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

Yang Lu[1] , Mengying Wang[2] , Menglu Li[1] , Kenny Q. Zhu[1] and Bo Yuan*[1]

[1]Department of Computer Science, Shanghai Jiao Tong University, Shanghai, China
[2]Software Engineering Institute, East China Normal University, Shanghai, China

Email: Yang Lu - luyang0415@sjtu.edu.cn; Mengying Wang - mywang@sei.ecnu.edu.cn; Menglu Li - iby07@sjtu.edu.cn; Kenny Q. Zhu - kzhu@cs.sjtu.edu.cn; Bo Yuan*- boyuan@sjtu.edu.cn;

*Corresponding author

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 LAMA (Large-Scale Model Averaging), making it possible to handle networks with very large scale by divide-and-conquer.

The method of LAMA comprises three steps. In general, LAMA first performs the partition by using a second-order partition strategy, which achieves more robust results. LAMA conducts sampling and structure learning within each overlapping community after the community is isolated from other variables by Markov Blanket. Finally LAMA 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, LAMA shows comparable results in five common benchmark datasets, evaluated by precision, recall and f-score. What's more, LAMA makes it possible to learn large-scale Bayesian structure by Model Averaging which used to be intractable.

In summary, LAMA 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.

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 synthetic 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 $O(n!2^{\binom{n}{2}})$ possible structures given n network variables [1]. 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 [2]. However, in real applications, ranging from casuality 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 [3], yet subjective factors would inevitably play a nontrivial role and possibly influence the ultimate result. Another widely-applied approach involves prior knowledge [4], 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 [5].

Unfortunately, the above 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 towards the prior knowledge could result, leading to unwarranted structures with less attention to the 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 [6], mutual information [7]. 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 LAMA (**La**rge Scale **M**odel **A**veraging') to learn Bayesian structure for sufficiently large networks. The basic idea of our framework is motivated by the philosophy of divide-and-conquer. Specifically, LAMA 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 [7] and Pearson Coefficient [6], show different partition results given the same data. No one dominate the other, for each one performs well in some datasets and less than satisfactory in others. So *ROPART* introduces a second-order partition strategy, to overcome this shortcoming in a robust way.

• 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.

• We propose an efficient algorithm, called "*MERGE*" to merge the intra-community results into a whole. *MERGE* involves seeking an efficient merge order, and resolves conflicts during the process of merge.

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

Related Works

The problem of static Bayesian Structure Learning is well-studied. Besides 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.

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 [5], such as RELNET [8], ARACNE [7] and CLR [9]. 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 [10]. 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 massive 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 [11] and RAI [12]. The drawback of such algorithms is that number of required 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 [13]. 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 [14] (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.

Much work has been devoted to detecting overlapped communities in large networks. The most popular algorithm is Clique Percolation Method (CPM) [15] which computes all k cliques and two variables belong to the same cluster if there exists a path going through *k-1* cliques between them. CPM is implemented by CFinder [16] (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 optimization given quality function W, thus detecting overlapping communities becomes finding all locally-optimized subgraphs [17]. Furthermore, the optimization could be augmented by combination with spectral mapping and fuzzy clustering [18].

Clustering approaches define clusters as either the set of nodes or the set of edges, and then perform the clustering according to the similarity among nodes [19] or edges [20], respectively. Besides, clustering could be conducted in an agglomerative hierarchy, for example, LinkComm [20] merges groups of edges pairwise in descending order of edge similarity and consequently achieve a dendrogram.

Partitioning approaches transform original graph into a larger graph without overlapping nodes before conduct traditional partitions. Those overlapping nodes are identified and split into multiple copies of themselves beforehand [21]. The identification of candidate overlapping nodes is based on *split betweenness* [22], and the splitting process continues as long as the *split betweenness* of variables is sufficiently high.

Definitions

Definition 1 (Weight Function)

Given a generic undirected, unweighted graph $\mathcal{G} = (V, E)$, where $E \subseteq V \times V$. Weight Function f maps any edges $e = (u, v) \in E$ to a numeric value:

$$f \colon E \mapsto \mathbb{R}, \quad (u, v) \mapsto f(u, v)$$

Generally speaking, weight functions play the role to map an unweighted graph into a weighted one. Given a weight function set consisting of n weight functions, $F = \{f_1, ..., f_n\}$, a generic unweighted graph G could be mapped to a weighted graph set $D = \{G_1, ...G_n\}$, where $G_i = (V, E_i)$.

