MEDINFO 2004
M. Fieschi et al. (Eds)
Amsterdam: IOS Press

A Novel Algorithm for Scalable and Accurate Bayesian Network Learning

Laura E. Brown, Ioannis Tsamardinos, Constantin F. Aliferis

Discovery Systems Laboratory, Department of Biomedical Informatics, Vanderbilt University, USA

Abstract

*Bayesian Networks (BN) is a knowledge representation formalism that has been proven to be valuable in biomedicine for constructing decision support systems and for generating causal hypotheses from data. Given the emergence of datasets in medicine with thousands of variables and that current algorithms do not scale more than a few hundred variables in practical domains, new efficient and accurate algorithms are needed to learn high quality BNs from data. We present a new algorithm called Max-Min Hill-Climbing (**MMHC**) that builds upon and improves the Sparse Candidate (**SC**) algorithm; a state-of-the-art algorithm that scales up to datasets involving hundreds of variables provided the generating networks are sparse. Compared to the **SC,** on a number of datasets from medicine and biology, (a) **MMHC** discovers BNs that are structurally closer to the data-generating BN, (b) the discovered networks are more probable given the data, (c) **MMHC** is computationally more efficient and scalable than **SC**, and (d) the generating networks are not required to be uniformly sparse nor is the user of **MMHC** required to guess correctly the network connectivity.*

Keywords:

Expert Systems, Causal Discovery, Artificial Intelligence, Algorithms, Bayesian Analysis, Bayesian Networks.

Introduction and Background

Bayesian Networks is a formalization that has proved to be a useful and important tool in biomedicine with example applications in decision support systems [1-3], information retrieval [4], discovery of gene regulatory pathways [5], and natural language processing [6]. Automatically *learning BNs* from observational data has been an area of intense research for more than a decade yielding practical algorithms and tools [7-9]. Using such algorithms one can construct a decision support system from data or infer plausible causal relations.

There are two main approaches for learning BNs from data. The search-and-score approach attempts to identify the most probable a posteriori network N given the data D and prior knowledge K, i.e., find the $argmax_N P(N|D,K)$. Depending on the assumptions made, maximizing this probability $P(N|D,K)$ corresponds to maximizing a score function, such as the BDeu scoring metric [8]. Algorithms in this category search the space of all possible structures for the one that maximizes the score using greedy, local, or some other search algorithm. The second approach for learning BNs is constraint-based [7]. Algorithms following this approach estimate from the data whether certain conditional independences between the variables hold. Typically, this estimation is performed using statistical or information theoretic measures. The results are propagated and the networks that are inconsistent with them are eliminated from further consideration. A sound strategy for performing conditional independence tests ultimately retains (and returns) only the statistically equivalent networks consistent with the tests.

One of the current, most prominent algorithms for learning BNs, and the first to be successfully applied to datasets with several hundred variables, is the Sparse Candidate (**SC**) [5, 9]. **SC** requires the network to be sparse and the user to have an estimate of the degree of connectivity.

The novel idea of the **SC** is to constrain the search: each variable X is constrained to have parents *only* from within a predetermined candidate-parents set $C(X)$ of size at most k, where k is defined by the user. Initially, the candidate sets are heuristically estimated, and then hill-climbing (or some other search method) is used to identify a network that (locally) maximizes the score metric. Subsequently, the candidate sets are re-estimated and another hill-climbing search round is initiated. We will call the cycle of candidate sets estimation and hill-climbing an *iteration*. **SC** iterates until there is no change in the candidate sets or a given number of iterations have passed with no improvement in the network score.

There are two main problems with the **SC**. The first is that the estimation of the candidate sets is heuristic (may not identify the true set of parents), and it may take a number of iterations to converge to an approximation of the true set of parents. The second problem with the **SC** is that the user has to guess the parameter k, i.e., the maximum number of parents allowed for any node in the network. If the user overestimates k the algorithm will take unnecessarily long to finish and *may even be rendered intractable* for large datasets. If the user underestimates k there is a risk of discovering a suboptimal network. As we show in our experiments, if a node has $m>k$ parents, then not only the node will be missing $m-k$ parents in the resulting network, but *the errors may propagate to the rest of the network*.

