LSBN: A Large-Scale Bayesian Structure Learning Framework for Model Averaging

Yang Lu*, Mengying Wang†, Menglu Li*, Qili Zhu† and Bo Yuan*

*Department of Computer Science
Shanghai Jiao Tong University, Shanghai, China
Email: {luyang0415, iby07, boyuan}@sjtu.edu.cn
†Software Engineering Institute
East China Normal University, Shanghai, China
Email: mywang@sei.ecnu.edu.cn

Abstract—The motivation for this paper is to apply Bayesian structure learning using Model Averaging in large-scale networks. Currently, Bayesian model averaging algorithm is applicable to networks with only tens of variables, restrained by its super-exponential complexity. We present a novel framework, called LSBN (Large-Scale Bayesian Network), making it possible to handle networks with infinite size by following the principle of divide-and-conquer.

The method of LSBN comprises three steps. In general, LSBN first performs the partition by using a second-order partition strategy, which achieves more robust results. LSBN conducts sampling and structure learning within each overlapping community after the community is isolated from other variables by Markov Blanket. Finally LSBN employs an efficient algorithm, to merge structures of overlapping communities into a whole.

In comparison with other four state-of-art large-scale network structure learning algorithms such as ARACNE, PC, Greedy Search and MMHC, LSBN shows comparable results in five common benchmark datasets, evaluated by precision, recall and f-score. What’s more, LSBN makes it possible to learn large-scale Bayesian structure by Model Averaging which used to be intractable.

In summary, LSBN provides an scalable and parallel framework for the reconstruction of network structures. Besides, the complete information of overlapping communities serves as the byproduct, which could be used to mine meaningful clusters in biological networks, such as protein-protein-interaction network or gene regulatory network, as well as in social network.

I. INTRODUCTION

Structure learning from sparse data serves as a central problem in a variety of research area, for it uncovers underlying relationships, dependencies among variables, and more importantly, brings forth a structured, easily-understood model for further prediction and inference. As a major structure learning approach, a Bayesian network describes a probabilistic graphical model by representing a set of random variables and conditional dependencies via a directed acyclic graph (DAG). What’s more, a Bayesian network provides a very flexible framework to fuse different types of data and prior knowledge together to derive a synthesized network.

To achieve more robust and proper results in Bayesian structure learning, it is preferable to integrate over all possible structure models by using Bayesian model averaging. However, with the number of network variables growing, the enumeration of all possible structures becomes intractable and impractical, for there are overall \( \frac{n!}{n^{(2)}} \) possible structures given \( n \) network variables\(^{[30]} \). In short, structure learning by using model averaging is NP-hard even when the maximum parents number of network variable is bound to certain constant value \( k \(^{[19]} \). However, in real applications, ranging from causality network to Protein-Protein-Interaction network, the scales are much larger than traditional structure learning by using model averaging could support.

A very natural and logical attempt to scale the Bayesian structure learning beyond its limitation of variable numbers is to partition the variables into multiples groups, thus employing the method separately and efficiently. Manual partition is one option\(^{[25]} \), yet subjective factors would inevitably play a nontrivial role and possibly influence the ultimate result. Another widely-applied approach involves prior knowledge\(^{[25]} \), where domain knowledge is exploited to distinguish closely related variables thus guide the partition. For example, a common application under guidance of prior knowledge is Gene Regulatory Network (GRN) inference from gene expression data. In this case, cluster analysis is frequently applied to find similar functional groups, based on the assumption that genes presented by similar expression patterns tend to be co-regulated or interact\(^{[18]} \).

Unfortunately, the partition strategy is confronted with three fundamental limitations. The first problem lies in the lack of prior knowledge in most cases. It is neither practical nor tractable to collect prior knowledge for a special purpose beforehand. The second problem is that even there does exist prior knowledge, it remains challenges to quantify the knowledge as the network prior distribution. For example,
if the prior distribution is assigned with higher value, significant bias could be resulted towards the prior knowledge, leading to unwarranted structures without paying sufficient attention to data. Or the prior distribution is insignificant, it won’t do help to improve learning results. The third problem arises when we attempt to obtain prior knowledge directly from data by using statistical measurements, such as correlation coefficients, mutual information. However, these measurements are limited to pairwise information so that different measurements would inevitably lead to different partition results.

In this paper, we propose a novel framework **LSBN** (Large-Scale Bayesian Network) to learn Bayesian structure for sufficiently large networks. The basic idea of our framework is motivated by the philosophy of divide-and-conquer. Specifically, **LSBN** recursively partitions network variables into multiple communities with much smaller sizes, learns intra-community variables respectively before merge them altogether again. Our contributions lie in three aspects:

- We propose a robust partition algorithm, called ‘**ROPART**’, to segment large-scale network variables into multiple overlapping communities with much smaller sizes. According to the traditional graph clustering problem, whether variables are allocated into the same group depends on their edge weight, denoting the interrelated closeness among each other. Therefore, how to measure edge weights among variables is a challenge. Common measurements, such as mutual information, pearson coefficient, show different partition results given the same data. No one dominate the other, for each one performs excellent in some datasets and dissatisfactory in others. So **ROPART** introduces a second-order partition strategy, to overcome this shortcoming in a robust way. (Section V-A)

- We propose a sampling strategy to generate smaller sub-communities when current community is still too large to perform practical Bayesian structure learning. Also, we figure out how to isolate the dependencies of intra-community variables from those outside the community. The isolation makes intra-community structure learning unbiased and credible. What’s more, we categorize and analyze primary types of error edges from structure learning, and apply a uniform strategy to resolve the problem with satisfactory results. (Section IV-B)

- We propose an efficient algorithm, called ‘**MERGENCE**’ to merge the intra-community results into a whole. **MERGENCE** involves seeking an efficient mergence order, and resolves conflicts during the process of mergence. (Section IV-C)

We benchmark evaluation of **LSBN** on five well-known datasets, in comparison with four state-of-art structure learning algorithms. The compared results reveal that **LSBN** achieves comparable results to state-of-art structure learning algorithms, meanwhile, **LSBN** makes it possible to learn Bayesian structure by Model Averaging which used to be intractable in large-scale network.

### II. RELATED WORK

The problem of static Bayesian Structure Learning is well-studied. Exclusive of Bayesian Model Averaging, the four major approaches are information-theoretic, constrain-based, score-and-search and hybrid. And several representative algorithms would be included in performance evaluation (Section V).

The first major approach to Bayesian Structure Learning is based on Information Theory Models. They weigh network edges by correlation coefficients or statistic scores derived from Mutual Information, such as RELNET, ARACNE and CLR. Though most of information-theoretic approaches are subject to unweighted networks, an asymmetric variation of Mutual Information measurement could be employed to obtain directed networks. The advantages of Information Theory Models lie in extreme simplicity and low computational cost. However, such models could only take into consideration pairwise relationship rather than multiple variables at the same time. Another drawback is such models usually require plenty of observation data for the sake of accuracy.