Definition 2 (Partition)

Given an undirected, weighted graph $\mathcal{G} = (V, E)$, $P = \{p_1, ..., p_C\}$ is a partition of the edges into C communities. Each node $v \in V$ belongs to at least one communities, even an isolated node would itself constitute a community of single member. Communities as the result could be partitioned again recursively thus grouped into a hierarchical structure.

Definition 3 (Partition Support Matrix)

Given a weighted graph set $D = \{G_1, ...G_n\}$, where $G_i = (V, E_i)$, each weighted graph G_i corresponds to a partition P_i. The partition support matrix of a node v in partition P_i is denoted as $PSM_i(v)$. $PSM_i(v)$ is a $|V| \times m$ matrix where $|V|$ is the node number of graphs and m is the number of communities in partition P_i that contains node v. The element in *j-th* row and *k-th* column of $PSM_i(v)$ denotes whether *k-th* node exists in *j-th* communities which contains node v in partition P_i. The partition support matrix of a node v for all partitions, written $PSM(v)$, is defined as:

$$PSM(v) = \begin{pmatrix} PSM_1(v) \\ PSM_2(v) \\ ... \\ PSM_n(v) \end{pmatrix}$$

For example, the partition support matrix of the $node_7$ for partition P_1 in Figure 1a is $PSM_1(node_7) = [[0,0,0,1,0,0,1,0,0]^T, [0,0,0,0,0,0,1,1,1]^T]^T$, while for partition P_2 in Figure 1b is $PSM_2(node_7) = [0,0,0,1,0,0,1,1,1]$. Then the overall partition support matrix of $node_7$ is

$$PSM(node_7) = \begin{pmatrix} 0,0,0,1,0,0,1,0,0 \\ 0,0,0,0,0,0,1,1,1 \\ 0,0,0,1,0,0,1,1,1 \end{pmatrix}$$

As one can see, $node_7$ tends to be grouped together with $node_4$, $node_8$ and $node_9$.

Definition 4 (Second-Order Network)

Given a weighted graph set $D = \{G_1, ...G_n\}$, where $G_i = (V, E_i)$, the corresponding partition set is $P = \{P_1, ...P_n\}$ and the Partition Support Matrix for each node v is $PSM(v)$. The second-order network is an undirected, weighted graph $G_S = (V, E_S)$ where an edge $e = (u, v) \in E_S$ if its edge weight exceeds a certain threshold. The edge weight is valued by the average joint probability between u and v in Partition Support Matrix $PSM(u)$ and $PSM(v)$.

Definition 5 (Second-Order Partition)

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

LAMA system: Large-Scale Bayesian Network Learning

The LAMA system provides a novel framework for Bayesian structure learning using model averaging in large-scale networks. The LAMA 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 LAMA system is as follows:

(1) Variable Partition.

(2) Sampling and Learning.

(3) Merge.

Variable Partition

One of the key requirements of LAMA is that it scales well with the number of variables. 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 belongs 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.

Algorithm 1 ROPART

1: build an undirected, unweighted complete graph $\mathcal{G}$ given all variables;
2: for each weight function f_i from a predefined weight function set $F = \{f_1, ..., f_n\}$, map $\mathcal{G}$ to a weighted graph $\mathcal{G}_i$ respectively;
3: for each weighted graph $\mathcal{G}_i$,keep those edges of which the weight exceed some truncate threshold $\mathcal{T}_{trunc}$, 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;

The initial input of the LAMA 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.

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 $D = \{\mathcal{G}_1, ...\mathcal{G}_n\}$, where f_i corresponds to $\mathcal{G}_i$. 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 [23]

$$MI_{plus}(X,Y) = \frac{2MI(X,Y)}{H(X)+H(Y)}$$

• Mutual Information normalized by the square root product of entropies [24]

$$MI_{sqrt}(X,Y) = \frac{MI(X,Y)}{\sqrt{H(X)}\sqrt{H(Y)}}$$

• Mutual Information normalized by PageRank weight

$$MI_{pr}(X,Y) = \frac{MI(X,Y)}{\sqrt{PR(X)}\sqrt{PR(Y)}}$$

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

• Mutual Information by Standard Normalization

$$MI_{sn}(X,Y) = \frac{MI(X,Y) - \mu_{MI}}{\sigma_{MI}}$$

where μ_{MI} and σ_{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 μ_X and μ_Y denote the mean value of variable X and Y respectively, and σ_X and σ_Y indicate the standard deviation of X and Y correspondingly.