In this paper, we present a new algorithm called Max-Min Hill-Climbing (**MMHC**) that alleviates both of these problems. **MMHC** uses the recently introduced constraint-based algorithm **MMPC** (Max-Min Parents and Children) [10] to estimate the candidate sets. Given a target variable T, **MMPC** soundly (provided adequate sample) and efficiently identifies a superset of the true parents set (it returns the set of parents and children). This has three ramifications. First, the user does not have to

guess the parameter k: it is set automatically and individually for each variable. Second, **MMPC** is more accurate than the current **SC** heuristics for estimating the candidate sets and so the networks discovered are *structurally closer* to the true networks and have a *higher score* than the ones returned by **SC**. Third, **MMHC** is so accurate in estimating the candidate sets that only one iteration is required rendering **MMHC** less computationally intensive than **SC**.

Methods

The Sparse Candidate Algorithm (SC): The **SC** first estimates for each variable X, a candidate set $C(X)$. It then performs a hill-climbing search in the space of all BNs starting from the empty network (i.e., the one with no edges) and using the operators *add_edge, delete_edge, reverse_edge*. The operator that results in the network with the highest score is applied on the current network and the search continues recursively. An important detail is that an edge $Y \rightarrow X$ is added only if $Y \in C(X)$.

Once hill-climbing reaches a local maximum, the candidate sets are re-estimated and hill-climbing search is performed again. The original **SC** paper [9] provides several methods for re-estimating the candidate sets $C(X)$. Suppose in iteration i the parents of variable X_m in the current network are $Pa(X_m)$ and we want to compute the candidate parents set $C(X_m)$ in iteration $i+1$. All other variables X_n are sorted according to how much they increase the Bayesian score of the network when added as parents of X_m to the existing parents. The top-rated variables X_n are selected to be added to the current candidate-parents set of X_m until $C(X_m)$ reaches size of k. Two other heuristics for ordering variables X_n for inclusion in $C(X_m)$ are the mutual information between X_n and X_m conditioned on $Pa(X_m)$ and the Kullback-Leibler divergence between the current network and the network with X_n added as a parent of X_m.

The Max-Min Hill Climbing Algorithm (MMHC): MMHC is shown in Figure 1. It takes as input an array of data D and returns a BN with high score. **MMHC** first identifies the candidate sets for each variable X by calling **MMPC** and then performs a hill-climbing search starting from the empty network. The real work in **MMHC** is performed by **MMPC**. **MMPC**, given data and a target variable T, returns the parents and children of T, denoted as $PC(T)$. Notice that there may be many BNs that represent the joint distribution of the data. The question is which of all those BNs parents and children set does **MMPC** return? It is proved in [10] that the set of parents and children is unique in all BN faithfully (i.e., encoding all dependencies and independencies) representing the joint distribution. **MMPC** provably returns this unique set in the sample limit, or an approximation for finite sample. Notice that, a node may be a parent in one BN and a child in another; however, the union set of the parents and children will be unique in all faithful BNs.

MMPC is based on tests of conditional independence and measures of association. The implementation of conditional tests of independence is based on the χ^2 test and is explained in detail in [7]. The standard 5% threshold on p-values was used to reject independency and accept dependency. In a BN faithful to its joint distribution, a variable X has an edge to or from T if and only if it is conditionally dependent with T given any other subset $\boldsymbol{Z}$ [7].

MMPC provides an efficient way of testing whether the above condition holds. Thus, if **MMPC** discovers that conditioned on a set of variables X becomes independent of T, then X does not belong in the candidate-parents set $C(T)$.