The second major approach is constrain-based algorithm. Specifically, constrain-based algorithms use conditional independence (CI) tests to reveal the target DAG, such as PC and RAI. The drawback of such algorithms is that number of requisite CI tests grows exponentially with the number of variables, so polynomial complexity could only be ensured by imposing the maximum parents number. Besides, such algorithms inevitably miss or wrongly identify V-structures, which would affect the orientation of edges and even subsequent stages.

The third major approach performs a score-and-search strategy. In general, score-and-search algorithms search through structure space guided by a scoring function. One of the most basic score-and-search algorithms is Greedy Search. Since the size of structure space grows super-exponentially, the search approach would get inevitably trapped into local maximum, even there are many ways to escape, such as random restarts, simulated annealing or search in the space of equivalence classes of DAGs, called PDAGs.

The forth major approach serves as a hybrid approach. Hybrid approaches integrate constrain-based and score-and-search algorithms together. MMHC (Max-Min Hill-Climbing) shows superiority to other algorithms by combining local learning, reconstructing the skeleton of a
Bayesian network by constrain-based approach, and performing greedy hill-climbing search for edge orientation.

Great amounts of work has been devoted to detecting overlapping communities in large networks. The most popular algorithm is Clique Percolation Method (CPM)\[23\] which computes all k cliques and two variables belong to the same cluster if there exists a path going through k - l cliques between them. CPM is implemented by CFinder[4](http://www.cfinder.org/). Besides, overlapping communities detection could be roughly categorized into threefold: optimization, clustering and partitioning.

One of traditional methods regards overlapping communities detection as an optimization problem, specifically, each community is identified as a subgraph reaching local optimum given quality function f. Besides, overlapping communities detection could be roughly categorized into threefold: optimization, clustering and partitioning. Those overlapping nodes are identified by combination with spectral mapping and fuzzy clustering\[39\]. As one can see, node 7 for partition P1 in Figure 1a is PSM(node7) = \([0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 1, 1]^T\), while for partition P2 in Figure 1b is PSM(node7) = \([0, 0, 1, 0, 0, 1, 1, 1]\). Then the overall partition support matrix of node7 is

\[
PSM(node7) = \begin{pmatrix}
0, 0, 0, 1, 0, 0, 0, 0, 1, 1, 1 \\
0, 0, 0, 0, 0, 0, 1, 1, 1 \\
0, 0, 0, 1, 0, 0, 1, 1, 1
\end{pmatrix}
\]

As one can see, node7 tends to be grouped together with node4, node8 and node9.

D. DEFINITION 4 (Second-Order Network)

Given a weighted graph set D = \(\{G_1, ... G_n\}\), where \(G_i = (V, E_i)\), and its corresponding partition set \(P = \{P_1, ... P_n\}\), the second-order network in an undirected, weighted graph \(S = (V, E_S)\) where an edge \(e = (u, v) \in E_S\) if its edge weight exceeds threshold. The edge weight is valued by co-occurrence probability in partition support matrix between u and v.

E. DEFINITION 5 (Second-Order Partition)

A second-order partition is the partition \(P_{sec}\) of a second-order network.

IV. LSBN SYSTEM: LARGE-SCALE BAYESIAN NETWORK LEARNING

The LSBN system provides a novel framework for Bayesian structure learning using model averaging in large-scale networks. The LSBN system proposes a divide-and-conquer strategy to segment the originally intractable Bayesian structure learning tasks into multiple tractable sub-tasks with a much smaller scale. The workflow of...
The LSBN system is as follows:

1. Variable Partition. (Section IV-A)
2. Sampling and Learning. (Section IV-B)
3. Mergence. (Section IV-C)

A. Variable Partition

The LSBN system is expected to perform well even the number of variables increases. Variable partition serves as a crucial step by drastically reducing the complexity and run-time of learning in our next stage. Partition incorporates overlapping, that is, each node may belong to more than one community. Ideally, a perfect partition should possess three properties: selectiveness, so that nodes within the communities have much higher probability to possess correct edges among each other than outside of the communities; high coverage, so that all correct edges are embodied within at least one community, in other words, partition doesn’t break any correct edges; and fine granularity, so that each community is small enough to be applicable for Bayesian structure learning algorithms.

The initial input of the LSBN system requires discrete-state data. If the data is continuous, discretization is necessary before any further steps. We develop a novel partition algorithm, called ‘ROPART’, to construct robust partition of all variables. ROPART consists of five steps, as outlined in Algorithm 1 and illustrated in Figure 2.

Algorithm 1 ROPART

1. build an undirected, unweighted complete graph $G$ given all variables;
2. for each weight function $f_i$ from a predefined weight function set $F = \{f_1, ..., f_n\}$, map $G$ to a weighted graph $G_i$ respectively;
3. for each weighted graph $G_i$, keep those edges of which the weight exceed some truncate threshold $T_{trunc}$, after that, generate partition $P_i$ respectively;
4. for each variable $v$, construct its corresponding partition support matrix $PSM(v)$;
5. construct second-order network and generate second-order partition;

In step 1, We start with a fully-connected, undirected, unweighted graph including all variables.

In step 2, We employ a predefined weight function set $F = \{f_1, ..., f_n\}$ to translate original unweighted graph into a set of various weighted graphs $D = \{G_1, ..., G_n\}$, where $f_i$ corresponds to $G_i$. The predefined weight functions are shown as follows:

- Mutual Information
  
  $MI(X, Y) = \sum_x \sum_y P(x, y) \log \frac{P(x, y)}{P(x)P(y)}$

- Mutual Information normalized by the sum of entropies
  
  $MI_{\text{plus}}(X, Y) = \frac{2MI(X, Y)}{H(X) + H(Y)}$

- Mutual Information normalized by the square root product of entropies
  
  $MI_{\text{sqrt}}(X, Y) = \frac{MI(X, Y)}{\sqrt{H(X)}\sqrt{H(Y)}}$

- Mutual Information normalized by PageRank weight
  
  $MI_{\text{pr}}(X, Y) = \frac{MI(X, Y)}{\sqrt{PR(X)PR(Y)}}$

  where $PR(X)$ and $PR(Y)$ are PageRank values of node $X$ and $Y$ respectively.

- Mutual Information by Standard Normalization
  
  $MI_{\text{sn}}(X, Y) = \frac{MI(X, Y) - \mu_{MI}}{\sigma_{MI}}$

  where $\mu_{MI}$ and $\sigma_{MI}$ denotes the mean value and standard deviation of all edge weights, which are valued by mutual information.