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

In **Step 3**, we partition each weighted graph $\mathcal{G}_i$ after pruning. For the sake of convenience in Mergence, 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 [20] for partition algorithm, which is introduced in Section .

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 $\mathcal{G}_i$.

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_{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.

Sampling and Learning

The LAMA 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

```
for each community C ∈ P_sec do
    identify candidate Markov Blanket MB(C)
    sample k sub-communities based on C and MB(C), SC = {sub_1(C), ..., sub_k(C)}
    for each sub-community sc_i ∈ SC do
        learn Bayesian structure
    end for
    ensemble structures of all sub-communities and resolve conflicts
end for
```

In **Step 1**, we try to find out the Markov Blanket $MB(\mathcal{C})$ for community $\mathcal{C}$. Conditioned on Markov Blanket, no other nodes outside the community $\mathcal{C}$ could influence nodes within the community. To be more precise, $\forall X \in C, X \perp Y | MB(X)$ for all $Y \notin \{X \cup MB(X)\}$. Therefore knowledge of Markov Blanket $MB(\mathcal{C})$ 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, $MB_{cand}(X) = \{Y | w_{XY} \geq \overline{w}_X\}$, where w_{XY} is the weight of edge e_{XY} and $\overline{w}_X$ is the average weight of edges connecting X.

(2) Given $MB_{cand}(X)$, we employ a scalable algorithm IAMB [25] for further justification. IAMB [25](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 $\mathcal{C}$ with its Markov Blanket $MB(\mathcal{C})$ into a expanded community $\mathcal{C}'$. The size of $\mathcal{C}'$ 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 $\mathcal{C}$ as well.

Given expanded community $\mathcal{C}'$, we conduct sampling for two reasons: (1) the size of $\mathcal{C}'$ may still 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) [26] with some modifications.

In our sampling algorithm, we build up an inner markov graph $\mathcal{G}_{IM}(\mathcal{C})$ based on $\mathcal{C}$, to be precise, $\mathcal{G}_{IM}(\mathcal{C}) = (\mathcal{C}, E)$ 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 $\mathcal{G}_{IM}(\mathcal{C})$ as starting node at random together with its neighbors, denoted by $\mathcal{S}$. The final sub-community for Bayesian structure learning is $\mathcal{S} \cup MB(\mathcal{S})$, if the size of ultimate sub-community is still too large to learn, we keep removing neighbors within $\mathcal{S}$ until acceptable size.

In **Step 3**, for each sub-community, LocalLearn use 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 $f(\mathcal{G}) = 1$ denotes whether there exists an edge from node i to node j and $f(\mathcal{G}) = 0$ otherwise. Usually posterior expectation of feature $f(\mathcal{G})$ is estimated by:

$$P(f|D) = \sum_{G} f(G)p(G|D) \tag{1}$$

$f(\mathcal{G})$ is classified to be 1 if $E(f|D)$ exceed certain threshold $\mathcal{T}_{avg}$ and 0 otherwise. However, traversing all candidate DAGs according to Eq.(1) is usually intractable for there are overall $O(n!2^{\binom{n}{2}})$ DAGs given n nodes. One solution is to average over structures which conform to some predetermined node order $\prec$ [27]. For example, if $X_i \in Pa_G(X_j)$ then $i \prec j$. Now the posterior expectation of feature $f(\mathcal{G})$ given a predetermined order $\prec$ could be transformed into [28]:

$$P(f|D) = \sum_{\prec} P(\prec |D)P(f|D, \prec) \tag{2}$$

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 [28] [29]. 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 $\prec_1, ..., \prec_T$ is obtained. Now the expectation of feature f could be approximately estimated as [28]:

$$P(f|D) \approx \frac{1}{T} \sum_{t=1}^{T} P(f|D, \prec_t) \tag{3}$$

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:

• **Additional edges** due to indirect interactions. This type of error is introduced by ARACNE [7]. ARACNE [7] tries to eliminate indirect edges by using Data Processing Inequality in all triplets of variables. However, ARACNE fails to eliminate all indirect edges and keep all valid edges at the same time for two reasons: (1) it is hardly to find fixed threshold to distinguish indirect interactions from direct ones; (2) In many cases, weakest edges in triplets do not necessarily indicate indirect interaction.