MMHC(Data D)
"Returns a BN with high score (a posteriori probability given the data)"
1. For all variables X, Set $C(X)$ = **MMPC**(X, D)
2. Starting from an empty BN perform greedy hill-climbing with operators *add_edge, ,delete_edge, reverse_edge*. Only try operator *add_edge* $Y \rightarrow X$ if $Y \in C(X)$

MMPC(Target T ; Data D)
"Returns the parents and children of T" (Details in [10])
Phase I: Forward
Start with an empty $PC(T)$. $PC(T)=PC(T) \cup X$, where X is the variable that maximizes a heuristic function based on tests of conditional independence. Continue, until all remaining variables are conditionally independent of T given some subset of $PC(T)$.
Phase II: Backwards
Remove from $PC(T)$ any variable independent of T, conditioned on some subset of $PC(T)$. Return $PC(T)$.

*Figure 1 - Sketch of Algorithms **MMHC** and **MMPC***

MMPC discovers the $PC(T)$ using a two-phase scheme. In phase I, the forward phase, variables enter sequentially $PC(T)$ by use of a heuristic function that *selects the variable that maximizes the association with T conditioned on the subset of the current estimate of PC(T) that minimizes that association* (hence the name of the algorithm). The heuristic is **admissible** (presented with all details in [10]) in the sense that all variables with an edge to or from T and possibly more will enter $PC(T)$. In phase II, **MMPC** removes all false positives that entered in the first phase

Table 1: Description of Bayesian Networks

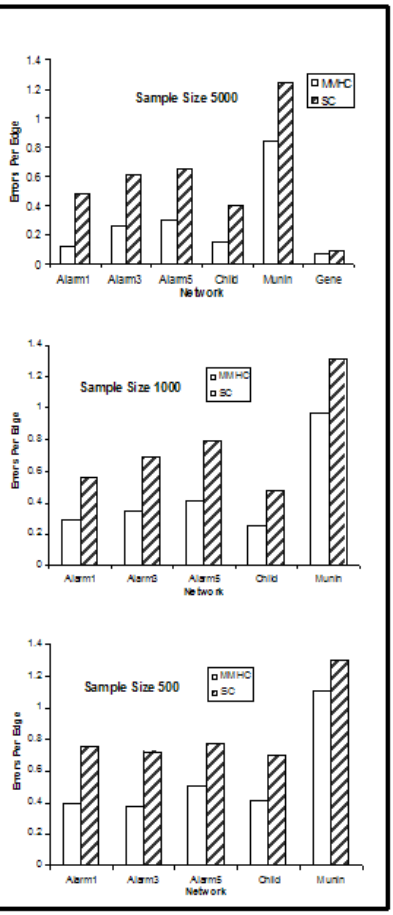
Name	# Vars	# Edges	Max in-Degree	Max Domain
Alarm	37	46	4	4
Alarm3	111	149	4	4
Alarm5	185	265	4	4
Child	20	25	2	6
Munin	189	282	3	21
Gene	801	972	4	5

Algorithms: For all subsequent experiments we used the **SC**'s authors' implementation of **SC** downloaded from http://www.cs.huji.ac.il/~nirf/LibB.html and our implementation in Matlab 6.5 of **MMHC**. We used the Bayesian scoring heuristic (with equivalent sample size 10 as suggested in [9]) because it slightly outperformed the other two in [9] and it is the only one used in all subsequent publications using **SC**.

Datasets: To test our hypotheses we used several BNs from medicine and biology that are freely available. These are the (i) ***Alarm*** [1], a BN in the heart of an expert system for monitoring