- Pearson Correlation Coefficient
  
  $\rho_{X,Y} = \frac{E[(X - \mu_X)(Y - \mu_Y)]}{\sigma_X \sigma_Y}$
where $\mu_X$ and $\mu_Y$ denote the mean value of variable $X$ and $Y$ respectively, and $\sigma_X$ and $\sigma_Y$ indicate the standard deviation of $X$ and $Y$ correspondingly.

Each weighted graph $G_i$ derived by weight function $f_i$ is pruned by removing edges whose weight is lower than some truncate threshold $\mathcal{T}_{\text{trunc}}$.

In step 3, we partition each weighted graph $G_i$ after pruning. For the sake of convenience in Mergence(Section IV-C), we prefer communities to be organized hierarchically. Meanwhile, for the sake of high coverage, we wish communities to maintain pervasive overlaps. Here we use LinkComm\textsuperscript{5} for partition algorithm, which is introduced in Section II.

In step 4, We construct partition support matrix $PSM(v)$ for each variable $v$, based on partition result set $P = \{P_1,...,P_n\}$ generated from step 3, where $P_i$ corresponds to weighted graph $G_i$.

In step 5, We build second-order network based on partition support matrix set $PSM = \{PSM(v_1),...,PSM(v_{|V|})\}$ and generate second-order partition $P_{\text{sec}}$. The resulting second-order partition will satisfy the criteria for a good partition: (1) the communities are of high cohesion and low coupling; and (2) the partition is robust to the selection of weight function.

B. Sampling and Learning

The LSBN system performs Bayesian structure learning per community, however, to learn the structure correctly poses many challenges. For example, what if an outlier variable is erroneously mixed into a community, how to pinpoint such variables, and what’s most important, how to eliminate the learning bias caused by irrelevant variables in the same community? What’s more, what if correct edges suffer disconnection during partition, is there any mechanism to retrieve those missing edges? We first attempt to find out candidate Markov Blanket given nodes within community, after that, we consider a sampling methodology to address mentioned challenges. The detailed learning
algorithm consists of four steps, as outlined in Algorithm 2 and illustrated in Figure 3.

**Algorithm 2 LocalLearn**

1: for each community \( C \in P_{sec} \) do
2: identify candidate Markov Blanket \( MB(C) \)
3: sample \( k \) sub-communities based on \( C \) and \( MB(C) \), \( SC = \{ sub_1(C), ..., sub_k(C) \} \)
4: for each sub-community \( sc_i \in SC \) do
5: learn Bayesian structure
6: end for
7: ensemble structures of all sub-communities and resolve conflicts
8: end for

In Step 1, we try to find out the Markov Blanket \( MB(C) \) for community \( C \). Conditioned on Markov Blanket, no other nodes outside the community \( C \) could influence nodes within the community. To be more precise, \( \forall X \in C, X \perp Y | MB(X) \) for all \( Y \not\in \{ X \cup MB(X) \} \). Therefore knowledge of Markov Blanket \( MB(C) \) is enough to infer intra-community structure, thus makes the learning problem localized and easier to be solved.

According to definition, the Markov Blanket of variable \( X \) is composed of all parents of \( X \), all children of \( X \) and all parents of \( X \)'s children. In other words, Markov Blanket of \( X \) should be closer to \( X \) topologically than any other variables in networks.

Since the Markov Blanket of variable \( X \) is composed of all parents of \( X \), all children of \( X \) and all parents of \( X \)'s children, in other words, Markov Blanket of \( X \) should be closer to \( X \) topologically than any other variables in structure. Besides, topological closeness is related to significance in edge weights, thus edge weights play an important role in identifying Markov Blanket. In addition, conditional independence (CI) tests and measurement of association among variables are required for further justification. The identification of Markov Blanket is divided into two steps:

1. For variable \( X \), we achieve its Markov Blanket candidates by looking for adjacent nodes whose edge weight exceeds the average. To be precise, \( MB_{cand}(X) = \{ Y | w_{XY} \geq \bar{w}_X \} \) where \( w_{XY} \) is the weight of edge \( e_{XY} \) and \( \bar{w}_X \) is the average weight of edges connecting \( X \).

2. Given \( MB_{cand}(X) \), we employ a scalable algorithm IAMB\(^{35} \) for further justification. IAMB\(^{35} \) (Incremental Association Markov Blanket) consists of a forward phase and a backward phase. In the forward phase, IAMB applies a heuristic approach to find an estimated Markov Blanket set \( CMB \). \( CMB \) starts with an empty set, then iteratively added variables \( Y \) from \( MB_{cand}(X) \) which maximizes a heuristic function \( f(Y; X | CMB) \). The heuristic function \( f(Y; X | CMB) \) denotes the mutual information between candidate Markov Blanket node \( Y \) and target node \( X \) given \( CMB \), which is informative and efficient. In the backward phase, conditional independence tests are used within \( CMB \), and invalid variables are removed from \( CMB \) one by one if it is independent of \( X \) given the remaining \( CMB \).

In Step 2, we combine each community \( C \) with its Markov Blanket \( MB(C) \) into a expanded community \( C' \). The size of \( C' \) would be much smaller than expectation, for intra-community variables are highly correlated due to partition. Chances are high that members of Markov Blanket is embodied in community \( C \) as well.

Given expanded community \( C' \), we conduct sampling for two reasons: (1) the size of \( C' \) may be too large for practical Bayesian structure learning; (2) bootstrap contributes to more consistent and robust results. Our sampling algorithm borrows the idea of Random Node Neighbor (RNN)\(^{24} \) with some modifications.

In our sampling algorithm, we build up an inner markov graph \( G_{IM}(C) \) based on \( C \), to be precise, \( G_{IM}(C) = (C, E) \) where \( e_{XY} = 1 \) if \( X \in MB(Y) \) or \( Y \in MB(X) \), and 0 otherwise. Then we uniformly pick an unvisited node from inner markov graph \( G_{IM}(C) \) as starting node at random together with its neighbors, denoted by \( S \). The final sub-community for Bayesian structure learning is \( S \cup MB(S) \), if the size of ultimate sub-community is still too large to learn, we keep removing neighbors within \( S \) until acceptable size.