• **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 [7]. 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 [20] for graph clustering. After clustering the candidate triplet graph, we re-learn the structure for each cluster using the same method in Step 3.

Merge

The LAMA system combines structures after learning individually from each community. The combination involves two concerns: (1) to find an efficient mergen order; (2) to resolve the conflicts during the merge. Intuitively, the merge strategy should proceed as a bottom-up approach. The LAMA 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 merge.

Inspired by the idea of Huffman's Algorithm [30], the LAMA system tries to merge communities in a greedy strategy by constantly picking two communities with maximum Jaccard similarity coefficient [31]. Jaccard similarity coefficient [31] 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.

Algorithm 3 Merge

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 $\mathcal{C}_i$ and $\mathcal{C}_j$ of maximum Jaccard similarity coefficient $\mathcal{J}(\mathcal{C}_i, \mathcal{C}_j)$ from the Community Pool;
4: Merge structures of two communities $\mathcal{C}_i$ and $\mathcal{C}_j$;
5: Create a new node $\mathcal{C}_{new} = \mathcal{C}_i \bigcup \mathcal{C}_j$ denoting the merge of two communities $\mathcal{C}_i$ and $\mathcal{C}_j$;
6: Add the new node to the Community Pool;
7: **end while**

At first, Mergence Algorithm put all communities into a Community Pool, then repeatedly choose two communities $\mathcal{C}_i$ and $\mathcal{C}_j$ with largest Jaccard similarity coefficient $\mathcal{J}(\mathcal{C}_i, \mathcal{C}_j) = \frac{|\mathcal{C}_i \bigcap \mathcal{C}_j|}{|\mathcal{C}_i \bigcup \mathcal{C}_j|}$. After two communities are selected, same approach described in step 3 of Section 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 merge steps.

In each iteration, assuming there are k communities left in the Community Pool, then it would take $\binom{k}{2}$ 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: $\sum_{k=2}^{n} \binom{k}{2} = \frac{1}{2}[\sum_{k=2}^{n} k^2 - \sum_{k=2}^{n} k] = \frac{n(n+1)(n-1)}{6}$. 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: $\binom{n}{2} + \sum_{k=2}^{n}[2(k-1) + 1 + (k-2)] = 2n(n-1)$.

Experiments

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

Datasets

We benchmark evaluation of Lama on five well-known datasets, shown as follows:

• alarm [32]. 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 [33]. 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 [34]. 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 [35]. 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 [36]. 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 '$Pearson_{sn}$'.

Parameter Setting

For each weighted network generated by certain weight function, the truncate threshold $\mathcal{T}_{trunc}$ for pruning is chosen by the Elbow Method [37]. 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 $\mathcal{T}_{trunc}$, also called 'elbow criterion'. The truncate thresholds we chosen are shown in Table 1, the percentage denotes the ratio between remaining edges and all edges in complete graph. From the results in Table 1, there is no significant difference in numbers of remaining edges.

We measure the average shortest path (Table 2) and diameter (Table 3) 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 excellently in some datasets but poorly 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 2), and the average ranking in diameter is 2 (Table 3).

The partition results are depicted in Table 4, 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.

Experimental Design

Our evaluation compares LAMA against four state-of-art large-scale network structure learning algorithms, namely ARACNE [7], PC [11], Greedy Search [13] and Max-Min Hill Climbing (MMHC) [14]. Among these common algorithms, ARACNE [7] is a very popular information-theoretic algorithm with extreme simplicity and low computational cost; the PC algorithm [11] is considered as the most popular constrain-based algorithm; Greedy Search [13] are very widely-used score-and-search approaches; and MMHC [14] serves as a hybrid method which proves to be superior to other algorithms in most cases.

We compare LAMA 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 [38] (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% by default for MMHC and Greedy Search; threshold on mutual information test is 1% by default for PC; prior type is chosen to be BDeu score [39] with Dirichlet Weight equals 10 for Greedy Search and MMHC. As for ARACNE, we implemented the algorithm manually due to its simplicity, and the low values threshold τ are selected manually based on tradeoff between true positives and false positives, shown in Table 5.