Table 2: Max-Min Hill Climbing vs Sparse Candidate

		BDeu Score		Structural Errors		Time in Seconds			# Iterations
	Network	MM-HC	Best of SC	MM-HC	Best of SC	MM-HC	SC k=5	SC k=10	Min of SC
Sample 5000	Alarm	**-17.6**	-17.8	**18.4**	34.8	**5.8**	8.2	78.4	3.4
	Alarm3	**-59.7**	-60.3	**56.2**	107	**31.1**	175.2	272.3	4
	Alarm5	**-100.7**	-102.1	**133.8**	205	**77.6**	779.8	874.3	4.8
	Child	**-19.1**	-21.6	**10.4**	17.4	6.7	**2.2**	26.8	2
	Munin	-91.1	**-91**	**313.6**	367	4.4K	**1.3K**	N/A	5
Sample 1000	Alarm	**-16**	-16.1	**13.2**	26	**6.9**	13.2	79.8	4
	Alarm3	**-55.7**	-56.3	**52.6**	102.7	**39.6**	306	389	3.4
	Alarm5	**-94.2**	-95.7	**109.4**	210.3	**92.6**	1.3K	1.5K	5
	Child	**-18.3**	-21.4	**6.4**	11.8	10.2	**3.4**	32.2	2.4
	Munin	**-78.7**	-80.4	**272.8**	371.8	4.6K	**2.7K**	N/A	5.6
Sample 500	Alarm	**-14.2**	-14.3	**5.4**	22.2	**17**	42.4	110	3
	Alarm3	**-51**	-51.6	**39**	91	**92.5**	1.4K	1416	3.7
	Alarm5	**-86.7**	-87.5	**78**	172.3	**222.2**	6.6K	6.3K	4
	Child	**-17.7**	-21	**3.8**	10	47.5	**13**	43.6	3
	Munin	**-64.7**	-66.2	**238**	349.8	**4.9K**	12K	N/A	4.6
	Gene	**-634.8**	-640.7	**62.5**	93	**14K**	682K	N/A	3



patients in intensive care, (ii) ***Child*** [2], a network used for the diagnosis of "blue babies", i.e., infants born with a congenital heart defect that causes a bluish coloration of the skin as a result of cyanosis, and (iii) ***Munin*** [3], a BN used in a electromyography assistant application.

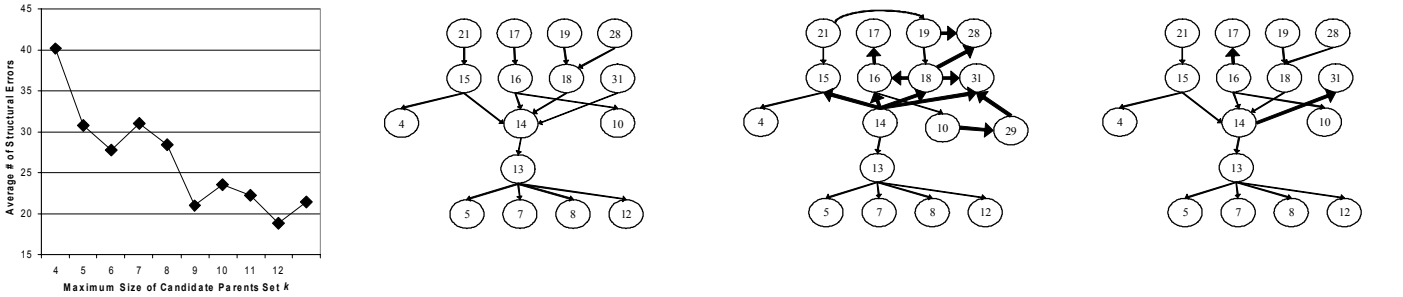
We also used a fourth BN modeling gene regulatory pathways that we named ***Gene***. Gene was constructed using the same procedure as in [9] for the evaluation of the **SC**: the gene expression microarray data produced in [11], involving 800 cell-cycle regulated genes over 76 experiments were discretized as in [9] and then fed to **SC** using the Bayesian scoring heuristic with *k=5* to output Gene.

Finally, we created two ***synthetic networks*** varying in a controlled manner the number of variables so as to test the scalability of the algorithms. We used a method first introduced in [10] that tiles and inter-connects several copies of the Alarm (or any other network) to create a network with the same structural and distributional properties. The structural properties are maintained in the tiled network by only adding a few inter-connecting edges. The distributional properties are maintained by imposing the constraint that the marginal joint of variables in the tiles is the same as the original Alarm. This is achieved by solving a quadratic system of equations to specially arrange the probability tables of the new edges. The details are explained in [12]. Specifically, we created versions of Alarm with 3 and 5 tiles named ***Alarm3*** and ***Alarm5***. The characteristic properties of all the networks used in our experiments are in Table 1. We note that there are several other BNs available for machine learning experimentation in the public domain. However, they are distinctly non-biomedical in nature and thus were excluded from the present experiments.