In Step 3, for each sub-community, LocalLearn uses
Bayesian model averaging to learn the structure of nodes within the sub-community. Bayesian model averaging is different from other structure learning algorithms, such as constrain-based approach and score-and-search approach, for it is interested in the confidence of some structure-related features rather than structure topology per se. In this case, feature \( f(G) = 1 \) denotes whether there exists an edge from node \( i \) to node \( j \) and \( f(G) = 0 \) otherwise. Usually posterior expectation of feature \( f(G) \) is estimated by:

\[
P(f|D) = \sum_G f(G) p(G|D)
\]

(3)

\( f(G) \) is classified to be 1 if \( E(f|D) \) exceed certain threshold \( T_{avg} \) and 0 otherwise. However, traversing all candidate DAGs according to Eq.(3) is usually intractable for there are overall \( O(n!2^n) \) DAGs given \( n \) nodes. One solution is to average over structures which conform to some predetermined node order \( \prec \) \[13\]. For example, if \( X_i \in Pa_G(X_j) \) then \( i \prec j \). Now the posterior expectation of feature \( f(G) \) given a predetermined order \( \prec \) could be transformed into\[13\]:

\[
P(f|D) = \sum_{\prec} P(\prec|D) P(f|D, \prec)
\]

(4)

Usually the order \( \prec \) is unobtainable due to little prior in current domain, so there are \( n! \) possible orders in all to be taken into consideration given \( n \) nodes, which remains intractable. So we use MCMC (Markov Chain Monte Carlo) techniques to sample orders, and sample DAGs consistent with that order\[13\]. Assuming a uniform prior over orders, a Markov Chain \( \mathcal{M} \) is constructed with state space consisting of all \( n! \) possible orders. This Markov Chain \( \mathcal{M} \) is simulated by conforming to stationary distribution \( P(\prec|D) \), and a sequence of sampled order \( \prec_1, \ldots, \prec_T \) is obtained. Now the expectation of feature \( f \) could be approximately estimated as\[13\]:

\[
P(f|D) \approx \frac{1}{T} \sum_{i=1}^{T} P(f|D, \prec_i)
\]

(5)

After learning sub-community structures, we integrate them into a uniform intra-community structure and resolve conflicts. We investigate the characteristics of error edges and find two major types of error:

- missing edges due to weak edge weight. This type of error originates from the adjacent nodes, maybe some nodes have relatively small PageRank value, in other words, they are periphery nodes; Or maybe some nodes are hubs, so the mutual information with its neighbors, that is, the edge weight, is relatively small.

To tackle these two major types of error, we first collect all candidate triplets of nodes. A candidate triplet contains three nodes, mutually-connected by the edge whose weight exceeds certain threshold value. Chances are high that these candidate triplets contain indirect interactions\[26\]. We build up a undirected, unweighted graph based on edges from candidate triplets. Then we cluster this graph into sparsely-connected dense subgraphs. Ideally, on one hand, indirect interactions are clustered with direct interactions. Since direct interactions have more significant edge weights, indirect interactions would be eliminated through re-learning. On the other hand, weak correct edges are expected to be grouped with weaker wrong edges. As a result, they survive through re-learning due to relative significance. Here we still employ LinkComm\[5\] for graph clustering. After clustering the candidate triplet graph, we re-learn the structure for each cluster using the same method in Step 3.

### C. Mergence

**Algorithm 3 Mergence**

1. Create a leaf node for every community and add it to the Community Pool;
2. while there is more than one node in community pool do
   3. Remove nodes \( C_i \) and \( C_j \) of maximum Jaccard similarity coefficient \( J(C_i, C_j) \) from the Community Pool;
   4. Merge structures of two communities \( C_i \) and \( C_j \);
   5. Create a new node \( C_{new} = C_i \cup C_j \) denoting the merge of two communities \( C_i \) and \( C_j \);
   6. Add the new node to the Community Pool;
3. end while

The \( LSBN \) system combines structures after learning individually from each community. The combination involves two concerns: (1) to find an efficient mergence order; (2) to resolve the conflicts during the mergence. Intuitively, the mergence strategy should proceed as a bottom-up approach. The \( LSBN \) system would keep piecing together all intra-community structures into larger structures, block by block until a whole structure is achieved. We expect the structure to be increasingly accurate and wrong edges would be continuously eliminated during the mergence.

By borrowing the idea of Huffman’s Algorithm\[20\], the \( LSBN \) system tries to merge communities in a greedy strategy by constantly pick two communities with maximum Jaccard similarity coefficient\[1\]. Jaccard similarity
coefficient\( J \) of two communities is proportional to their overlap and inversely proportional to their union size. We propose Mergence Algorithm to perform such greedy strategy in order to combine the structures of each community into whole better, and the detailed algorithm is outlined in Algorithm 3.

At first, Mergence Algorithm put all communities into a Community Pool, then repeatedly choose two communities \( C_i \) and \( C_j \) with largest Jaccard similarity coefficient \( J(C_i, C_j) = \frac{|C_i \cap C_j|}{|C_i \cup C_j|} \). After two communities are selected, same approach described in step 3 of Section IV-B is applied to resolve the conflicts by clustering triplets. Then Mergence Algorithm combine these two communities into a new hybrid community, and put it into the Community Pool for further mergence steps.

In each iteration, assuming there are \( k \) communities left in the Community Pool, then it would take \( \binom{k}{2} \times \sum_{k=2}^{n} \binom{n}{k} \) times of calculation to select the maximum value of Jaccard similarity coefficient. If there are \( n \) communities initially, the total calculation sums up to be:

\[
\sum_{k=2}^{n} \binom{n}{k} = \frac{1}{2} \sum_{k=2}^{n} k^2 - \frac{n(n+1)(n-1)}{6}.
\]

For the sake of computational efficiency, the values of Jaccard similarity coefficient could be calculated in advance. For each iteration, assuming there are \( k \) communities left in the Community Pool, removing old values would take only \( 2(k-1)+1 \) times and adding new values would take \( k-2 \) times of calculation. After computational optimization, the overall calculation shrinks to:

\[
\binom{k}{2} + \sum_{k=2}^{n} [2(k-1)+1 + (k-2)] = 2n(n-1).
\]

V. EXPERIMENTS

We benchmark evaluation of LSBN on five well-known datasets. We expect the structures learned by LSBN to be close enough to those learned by other Bayesian structure learning algorithms. Closeness in results indicates that partition and local learning in LSBN hardly cause any losses. In addition, since LSBN is designed to work on Bayesian structure learning problem in large-scale network, we expect LSBN to learn structures of which the sizes exceed the computational upper bound of traditional Bayesian model averaging approach.

A. Datasets

- alarm [8]. The alarm network consists of 37 random variables and 46 arcs, with average degree of network being 2.49, maximum in-degree being 4 and average Markov Blanket size being 3.51.

- insurance [9]. The insurance network contains 27 random variables and 52 arcs, with average degree of network being 3.85, maximum in-degree being 3 and average Markov Blanket size being 5.19.

- win95pts [3]. The win95pts network includes 76 random variables and 112 arcs, with average degree of network being 2.95, maximum in-degree being 7 and average Markov Blanket size being 5.92.

- pigs [2]. The pigs network includes 441 random variables and 592 arcs, with average degree of network being 2.68, maximum in-degree being 2 and average Markov Blanket size being 3.66.

- link [21]. The link network embodies 724 random variables and 1125 arcs, with average degree of network being 3.11, maximum in-degree being 3 and average Markov Blanket size being 4.8.

