem of* S'_{MIT} *maximization is equivalent to the problem of* S_{MIT} *minimization.*

Proof. Obvious, since $\sum_{i=1}^{n} 2N_e.I(X_i^{\overrightarrow{1}}, \mathbf{X}) = const$. □

Proposition 2. d_{MIT}, g_{MIT} *satisfy assumption 3*

Proof. Trivial. $d_{MIT} \geq 0$ since of all parent sets $\mathbf{Pa}_i$, $\mathbf{X}$ has the maximum mutual information with $X_i^{\overrightarrow{1}}$. And since the support of the Chi-square distribution is $\mathbb{R}^+$, i.e., $\chi_{\alpha,\cdot} \geq 0$, therefore $\mathbf{Pa}_i \subseteq \mathbf{Pa}'_i \Rightarrow 0 \leq g_{MIT}(\mathbf{Pa}_i) \leq g_{MIT}(\mathbf{Pa}'_i)$. □

Unfortunately, g_{MIT} does not satisfy assumption 4. However, for many applications, if all the variables have the same number of states then it can be shown that g_{MIT} satisfies assumption 4.

Assumption 5. *(variable uniformity) All variables in $\mathbf{X}$ have the same number of discrete states k.*

Proposition 3. *Under the assumption of variable uniformity, g_{MIT} satisfies assumption 4.*

Proof. It can be seen that if $|\mathbf{Pa}_i| = |\mathbf{Pa}'_i| = s_i$, then $g_{MIT}(\mathbf{Pa}_i) = g_{MIT}(\mathbf{Pa}'_i) = \sum_{j=1}^{s_i} \chi_{\alpha,(k-1)^2 k^{j-1}}$. □

Since $g_{MIT}(\mathbf{Pa}_i)$ is the same for all parent sets of the same cardinality, we can write $g_{MIT}(|\mathbf{Pa}_i|)$ in place of $g_{MIT}(\mathbf{Pa}_i)$. With assumptions 1-5 satisfied, we can employ the following Algorithm 1, named globalMIT, to find the globally optimal DBN with MIT, i.e., the one with the minimal S_{MIT} score.

Algorithm 1 globalMIT : Optimal DBN with MIT

```
Pa_i := ∅
for p = 1 to n do
  If g_MIT(p) ≥ s_MIT(Pa_i) then return Pa_i; Stop.
  P = arg min_{Y⊆X:|Y|=p} s_MIT(Y)
  If s_MIT(P) < s_MIT(Pa_i) then Pa_i := P.
end for
```

Theorem 1. *Under assumptions 1-5, globalMIT applied to each variable in $\mathbf{X}$ finds a globally optimal DBN under the MIT scoring metric.*

Proof. The key insight here is that once a parent set grows to a certain extent, its complexity alone surpasses the total score of a previously found sub-optimal parent set. In fact, all the remaining potential parent sets $\mathbf{P}$ omitted by the algorithm have a total score higher than the current best score, i.e., $s_{MIT}(\mathbf{P}) \geq g_{MIT}(|\mathbf{P}|) \geq s_{MIT}(\mathbf{Pa}_i)$, where $\mathbf{Pa}_i$ is the last sub-optimal parent set found. □

We note that the terms $2N_e.I(X_i^{\overrightarrow{1}}, \mathbf{X})$ in the S_{MIT} score in (2.1) do not play any essential role, since they are all constant and would not affect the outcome of our optimization problem. Knowing their exact value is however, necessary for the stopping criterion in Algorithm 1, and also for constructing its complexity bound, as we shall do shortly. Unfortunately, calculating $I(X_i^{\overrightarrow{1}}, \mathbf{X})$ is by itself a hard problem, requiring $O(k^{n+1})$ space and time in general. However, for our purpose, since the only requirement for d_{MIT} is that it must be non-negative, it is sufficient to use an upper bound of $I(X_i^{\overrightarrow{1}}, \mathbf{X})$. A fundamental

property of the mutual information states that $I(\mathbf{X}, \mathbf{Y}) \leq \min\{H(\mathbf{X}), H(\mathbf{Y})\}$, i.e., mutual information is bounded by the corresponding entropies. We therefore have:

$$2N_e.I(X_i^{\overrightarrow{1}}, \mathbf{X}) \leq 2N_e.H(X_i^{\overrightarrow{1}}),$$