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

As for the evaluation of LAMA framework, we slightly change LAMA 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 LAMA whose structure learning algorithm in LocalLearn is also MMHC while keep everything else unchanged. As a result, the performance of LAMA framework could be measured independently, regardless of influence brought from the use of different structure learning algorithms.

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 LAMA system. These metrics are defined as follows:

$$Precision = TP/(TP + FP)$$

$$Recall = TP/(TP + FN)$$

$$F - Score = \frac{2 * Precision * Recall}{Precision + Recall}$$

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 LAMA and other state-of-art algorithms such as ARACNE, PC, Greedy Search and MMHC are depicted in Table 6 on Alarm Dataset, Table 7 on Insurance Dataset, Table 8 on Win95pts Dataset, Table 9 on Pigs DataSet and Table 10 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 LAMA 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 LAMA is slight worse off than global in benchmark algorithms such as ARACNE, PC and MMHC, but better in Greedy Search (Table 11). The recall of LAMA shows superiority to global in ARACNE and PC, but little inferiority in Greedy Search and MMHC (Table 12). 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 13). The results of precision, recall and F-Score reveal that LAMA 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, LAMA 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.

Conclusion

In this paper we present a novel framework for Bayesian structure learning using Model Averaging in large-scale networks, called LAMA(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, LAMA first performs the partition by using a second-order partition strategy, called *ROPART*, which is verified to achieve more robust results. Then LAMA 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 LAMA employs an efficient algorithm, called *MERGENCE*, 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, LAMA shows comparable results in five common benchmark datasets, evaluated by precision, recall and f-score. What's more, LAMA makes it possible to learn large-scale Bayesian structure by Model Averaging which used to be intractable.

In summary, LAMA 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.

Author's contributions

All authors contributed equally to this work.

Acknowledgements

We thank Microsoft China Cloud Innovation Lab for providing us private cloud to perform parallel computing. We also thank Justin Ding from Microsoft Research Asia for providing valuable suggestions and comments.

References

1. Robinson RW: **Counting labeled acyclic digraphs**. In *New Directions in Graph Theory*. Edited by Harary F, Academic Press 1973.
2. Heckerman D, Geiger D, Chickering DM: **Learning Bayesian Networks: The Combination of Knowledge and Statistical Data**. *Machine Learning* 1995, **20**(3):197–243.
3. Koivisto M, Sood K: **Exact Bayesian Structure Discovery in Bayesian Networks**. *Journal of Machine Learning Research* 2004, **5**:549–573.
4. Mansinghka VK, Kemp C, Tenenbaum JB, Griffiths TL: **Structured priors for structure learning.** *Proceedings of the Twenty-Second Conference on Uncertainty in Artificial Intelligence (UAI 2006)* 2006, [http://cocosci.berkeley.edu/tom/papers/structure.pdf].
5. Hecker M, Lambeck S, Töpfer S, van Someren EP, Guthke R: **Gene regulatory network inference: Data integration in dynamic models - A review**. *Biosystems* 2009, **96**:86–103.
6. Stuart J, Segal E, Koller D, Kim S: **A Gene Co-Expression Network for Global Discovery of Conserved Genetic Modules**. *Science* 2003, **302**(5643):249–55.
7. Margolin AA, Nemenman I, Basso K, Wiggins C, Stolovitzky G, Favera RD, Califano A: **ARACNE: An Algorithm for the Reconstruction of Gene Regulatory Networks in a Mammalian Cellular Context**. *BMC Bioinformatics* 2006, **7**(S-1).
8. Butte AJ, Kohane IS, Kohane IS: **Mutual Information Relevance Networks: Functional Genomic Clustering Using Pairwise Entropy Measurements**. *Pacific Symposium on Biocomputing* 2000, **5**:415–426.
9. Jeremiah J Faith JTTIMJWGCSKJJCTSG Boris Hayete: **Large-Scale Mapping and Validation of Escherichia coli Transcriptional Regulation from a Compendium of Expression Profiles**. *Public Library of Science Biology* 2007, **5**:54–66.
10. Rao A, Hero AO, States DJ, Engel JD: **USING DIRECTED INFORMATION TO BUILD BIOLOGICALLY RELEVANT INFLUENCE NETWORKS**.
11. Spirtes P, Glymour C, Scheines R: *Causation, Prediction, and Search*. MIT press, 2nd edition 2000.
12. Yehezkel R, Lerner B: **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, Berlin, Heidelberg: Springer-Verlag 2006:154–162, [http://dx.doi.org/10.1007/11815921_16].
13. Brown LE, Tsamardinos I, Aliferis CF: **A comparison of novel and state-of-the-art polynomial bayesian network learning algorithms**. In *In Proceedings of the Twentieth National Conference on Artificial Intelligence (AAAI*, AAAI Press 2005:739–745.
14. Tsamardinos I, Brown LE, Aliferis CF: **The max-min hill-climbing bayesian network structure learning algorithm**. In *Machine Learning* 2006:2006.
15. Palla G, Dernyi I, Farkas I, Vicsek T: **Uncovering the overlapping community structure of complex networks in nature and society**. *Nature* 2005, **435**(7043):814–818, [http://dx.doi.org/10.1038/nature03607].
16. Adamcsek B, Palla G, Farkas IJ, Derényi I, Vicsek T: **CFinder: locating cliques and overlapping modules in biological networks**. *Bioinformatics* 2006, **22**(8):1021–1023.
17. Baumes J, Goldberg M, Magdon-Ismail M, Wallace A: **Discovering Hidden Groups in Communication Networks**. In *IN PROCEEDINGS OF THE 2ND NSF/NIJ SYMPOSIUM ON INTELLIGENCE AND SECURITY INFORMATICS* 2004.
18. Zhang S, Wang RS, Zhang XS: **Identification of Overlapping Community Structure in Complex Networks Using Fuzzy c-means Clustering**. *Physica A: Statistical Mechanics and its Applications* 2007, **374**:483–490.
19. Nepusz T, Petrczi A, Ngyessy L, Bazs F: **Fuzzy communities and the concept of bridgeness in complex networks**. *E-print* 2007, [http://www.arxiv.org/abs/0707.1646].
20. Ahn YY, Bagrow JP, Lehmann S: **Link communities reveal multiscale complexity in networks**. *Nature* 2010, **466**:761–764.