From each benchmark network, we sampled 20000 instances as the observed data. Besides, the pre-defined weight function set includes: (1) mutual information, abbreviated as ’MI’; (2) mutual information normalized by the sum of entropies, abbreviated as ’MI_{plus}’; (3) mutual information normalized by the square root product of entropies, abbreviated as ’MI_{sqrt}’; (4) mutual information normalized by the PageRank values, abbreviated as ’MI_{pr}’; (5) mutual information after standard normalization, abbreviated as ’MI_{sn}’; (6) Pearson Coefficient in absolute value, abbreviated as ’Pearson’; (7) absolute Pearson Coefficient after standard normalization, abbreviated as ’Pearson_{sn}’.

B. Parameter Setting

For each weighted network generated by certain weight function, the truncate threshold \( T_{trunc} \) for pruning is chosen by the Elbow Method [34]. Specifically, we first transform all edge weights into histogram, and each bin in the histogram denotes the frequency of edges weights falling in certain range. Then we look at the variance descent between each pair of adjacent bins. For example, the first bin will add much information (encompass a lot of variance), for the majority of the edges possess relatively very small weights. Yet at certain bin, the variance ratio slows down. And we choose that bin as the truncate threshold \( T_{trunc} \), also called ’elbow criterion’. The truncate thresholds we chosen are shown in Table II the percentage denotes the ratio between remaining edges and all edges in complete graph. From the results in Table II there is no significant difference in numbers of remaining edges.

We measure the average shortest path (Table II) and diameter (Table III) for each partition, in comparison of weighted network partitions and second-order partition. There are several noticeable phenomenon in these results. First, there is no dominating weight function, for the partition result of its weighted network performs excellent in some datasets but poor in others. For example, mutual information proves to be the best weight function in alarm according to average shortest path, yet among the worst ones in win95pts. Second, as expected, second-order partition achieves more stable results. The results always belong to the best ones and never
oscillate drastically. The average ranking for second-order partition in average shortest path is 2.4 (Table II), and the average ranking in diameter is 2.

The partition results are depicted in Table IV and the partition size distribution reveals that the size of the majority of communities ranges less or equal than 25 (100% in alarm, 100% in insurance, 100% in win95pts, 92.827% in pigs and 95.181% in link). The average community size is 10.286 in alarm, 9.8 in insurance, 8.345 in win95pts, 9.527 in pigs and 8.904 in link.

C. Experimental Design

Our evaluation benchmarks LSNB four state-of-art large-scale network structure learning algorithms, namely ARACNE[26], PC[31], Greedy Search[10] and Max-Min Hill Climbing (MMHC)[36]. Among these common algorithms, ARACNE[26] is a very popular information-theoretic algorithm with extreme simplicity and low computational cost; the PC algorithm[31] is considered as the most popular constrain-based algorithm; Greedy Search[10] are very widely-used score-and-search approaches; and MMHC[36] serves as a hybrid method which proves to be superior to other algorithms in most cases.

We compare LSNB to these four state-of-art algorithms with respect to its correctness in structures. In implementation of PC, Greedy Search and MMHC, we were aided by Causal Explorer Toolkit[6] (http://www.dsl-lab.org/causal_explorer) and structural results in the format of directed edges are evaluated by ourselves. The parameters used in Causal Explorer Toolkit for each algorithm are just set up as default, for example, threshold on statistical test is 5% in default for MMHC and Greedy Search; threshold on mutual information test is 1% in default for PC; prior type is chosen to be BDeu score[11] with Dirichlet Weight equals 10 for Greedy Search and MMHC. As for ARACNE, we implement the algorithm in person due to its simplicity, and the low values threshold \( \tau \) are selected manually based on tradeoff between true positives and false positives, shown in Table V.

| Dataset Name | low values threshold \( \tau \) |
|-------------|---------------------|
| alarm       | 0.01                |
| insurance   | 0.05                |
| win95pts    | 0.025               |
| pigs        | 0.01                |
| link        | 0.005               |

It’s inappropriate to make comparison between LSNB and other algorithms directly for the result of LSNB is determined by two factors: (1) the performance of Bayesian model averaging on five benchmark datasets; (2) the performance of LSNB to divide-and-conquer Bayesian model averaging. Due to the intractability of Bayesian model averaging, we expect to evaluate the framework of LSNB itself per se. If the performance of LSNB is proved to be satisfactory, we would conclude that LSNB could be well-applied to Bayesian model averaging as well.

As for the evaluation of LSNB framework, we slightly change LSNB to make more fair comparison between ARACNE, PC, Greedy Search and MMHC. Specifically, we replace the Bayesian model averaging process(Step 3 in Algorithm 2) with corresponding targeted structure learning algorithm. For example, given MMHC algorithm as comparison target, we would use a modified version of LSNB whose structure learning algorithm in LocalLearn is also MMHC while keep everything else unchanged. As a result, the performance of LSNB framework could be measured independently, regardless of influence brought from the usage of different structure learning algorithms.

D. Performance Evaluation

As for the evaluation of structure learning, we regard the Bayesian structure learning problem as a binary classification problem. For each pair of nodes, the Bayesian structure learning algorithm either assigns a positive label or a negative label to declare whether there is an edge existing between them or not.

We use precision, recall and F-score as our metrics to evaluate the performances of LSNB system. These metrics are defined as follows:

\[
\text{Precision} = \frac{TP}{TP + FP} \\
\text{Recall} = \frac{TP}{TP + FN} \\
F - \text{Score} = \frac{2 \cdot \text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}
\]

Where TP(True Positive) is the number of positive edges correctly classified as positive, corresponding to the hitting edges; FP(False Positive) is the number of negative edges mistakenly classified as positive, corresponding to the additional edges (error edges); TN(True Negative) is the number of negative edges correctly classified as negative; and FN(False Negative) is the number of positive edges mistakenly classified as negative, corresponding to the missing edges.

The comparison results between LSNB and other state-of-art algorithms such as ARACNE, PC, Greedy Search and MMHC are depicted in Table VI on Alarm Dataset, Table VII on Insurance Dataset, Table VIII on Win95pts Dataset, Table IX on Pigs DataSet and Table X on Link DataSet. The hit edge number, additional edge number and missing edge number as well as corresponding metrics such as Precision, Recall and F-Score of network results reconstructed LSNB are shown versus their counterparts generated by the algorithms performed in global space. Note that Bayesian model averaging, abbreviated as ‘Model Avg’,...
is intractable in global scope, so the relevant blankets are filled with 'NA'.