Measures of Performance: For the evaluation of the algorithms we report two measures of comparison. The first is the ***BDeu score*** [8] that corresponds to the logarithm of the posterior probability of the network given the data: the closer to zero a score is, the higher the probability. The score should correspond to the ability of a network to capture the joint probability of the data and predict new instances, and thus is directly related to the ***inference capabilities*** of the induced network. The second measure of comparison is the ***total number of structural errors*** made by each algorithm, i.e., the number of incorrectly added, deleted, or reversed edges. This measure is of particular interest when the algorithms are used for ***causal discovery***.

Results and Interpretation

We sampled five datasets per number of training instances in the set {500, 1000, 5000} per network of Table 1. We then applied **SC** with *k=5* and *k=10* and **MMHC** and report the average results over the five samplings in Table 2. An exception to the above is that due to high computation time required, only sample size of 5000 was tried for the Gene network; also, we were not able to run **SC** with *k=10* for Munin and Gene because the **SC**'s authors' implementation would run out of memory (even with 2GB of memory). In all networks the number of parents of all nodes is less or equal to 4 so the values *k=5* and *k=10* should be



(a) (b) True structure around node 14 (c) Structure discovered for *k=8* (d) Structure discovered for *k=9*

Figure 2 - A representative example

reasonable (*notice that, in practice where the true structure is unknown the user has no principled way for selecting k*).

The columns named "Best" and "Min" of **SC** have the best and minimum values respectively returned by **SC** for *k=5* and *k=10*. The numbers in boldface indicate the winning algorithm for each performance measure, network, and sample size.

MMHC vs SC in terms of BN learning. From Table 2 (BDeu scoring column) it is evident that **MMHC** outperforms **SC** in terms of BDeu score in all cases, except for the Munin network and sample size equal to 500. Notice that the BDeu score is logarithmic, where even small score differences imply an exponential difference in the probability of the discovered networks. For example, for Munin and 5000 sample size, an absolute difference of 1.5 in the score in favor of **MMHC** implies the network returned by **MMHC** is 2.8 times more probable than the one returned by **SC**. In terms of structural differences of the induced networks from the true structures, **MMHC** greatly outperforms **SC** (Table 2, Structural Errors column). Surprisingly, **MMHC** also outperforms **SC** even on the *Gene* network, which was initially generated by the latter algorithm. The graphs in the right of Table 2 illustrate pictorially the data in the table for easier interpretation. Each bar corresponds to the number of structural errors in the reconstructed network per edge of the true network.

Effects of over-constraining the value of *k* in SC. One of the problems with **SC** is that the user has to select a value for the parameter *k*. Figure 2(a) plots the average number of structural errors over five runs of **SC** on 5000-instances datasets sampled from Alarm as *k* increases. As we can see, the number of errors highly depends on the value of *k*.

A theoretical interpretation of this dependence follows. Consider a node *X* with several parents, including *Y* and let us assume *Y* failed to enter the candidate parents set of *X*, e.g., because *X* has more than *k* parents. Since *Y* is a parent of *X*, it will probably have a significant association with *X*. That means that the hill-climbing search will erroneously tend to add the reverse edge $X \rightarrow Y$ since, like the true edge, it also increases the Bayesian scoring (because it explains the high association between the two variables). This in turn may prohibit other true edges from being added because they create cycles with the erroneously directed edge. In other words, we expect errors created by over-constraining the search (i.e., setting a low value for *k*) to propagate to other parts of the network.