21. Gregory S: **Finding overlapping communities in networks by label propagation**. *New Journal of Physics* 2010, **12**(10).

22. Girvan M, Newman MEJ: **Community structure in social and biological networks**. *PROC.NATL.ACAD.SCI.USA* 2002, **99**:7821, [http://www.citebase.org/cgi-bin/citations?id=oai:arXiv.org:cond-mat/0112110].

23. Witten IH, Frank E: *Data Mining: Practical Machine Learning Tools and Techniques*. Morgan Kaufmann 2005.

24. Strehl A, Ghosh J: **Cluster Ensembles A Knowledge Reuse Framework for Combining Partitionings**. In *AAAI/IAAI* 2002:93–.

25. Tsamardinos I, Aliferis CF, Statnikov A: **Algorithms for Large Scale Markov Blanket Discovery**. In *Proceedings of the Sixteenth International Florida Artificial Intelligence Research Society Conference*, AAAI Press 2003:376–381.

26. Leskovec J, Faloutsos C: **Sampling from large graphs**. In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, ACM 2006:631–636, [http://scholar.google.de/scholar.bib?q=info:2FlyT6PW7ucJ:scholar.google.com/&output=citation&hl=de&as_sdt=0,5&ct=citation&cd=0].

27. Cooper GF, Herskovits E: **A Bayesian Method for the Induction of Probabilistic Networks from Data**. *Machine Learning* 1992, **9**:309–347.

28. Friedman N, Koller D: **Being Bayesian about Network Structure: A Bayesian Approach to Structure Discovery in Bayesian Networks**. *Machine Learning* 2003, **50**:95–126.

29. Eaton D, Murphy K: **Bayesian structure learning using dynamic programming and MCMC** 2007[http://www.cs.ubc.ca/~murphyk/Papers/eaton-uai07.pdf].

30. Huffman DA: **A Method for the Construction of Minimum-Redundancy Codes**. *Proceedings of the Institute of Radio Engineers* 1952, **40**(9):1098–1101.

31. **Jaccard's similarity coefficient**. *http://en.wikipedia.org/wiki/Jaccard_similarity_coefficent*[http://en.wikipedia.org/wiki/Jaccard_similarity_coefficent].

32. Beinlich IA, Suermondt HJ, Chavez RM, Cooper GF: **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*, Springer-Verlag 1989:247–256, [http://www.cs.huji.ac.il/site/labs/compbio/Repository/Datasets/alarm/alarm.htm].