The precision of $LSBN$ shows little inferiority to global in benchmark algorithms such as ARACNE, PC and MMHC, but superiority in Greedy Search (Table XI). The recall of $LSBN$ shows superiority to global in ARACNE and PC, but little inferiority in Greedy Search and MMHC (Table XII). The F-Score serves as the harmonic mean of precision and recall, which shows comparable results to global in ARACNE, PC, Greedy Search and MMHC (Table XIII). The results of precision, recall and F-Score reveal that $LSBN$ per se does not introduce noticeable errors in the procedure of partition, sampling, intra-community structure learning and mergence. What’s more, in some cases, $LSBN$ even improve the learning quality.

As for our target, Bayesian model averaging, there is no comparison result available. By referring to other algorithms, it performs well in datasets such as Alarm, Insurance, Win95pts and link. Despite the significant disparity in Pigs, the results learned by using Bayesian model averaging is close to the results learned by other state-of-art algorithms in most cases.

| Table I | TRUNCATE THRESHOLD FOR EACH WEIGHT FUNCTION ON EACH DATASET |
|---------|------------------------------------------------------------|
|         | insurance | alarm | win95pts | pigs | link |
| MI      | 0.08      | 0.2308 | 0.01     | 0.1659 | 0.005 | 0.0841 | 0.035 | 0.0566 |
| $MI_{\text{plus}}$ | 0.2 | 0.2792 | 0.03 | 0.2312 | 0.05 | 0.1635 | 0.05 | 0.0958 | 0.4 | 0.0435 |
| $MI_{\text{sqrt}}$ | 0.5 | 0.3646 | 0.5 | 0.2132 | 0.4 | 0.1618 | 0.4 | 0.0726 | 2.5 | 0.0493 |
| $MI_{\text{pr}}$ | 0.05 | 0.3276 | 0.02 | 0.2267 | 0.013 | 0.1687 | 0.005 | 0.0762 | 0.5 | 0.0510 |
| $MI_{\text{sn}}$ | 0 | 0.2707 | 0 | 0.2072 | 0.05 | 0.1368 | 0.005 | 0.0762 | 0.035 | 0.0566 |
| Pearson | 0.1 | 0.4017 | 0.4 | 0.2747 | 0.02 | 0.1894 | 0.026 | 0.1516 | 0.075 | 0.0585 |
| $Pearson_{\text{sn}}$ | 0 | 0.3789 | 0 | 0.2087 | 0 | 0.0731 | 1.0 | 0.0490 |

| Table II | COMPARISON OF AVERAGING SHORTEST PATH FOR VARIOUS WEIGHT FUNCTIONS |
|---------|----------------------------------------------------------|
|         | insurance | alarm | win95pts | pigs | link |
| MI      | 1.533(3) | 3.8162(1) | 2.9799(6) | 5.2616(4) | 2.6427(4) |
| $MI_{\text{plus}}$ | 1.55(4) | 4.1567(8) | 3.1130(8) | 5.4789(5) | 2.4552(1) |
| $MI_{\text{sqrt}}$ | 1.9456(8) | 3.8792(3) | 3.0156(7) | 5.6248(7) | 2.4622(2) |
| $MI_{\text{pr}}$ | 1.7983(6) | 3.8567(2) | 2.5521(1) | 5.1297(3) | 3.7409(6) |
| $MI_{\text{sn}}$ | 1.4927(2) | 3.9682(7) | 2.8828(5) | 5.5749(6) | 4.4982(8) |
| Pearson | 1.7357(5) | 3.8991(5) | 2.7558(2) | 4.4463(2) | 2.6801(5) |
| $Pearson_{\text{sn}}$ | 1.8430(7) | 3.9178(6) | 2.8380(4) | 5.6633(8) | 4.0882(2) |
| second-order | 1.36(1) | 3.8875(4) | 2.8240(3) | 4.2819(8) | 2.5624(3) |

Note: The number within the parentheses denotes the ascending ranking of current weight function in certain dataset.

| Table III | COMPARISON OF AVERAGING DIAMETERS FOR VARIOUS WEIGHT FUNCTIONS |
|---------|----------------------------------------------------------|
|         | insurance | alarm | win95pts | pigs | link |
| MI      | 3.0(3) | 6.1667(2) | 5.1429(6) | 10.6875(4) | 5.0026(8) |
| $MI_{\text{plus}}$ | 3.0(3) | 6.87(7) | 5.4(8) | 11.2247(5) | 4.3666(4) |
| $MI_{\text{sqrt}}$ | 4.0(8) | 6.8333(8) | 5.1667(7) | 11.672(8) | 4.3480(3) |
| $MI_{\text{pr}}$ | 3.6(6) | 6.2(3) | 4.05(1) | 9.4707(3) | 4.6027(6) |
| $MI_{\text{sn}}$ | 2.7143(2) | 6.7143(6) | 4.6857(4) | 11.4805(7) | 4.4982(5) |
| Pearson | 3.33(5) | 6.1(1) | 4.5556(2) | 8.3585(2) | 4.9513(7) |
| $Pearson_{\text{sn}}$ | 3.778(7) | 6.625(5) | 4.886(5) | 11.3907(6) | 4.0882(2) |
| second-order | 2.423(1) | 6.4211(4) | 4.619(3) | 7.7565(1) | 3.7409(1) |

Note: The number within the parentheses denotes the ascending ranking of current weight function in certain dataset.

| Table IV | PARTITION SIZE DISTRIBUTION FOR ALL DATASETS |
|---------|-----------------------------------------------|
|         | 1-5 | 6-10 | 11-15 | 16-20 | 21-25 | 26-30 | 31-35 | 36-40 | 41-45 | 46-50 | >50 |
| alarm    | 3 | 2 | 1 | 2 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| insurance | 1 | 3 | 0 | 6 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| win95pts | 19 | 19 | 13 | 4 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| pigs     | 102 | 76 | 24 | 11 | 7 | 6 | 6 | 1 | 1 | 1 | 2 |
| link     | 81 | 47 | 16 | 5 | 9 | 3 | 0 | 2 | 3 | 0 | 0 |
TABLE VI
Evaluation of LSBN against other algorithms on Alarm Dataset

| Algorithm | Hit(TP) | Miss(FN) | Error(FP) | Precision | Recall | F-Score | Hit(TP) | Miss(FN) | Error(FP) | Precision | Recall | F-Score |
|-----------|---------|----------|-----------|-----------|--------|---------|---------|----------|-----------|-----------|--------|---------|
| ARACNE    | 39      | 7        | 6         | 86.667    | 84.783 | 85.714  | 31      | 15       | 4         | 88.571    | 67.391 | 76.543  |
| PC        | 38      | 8        | 0         | 100       | 82.609 | 90.476  | 35      | 11       | 0         | 100       | 76.087 | 86.420  |
| Greedy    | 43      | 3        | 4         | 82.979    | 84.783 | 83.871  | 44      | 2        | 9         | 83.019    | 95.652 | 88.889  |
| MMHC      | 43      | 3        | 3         | 93.478    | 93.478 | 93.478  | 44      | 2        | 1         | 97.778    | 95.652 | 96.703  |
| Model Avg | 43      | 3        | 8         | 84.314    | 93.478 | 88.660  | NA      | NA       | NA        | NA        | NA     | NA     |