where $H(X_i^{\overrightarrow{1}})$ can be estimated straightforwardly from the data. Or else, we can use an a priory fixed upper bound for all $H(X_i^{\overrightarrow{1}})$, that is $\log k$, then:

$$2N_e.I(X_i^{\overrightarrow{1}}, \mathbf{X}) \leq 2N_e.\log k.$$

Using these bounds, we obtain the following more practical versions of d_{MIT}:

$$\begin{aligned} d'_{MIT}(\mathbf{Pa}_i) &= 2N_e.H(X_i^{\overrightarrow{1}}) - 2N_e.I(X_i^{\overrightarrow{1}}, \mathbf{Pa}_i) \\ d''_{MIT}(\mathbf{Pa}_i) &= 2N_e.\log k - 2N_e.I(X_i^{\overrightarrow{1}}, \mathbf{Pa}_i) \end{aligned}$$

It is straightforward to show that Algorithm 1 and Theorem 1 are still valid when d'_{MIT} or d''_{MIT} are used in place of d_{MIT}.

2.2.1 Complexity bound

Theorem 2. *globalMIT admits a polynomial worst-case time complexity in the number of variables.*

Proof. Our aim is to find a number p^* satisfying $g_{MIT}(p^*) \geq s_{MIT}(\emptyset)$. Clearly, there is no need to examine any parent set of cardinality p^* and over. In the worse case, our algorithm will have to examine all the possible parent sets of cardinality from 1 to $p^* - 1$. We have:

$$\begin{aligned} g_{MIT}(p^*) &\geq s_{MIT}(\emptyset) \\ \Leftrightarrow \sum_{j=1}^{p^*} \chi_{\alpha, l_i \sigma_i(j)} &\geq d_{MIT}(\emptyset) = 2N_e.I(X_i^{\overrightarrow{1}}, \mathbf{X}). \end{aligned}$$

As discussed above, since calculating d_{MIT} is not convenient, we use d'_{MIT} and d''_{MIT} instead. With d'_{MIT} p^* can be found as:

$$p^* = \underset{\sum_{j=1}^{p} \chi_{\alpha, l_i \sigma_i(j)} \geq 2N_e.H(X_i^{\overrightarrow{1}})}{\arg\min} p, \tag{2.2}$$

while with d''_{MIT}:

$$p^* = \underset{\sum_{j=1}^{p} \chi_{\alpha, l_i \sigma_i(j)} \geq 2N_e.\log k}{\arg\min} p. \tag{2.3}$$

Since there are $O(n^{p^*})$ subsets with at most p^* parents, and each set of parents can be scored in polynomial time, globalMIT admits an overall polynomial worst-case time complexity. □

We now give some examples to demonstrate the practicability of Theorem 2.

Example 1: Consider a gene regulatory network reconstruction problem, where each gene has been discretized to $k = 3$ states, corresponding to up, down and regular gene

expression. With the level of significance α set to 0.999 as recommended in [dC06], we have:

$$\begin{aligned}
g_{MIT}(1) &= \chi_{0.999,4} &&= 18.47\\
g_{MIT}(2) &= g_{MIT}(1) + \chi_{0.999,12} &&= 51.37\\
g_{MIT}(3) &= g_{MIT}(2) + \chi_{0.999,36} &&= 119.35\\
g_{MIT}(4) &= g_{MIT}(3) + \chi_{0.999,108} &&= 278.51\\
g_{MIT}(5) &= g_{MIT}(4) + \chi_{0.999,324} &&= 686.92\\
g_{MIT}(6) &= g_{MIT}(5) + \chi_{0.999,972} &&= 1800.9\\
g_{MIT}(7) &= g_{MIT}(6) + \chi_{0.999,2916} &&= 4958.6\\
g_{MIT}(8) &= g_{MIT}(7) + \chi_{0.999,8748} &&= 14121\\
g_{MIT}(9) &= g_{MIT}(8) + \chi_{0.999,26244} &&= 41079\\
g_{MIT}(10) &= g_{MIT}(9) + \chi_{0.999,78732} &&= 121042\\
&\dots
\end{aligned}$$

Consider a data set of $N = 12$ observations, which is the popular length of microarray time-series experiments (in fact N often ranges within $4 - 15$), then $d''_{MIT}(\emptyset) = 2(N - 1)\log k = 24.16$. Observing that $g_{MIT}(2) > d''_{MIT}(\emptyset)$, then $p^* = 2$ and we do not have to consider any parent sets of 2 variables or more.