Figures 2(b), 2(c), and 2(d) show a representative example. **SC** was run on 5000 instances sampled from the distribution of the Alarm system. Figure 2(b) shows a piece of the true Alarm network structure around variable with index 14, which is the variable with the maximum in-degree of 4; Figures 2(c) and 2(d) show the structures identified by **SC** with *k=8* and *k=9* respectively after 3 iterations in each case. The bold edges denote false positives. As we can see when *k=8, not one but all of the parents of variable 14 are erroneously reported as children*. The errors are propagated to other nodes, such as variable 19 and a number of other false positive edges are added to the structure. When *k* is increased to 9, the number of structural errors falls from 10 to 2, a sharp reduction. In contrast, for variable 14, the candidate parent set as estimated by **MMPC** is the set {15,16,18,31,13} that includes all the parents (and all children as it is supposed to). This allows hill-climbing to discover the correct structure within one iteration with only one missing edge $15 \rightarrow 14$.

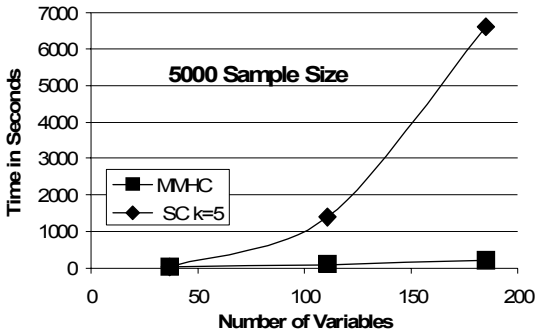


Figure 3 -

MMHC vs SC in terms of time efficiency. The timing results of the algorithms are again shown in Table 2 (column "Time in Seconds"). A variety of hardware systems were used for the experiments (Pentium IV, 1.7GHz-2.4GHz, .5GB-2GB RAM). However, the algorithms ran on the same machine for a given network; thus, the results are comparable across algorithms, but not among different networks. **SC** is faster than **MMHC** in the small Child network and the Munin network for 500 sample size, where the conditional independence tests fail often, resulting in large candidate sets. The most impressive cases of **MMHC** outperforming **SC** were observed for the largest networks, namely Alarm5 and Gene. For Alarm5, **MMHC** runs about 63 times faster, and for Gene, 49 times faster.

Most importantly, when we plot the average time in seconds (over five runs) as the number of variables increases (going from regular Alarm to Alarm3 and Alarm5), the difference in computation time required quickly grows between the two algorithms

(Figure 3). As can been seen in this experiment, the time taken by **MMHC** grows linearly to the number of variables, while **SC** shows exponential growth.

We further experimented by modifying **MMHC** to continue iterating in a similar fashion as **SC** and re-estimating the candidate sets. Our results, not shown here, indicate that there is no improvement in the score with further iterations and that one iteration is enough for **MMHC** to reach its potential. In general, **SC** uses simpler heuristics than **MMHC** to estimate the candidate sets, however, it requires iterating between search and re-estimation of the sets. **MMHC** on the other hand puts more effort in the initial selection of candidate sets so that only one search cycle is required. Our experiments indicate that the different strategy pays off in great computational savings.

Discussion and Conclusions

Prior work exists in combining constrained-based methods with Bayesian scoring methods [13, 14]. All such methods use the **PC** [7] algorithm to constraint the search space, which does not scale up to a large number of variables. Other algorithms exist for BN learning such as K2, however, we chose to compare with **SC** for two reasons: (a) it is the only algorithm that has been shown experimentally to scale up to hundreds of variables in biomedical applications, and (b) uses the theoretically advantageous BDeu scoring metric.

For the future, we intend to further test potential improvements to the algorithm such as incorporating the monotone faithfulness assumption in order to increase the time efficiency, and trying other statistical tests to increase the accuracy. We also intend to compare the algorithm with other standard BN learning algorithms on a larger set of datasets.

The contributions of this paper are two-fold. First, we show the sensitivity of the **SC** to the parameter *k*. This is important information for practitioners who apply **SC** for causal discovery such as identification of gene regulatory pathways, since the structure discovered may be an artifact of a low value of *k* that over-constrains the search.