TABLE VII
Evaluation of LSBN against other algorithms on Insurance Dataset

| Algorithm | Hit(TP) | Miss(FN) | Error(FP) | Precision | Recall | F-Score | Hit(TP) | Miss(FN) | Error(FP) | Precision | Recall | F-Score |
|-----------|---------|----------|-----------|-----------|--------|---------|---------|----------|-----------|-----------|--------|---------|
| ARACNE    | 33      | 19       | 4         | 89.189    | 63.462 | 74.157  | 25      | 27       | 2         | 92.593    | 48.077 | 63.291  |
| PC        | 36      | 16       | 3         | 92.308    | 69.231 | 79.121  | 31      | 21       | 1         | 96.875    | 59.615 | 73.810  |
| Greedy    | 41      | 11       | 7         | 87.179    | 65.385 | 82.000  | 47      | 5        | 11        | 81.034    | 90.385 | 88.889  |
| MMHC      | 43      | 9        | 4         | 91.489    | 82.692 | 86.869  | 43      | 9        | 2         | 95.556    | 82.692 | 88.660  |
| Model Avg | 45      | 7        | 7         | 85.538    | 86.538 | 86.538  | NA      | NA       | NA        | NA        | NA     | NA     |

TABLE VIII
Evaluation of LSBN against other algorithms on Win95pts Dataset

| Algorithm | Hit(TP) | Miss(FN) | Error(FP) | Precision | Recall | F-Score | Hit(TP) | Miss(FN) | Error(FP) | Precision | Recall | F-Score |
|-----------|---------|----------|-----------|-----------|--------|---------|---------|----------|-----------|-----------|--------|---------|
| ARACNE    | 81      | 31       | 39        | 67.500    | 72.321 | 69.828  | 53      | 59       | 8         | 86.885    | 47.321 | 61.272  |
| PC        | 64      | 48       | 8         | 92.308    | 57.143 | 69.565  | 38      | 74       | 3         | 92.683    | 33.921 | 49.673  |
| Greedy    | 99      | 13       | 133       | 40.909    | 88.393 | 55.932  | 94      | 18       | 106       | 47.000    | 83.929 | 60.256  |
| MMHC      | 92      | 20       | 56        | 62.162    | 80.357 | 76.923  | 90      | 22       | 32        | 73.770    | 80.357 | 76.923  |
| Model Avg | 98      | 14       | 93        | 51.309    | 87.500 | 64.686  | NA      | NA       | NA        | NA        | NA     | NA     |

TABLE IX
Evaluation of LSBN against other algorithms on Pigs Dataset

| Algorithm | Hit(TP) | Miss(FN) | Error(FP) | Precision | Recall | F-Score | Hit(TP) | Miss(FN) | Error(FP) | Precision | Recall | F-Score |
|-----------|---------|----------|-----------|-----------|--------|---------|---------|----------|-----------|-----------|--------|---------|
| ARACNE    | 592     | 0        | 14        | 97.690    | 100.00 | 98.831  | 592     | 15       | 4         | 99.831    | 100.00 | 99.916  |
| PC        | 574     | 18       | 0         | 100.00    | 96.959 | 98.456  | 591     | 1        | 8         | 98.664    | 99.831 | 99.244  |
| Greedy    | 570     | 22       | 13        | 97.770    | 96.284 | 97.021  | 592     | 0        | 47        | 92.645    | 100.00 | 96.182  |
| MMHC      | 574     | 18       | 2         | 99.653    | 96.959 | 98.288  | 592     | 0        | 0         | 100.00    | 100.00 | 100.00  |
| Model Avg | 447     | 145      | 940       | 32.228    | 75.507 | 45.174  | NA      | NA       | NA        | NA        | NA     | NA     |

VI. CONCLUSION
In this paper we present a novel framework for Bayesian structure learning using Model Averaging in large-scale networks, called LSBN (Large-Scale Bayesian Network). In general, the framework follows the principle of divide-and-conquer by partitioning variables into multiple overlapping...
communities, learning intra-community structures individually and merging them together. Specifically, LSBN first performs the partition by using a second-order partition strategy, called ROPART, which is verified to achieve more robust results. Then LSBN proposes a learning algorithm, named LocalLearn, to conduct sampling and structure learning within each overlapping community after the community is isolated from other variables by Markov Blanket. Finally LSBN employs an efficient algorithm, called MERGENCE, to merge structures of overlapping communities into a whole.

In comparison with other state-of-art large-scale network structure learning algorithms such as ARACNE, PC, Greedy Search and MMHC, LSBN shows comparable results in five common benchmark datasets, evaluated by precision, recall and f-score. What’s more, LSBN makes it possible to learn large-scale Bayesian structure by Model Averaging which used to be intractable.

In summary, LSBN provides an scalable and parallel framework for the reconstruction of network structures. Besides, the complete information of overlapping communities serves as the byproduct, which could be used to mine meaningful clusters in biological networks, such as protein-protein-interaction network or gene regulatory network, as well as in social network.

REFERENCES

[1] Jaccard’s similarity coefficient. [http://en.wikipedia.org/wiki/Jaccard_similarity_coefficient](http://en.wikipedia.org/wiki/Jaccard_similarity_coefficient)

[2] Pigs dataset. [http://www.cs.huji.ac.il/labs/compbio/Repository/Datasets/pigs/pigs.htm](http://www.cs.huji.ac.il/labs/compbio/Repository/Datasets/pigs/pigs.htm)

[3] Win95pts dataset. [http://www.cs.huji.ac.il/labs/compbio/Repository/Datasets/win95pts/win95pts.htm](http://www.cs.huji.ac.il/labs/compbio/Repository/Datasets/win95pts/win95pts.htm)

[4] Balázs Adamcsek, Gergely Palla, Illés J. Farkas, Imre Derényi, and Tamás Vicsek. Cfinder: locating cliques and overlapping modules in biological networks. *Bioinformatics*, 22(8):1021–1023, 2006.

[5] Y.-Y. Ahn, J. P. Bagrow, and S. Lehmann. Link communities reveal multiscale complexity in networks. *Nature*, 466:761–764, 2010.

[6] Constantin F. Aliferis, Ioannis Tsamardinos, Alexander R. Statnikov, and Laura E. Brown. Causal explorer: A causal probabilistic network learning toolkit for biomedical discovery. In *METMBS*, pages 371–376, 2003.