Let us compare this bound with those of the algorithms for learning the globally optimal DBN under the BIC/MDL and BDe scoring metrics. For BIC/MDL, p^*_{MDL} is given by $\lceil \log_k N \rceil$, while for BDe, $p^*_{BDe} = \lceil N \log_{\gamma^{-1}} k \rceil$, where the distribution $P(G) \propto \gamma^{\sum |\mathbf{Pa}_i|}$, with a penalty parameter $0 < \gamma < 1$, is used as a prior over the network structures [Doj06]. In this case, $p^*_{MDL} = 3$. If we choose $\log \gamma^{-1} = 1$ then $p^*_{BDe} = \lceil N \log k \rceil = 14$. In general, p^*_{BDe} scales linearly with the number of data items N, making its value less of practical interest, even for small data sets.

Example 2: Since the number of observations in a single microarray time-series experiment is often limited, it is a popular practice to concatenate several time-series to obtain a larger data set for analysis. Let us merge $N_t = 10$ data sets, each with 12 observations, then $N_e = N - N_t = 120 - 10 = 110$. For this combined data set, $g_{MIT}(4) > d''_{MIT}(\emptyset) = 2N_e \log k = 241.69 \Rightarrow p^* = 4$, thus there is no need to consider any parent set of more than 3 variables. For comparison, we have $p^*_{MDL} = 5$, and $p^*_{BDe} = 132$ with $\log \gamma^{-1} = 1$.

Of course, this analysis only gives us the worst-case time complexity. In practice, the execution of Algorithm 1 can often be much shorter, since $s_{MIT}(\mathbf{Pa}_i)$ is often much greater than $s_{MIT}(\emptyset)$. This observation also applies for the global algorithms based on the BIC/MDL and BDe scoring metrics.

Even though Algorithm 1 admits a polynomial time complexity, exhaustive search for the optimal parent sets over all the subsets of $\mathbf{X}$ with at most $p^* - 1$ elements may still be extremely time consuming, especially when the number of variable n is large. In such cases, even if heuristic search algorithms were employed, our analysis gives a theoretical guidance and justification for setting the so-called "max-fan-in" parameter, which dictates the maximum number of parents allowed for each node, as found popular in many soft-

wares for DBN learning. There seems to be no systematic rules for setting this parameter in the BN literature in our observation.

Example 3: Let's consider some large scale data sets, with $k = 3, \alpha = 0.999$ and a set of $N = 10000$ observations, then $p^* = 9$. The max-fan-in parameter can then be set to 8. For comparison, we have $p^*_{MDL} = 9$ and $p^*_{BDE} = 10987$ with $\log \gamma^{-1} = 1$.

2.2.2 Efficient Implementation for globalMIT

The search procedure involves examining all potential parent sets of increasing cardinality. The following decomposition property of the mutual information is handy when it comes to design an efficient implementation for globalMIT:

$$I(X_i, \mathbf{Pa}_i \cup X_j) = I(X_i, \mathbf{Pa}_i) + I(X_i, X_j | \mathbf{Pa}_i)$$

This implies that the mutual information can be computed incrementally, and suggests that, for efficiency, the computed mutual information values should be cached to avoid redundant computations, subject to memory availability.

Bibliography

[dC06] Luis M. de Campos. A scoring function for learning bayesian networks based on mutual information and conditional independence tests. *J. Mach. Learn. Res.*, 7:2149–2187, December 2006.

[Doj06] Norbert Dojer. Learning Bayesian Networks Does Not Have to Be NP-Hard. In *Proceedings of International Symposium on Mathematical Foundations of Computer Science*, pages 305–314, 2006.

[Hus03] Dirk Husmeier. Sensitivity and specificity of inferring genetic regulatory interactions from microarray experiments with dynamic Bayesian networks. *Bioinformatics*, 19(17):2271–2282, 2003.

[Kul68] Solomon Kullback. *Information Theory and Statistics*. Dover publications, 1968.

[VCWC11] Nguyen Xuan Vinh, Madhu Chetty, Pramod Wangikar, and Ross Coppel. Learning globally optimal dynamic bayesian network with an information theoretic criterion in polynomial time. submitted, 2011.

[YSW+04] Jing Yu, V. Anne Smith, Paul P. Wang, Alexander J. Hartemink, and Erich D. Jarvis. Advances to Bayesian network inference for generating causal networks from observational biological data. *Bioinformatics*, 20(18):3594–3603, 2004.