Second, we present a sound (in the sample limit) and efficient way of estimating the candidate sets that results in the improved algorithm **MMHC** that discovers networks that are structurally closer to the real network, more closely capture the joint distribution of the data than those returned by **SC**, and does not require guessing the value of the parameter *k*.

Learning Bayesian Networks provides a formal and intuitive way of representing and reasoning about causality and thus, is expected to play a significant role in biological knowledge discovery. BN learning is also important for automatically constructing decision support systems from data, and thus for the clinical practice. **MMHC** is a step forward in improving the algorithmic arsenal available in solving the BN learning problem.

Acknowledgements:

First author was supported by NLM grant T15 LM07450-01. The second third authors by NIH grants R01 LM007948-01 and P20 LM007613-01.

References

[1] Beinlich IA, Suermondt H, Chavez R, Cooper G. The ALARM monitoring system: A case study with two probabilistic inference techniques for belief networks. In: *Second European Conference in Artificial Intelligence in Medicine*; 1989; 1989.

[2] Cowell RG, Dawid AP, Lauritzen SL, Spiegelhalter DJ. *Probabilistic Networks and Expert Systems*: Springer; 1999.

[3] Andreassen S, Jensen FV, Andersen SK, Falck B, Kjarulff U, Woldbye M, et al. MUNIN - An Expert EMG Assistant. In: Desmedt JE, editor. *Computer-Aided Electromyography and Expert Systems*; 1989.

[4] Baeze-Yates R, Ribiero-Neto B. *Modern Information Retrieval*: Addison-Wesley Pub Co; 1999.

[5] Friedman N, Linial M, Nachman I, Pe'er D. Using Bayesian Networks to Analyze Expression *Data. Computational Biology* 2000;7:601-620.

[6] Chapman WW, Fizman M, Chapman BE, Haug PJ. A comparison of classification algorithms to automatically identify chest X-ray reports that support pneumonia. *Journal of Biomedical Informatics* 2001;34(1):4-14.

[7] Spirtes P, Glymour C, Scheines R. *Causation, Prediction, and Search.* Second ed. Cambridge, Massachusetts, London, England: The MIT Press; 2000.

[8] Heckerman DE, Geiger D, Chickering DM. Learning Bayesian Networks: The Combination of Knowledge and Statistical Data. *Machine Learning* 1995;20(3):197-243.

[9] Friedman N, Nachman I, Pe'er D. Learning Bayesian Network Structure from Massive Datasets: The ``Sparse Candidate'' Algorithm. In: *Fifteenth Conference on Uncertainty in Artificial Intelligence* (UAI); 1999; 1999.

[10] Tsamardinos I, Aliferis CF, Statnikov A. Time and Sample Efficient Discovery of Markov Blankets and Direct Causal Relations. In: *The Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (KDD 2003); 2003; 2003.

[11] Spellman PT, Sherlock G, Zhang MQ, Iyer VR, Anders K, Eisen MB, et al. Comprehensive identification of cell cycle-regulated genes of the yeast saccharomyces cerevisiae by microarray hybridization. *Molecular Biology of the Cell* 1998;9:3273-3297.

[12] Statnikov A, Tsamardinos I, Aliferis CF. *An Algorithm for the Generation of Large Bayesian Networks*: Department of Biomedical Informatics, Vanderbilt University; 2003. Report No.: DSL TR-03-01.

[13] Singh M, Valtorta M. An Algorithm for the Construction of Bayesian Network Structures from Data. In: *Uncertainty in Artificial Intelligence*; 1993; 1993. p. 259-265.

[14] Spirtes P, Meek C. Learning Bayesian Networks with Discrete Variables from Data. In: *Knowledge Discovery and Data Mining*; 1995; 1995. p. 294-299.

Address for correspondence

Laura E. Brown, 2209 Garland Ave, Nashville, TN, 37232-8340, laura.e.brown@vanderbilt.edu