[7] J. Baumes, M. Goldberg, M. Magdon-Ismail, and A. Wallace. Discovering hidden groups in communication networks. In *IN PROCEEDINGS OF THE 2ND NSF/NIJ SYMPOSIUM ON INTELLIGENCE AND SECURITY INFORMATICS*, 2004.

[8] I. A. Beinlich, H. J. Suermondt, R. M. Chavez, and G. F. Cooper. The alarm monitoring system: A case study with two probabilistic inference techniques for belief networks. In *Proceedings of the 2nd European Conference on Artificial Intelligence in Medicine*, pages 247–256. Springer-Verlag, 1989.

[9] J. Binder, D. Koller, S. Russell, and K. Kanazawa. Adaptive probabilistic networks with hidden variables. *Machine Learning*, 29(2–3):213–244, 1997.

[10] Laura E. Brown, Ioannis Tsamardinos, and Constantin F. Aliferis. A comparison of novel and state-of-the-art polynomial bayesian network learning algorithms. In *Proceedings of the Twentieth National Conference on Artificial Intelligence (AAAI)*, pages 739–745. AAAI Press, 2005.

[11] Wray Buntine. Theory refinement on bayesian networks. pages 52–60. Morgan Kaufmann, 1991.

[12] Atul J. Butte, Isaac S. Kohane, and I. S. Kohane. Mutual information relevance networks: Functional genomic clustering using pairwise entropy measurements. *Pacific Symposium on Biocomputing*, 5:415–426, 2000.

[13] Gregory F. Cooper and Edward Herskovits. A bayesian method for the induction of probabilistic networks from data. *Machine Learning*, 9:309–347, 1992.

[14] D. Eaton and K. Murphy. Bayesian structure learning using dynamic programming and mcnc. 2007.

[15] N. Friedman and D. Koller. Being bayesian about network structure: A bayesian approach to structure discovery in bayesian networks. *Machine Learning*, 50:95–126, 2003.
[16] Michelle Girvan and M. E. J. Newman. Community structure in social and biological networks. *PROC.NATL.ACAD.SCI.USA*, 99:7821, 2002.

[17] Steve Gregory. Finding overlapping communities in networks by label propagation. *New Journal of Physics*, 12(10), October 2010.

[18] Michael Hecker, Sandro Lambeck, Susanne Töpfer, Eugene P. van Someren, and Reinhard Guthke. Gene regulatory network inference: Data integration in dynamic models - a review. *Biosystems*, 96(1):86–103, 2009.

[19] David Heckerman, Dan Geiger, and David Maxwell Chickering. Learning bayesian networks: The combination of knowledge and statistical data. *Machine Learning*, 20(3):197–243, 1995.

[20] David A. Huffman. A method for the construction of minimum-redundancy codes. *Proceedings of the Institute of Radio Engineers*, 40(9):1098–1101, September 1952.

[21] Claus Skaaning Jensen and Augustine Kong. Blocking gibbs sampling for linkage analysis in large pedigrees with many loops. Research Report R-96-2048, Department of Computer Science, Aalborg University, Denmark, Fredrik Bajers Vej 7, DK-9220 Aalborg Ø, 1996.

[22] Joshua T. Thaden Ilaria Mogno Jamey Wierzbowski Guillaume Cottarel Simon Kasif James J. Collins Timothy S. Gardner Jeremiah J. Faith, Boris Hayete. Large-scale mapping and validation of escherichia coli transcriptional regulation from a compendium of expression profiles. *Public Library of Science Biology*, 5:54–66, 2007.

[23] Mikko Koivisto and Kismat Sood. Exact bayesian structure discovery in bayesian networks. *Journal of Machine Learning Research*, 5:549–573, 2004.

[24] J. Leskovec and C. Faloutsos. Sampling from large graphs. In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 631–636. ACM, 2006.

[25] V. K. Mansinghka, C. Kemp, J. B. Tenenbaum, and T. L. Griffiths. Structured priors for structure learning. *Proceedings of the Twenty-Second Conference on Uncertainty in Artificial Intelligence (UAI 2006)*, 2006.

[26] Adam A. Margolin, Ilya Nemenman, Katia Basso, Chris Wiggins, Gustavo Stolovitzky, Riccardo Dalla Favera, and Andrea Califano. Aracne: An algorithm for the reconstruction of gene regulatory networks in a mammalian cellular context. *BMC Bioinformatics*, 7(S(1)), 2006.

[27] Tams Nepusz, Andrea Petrczi, Lszl Ngyessy, and Flp Bazs. Fuzzy communities and the concept of bridgeness in complex networks. E-print, 2007.

[28] Gergely Palla, Imre Dernyi, Ills Farkas, and Tams Vicsek. Uncovering the overlapping community structure of complex networks in nature and society. *Nature*, 435(7043):814–818, June 2005.

[29] Arvind Rao, Alfred O. Hero, David J. States, and James Douglas Engel. Using directed information to build biologically relevant influence networks.

[30] R. W. Robinson. Counting labeled acyclic digraphs. In F. Harary, editor, *New Directions in Graph Theory*. Academic Press, 1973.

[31] P. Spirites, C. Glymour, and R. Scheines. *Causation, Prediction, and Search*. MIT press, 2nd edition, 2000.

[32] Alexander Strehl and Joydeep Ghosh. Cluster ensembles a knowledge reuse framework for combining partitionings. In *AAAI/IAAI*, pages 93–, 2002.

[33] J. Stuart, E. Segal, D. Koller, and S. Kim. A gene co-expression network for global discovery of conserved genetic modules. *Science*, 302(5643):249–55, October 2003.

[34] Robert L. Thorndike. Who belongs in the family. *Psychometrika*, pages 267–276, 1953.

[35] I. Tsamardinos, C. F. Aliferis, and A. Statnikov. Algorithms for large scale markov blanket discovery. In *Proceedings of the Sixteenth International Florida Artificial Intelligence Research Society Conference*, pages 376–381. AAAI Press, 2003.

[36] Ioannis Tsamardinos, Laura E. Brown, and Constantin F. Aliferis. The max-min hill-climbing bayesian network structure learning algorithm. In *Machine Learning*, page 2006, 2006.

[37] Ian H. Witten and Eibe Frank. *Data Mining: Practical Machine Learning Tools and Techniques*. Morgan Kaufmann, 2005.

[38] Raanan Yehezkel and Boaz Lerner. Bayesian network structure learning by recursive autonomy identification. In *Proceedings of the 2006 joint IAPR international conference on Structural, Syntactic, and Statistical Pattern Recognition, SSPR’06/SPR’06*, pages 154–162, Berlin, Heidelberg, 2006. Springer-Verlag.

[39] Shihua Zhang, Rui-Sheng Wang, and Xiang-Sun Zhang. Identification of overlapping community structure in complex networks using fuzzy c-means clustering. *Physica A: Statistical Mechanics and its Applications*, 374(1):483–490, 2007.