33. Binder J, Koller D, Russell S, Kanazawa K: **Adaptive Probabilistic Networks with Hidden Variables**. *Machine Learning* 1997, **29**(2–3):213–244, [http://www.cs.huji.ac.il/site/labs/compbio/Repository/Datasets/insurance/insurance.htm].

34. **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].

35. **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].

36. Jensen CS, Kong A: **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 Ø1996.

37. Thorndike RL: **Who belongs in the family**. *Psychometrika* 1953, :267–276.

38. Aliferis CF, Tsamardinos I, Statnikov AR, Brown LE: **Causal Explorer: A Causal Probabilistic Network Learning Toolkit for Biomedical Discovery**. In *METMBS* 2003:371–376.

39. Buntine W: **Theory Refinement on Bayesian Networks**. In *METMBS*, Morgan Kaufmann 1991:52–60.

Figures

Figure 1 - Two different partition on the same set of variables

Figure 2 - ROPART: a robust partition strategy combining multiple different weight functions

Figure 3 - LocalLearn: a bootstrap strategy for bayesian structure learning per community

Tables

Table 1 - Truncate threshold for each weight function on each dataset

	insurance		alarm		win95pts		pigs		link	
	$\mathcal{T}_{trunc}$	percent	$\mathcal{T}_{trunc}$	percent	$\mathcal{T}_{trunc}$	percent	$\mathcal{T}_{trunc}$	percent	$\mathcal{T}_{trunc}$	percent
MI	0.08	0.2308	0.01	0.2267	0.007	0.1659	0.005	0.0841	0.035	0.0566
MI_{plus}	0.2	0.2792	0.03	0.2312	0.05	0.1635	0.05	0.0958	0.4	0.0435
MI_{sqrt}	0.5	0.3646	0.5	0.2132	0.4	0.1618	2	0.0726	2.5	0.0493
MI_{pr}	0.05	0.3276	0.02	0.2267	0.013	0.1687	0.005	0.0762	0.5	0.0510
MI_{sn}	0	0.2707	0	0.2072	0.05	0.1368	0.8	0.0861	0.6	0.0441
Pearson	0.1	0.4017	0.4	0.2747	0.02	0.1894	0.026	0.1516	0.075	0.0585
$Pearson_{sn}$	0	0.3789	0	0.2087	0.75	0.1175	0	0.0731	1.0	0.0490

Table 2 - Comparison of averaging shortest path for various weight functions

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

	insurance	alarm	win95pts	pigs	link	$ranking_{avg}$
MI	1.5333(3)	3.8162(1)	2.9799(6)	5.2616(4)	2.6427(4)	3.6
MI_{plus}	1.55(4)	4.1567(8)	3.1130(8)	5.4789(5)	2.4552(1)	5.2
MI_{sqrt}	1.9456(8)	3.8792(3)	3.0156(7)	5.6248(7)	2.4622(2)	5.4
MI_{pr}	1.7983(6)	3.8567(2)	2.5521(1)	5.1297(3)	3.7409(6)	3.6
MI_{sn}	1.4927(2)	3.9682(7)	2.8828(5)	5.5749(6)	4.4982(8)	5.6
Pearson	1.7357(5)	3.8991(5)	2.7558(2)	4.4463(2)	2.6801(5)	3.8
$Pearson_{sn}$	1.8430(7)	3.9178(6)	2.8380(4)	5.6633(8)	4.0882(7)	6.4
second-order	1.36(1)	3.8875(4)	2.8240(3)	4.28198(1)	2.5624(3)	**2.4**

Table 3 - Comparison of averaging Diameters for various weight functions

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

	insurance	alarm	win95pts	pigs	link	$ranking_{avg}$
MI	3.0(3)	6.1667(2)	5.1429(6)	10.6875(4)	5.0026(8)	4.6
MI_{plus}	3.0(3)	6.8(7)	5.4(8)	11.2247(5)	4.3666(4)	5.4
MI_{sqrt}	4.0(8)	6.8333(8)	5.1667(7)	11.672(8)	4.3480(3)	6.8
MI_{pr}	3.6(6)	6.2(3)	4.05(1)	9.4707(3)	4.6027(6)	3.8
MI_{sn}	2.7143(2)	6.7143(6)	4.6857(4)	11.4805(7)	4.4982(5)	4.8
Pearson	3.33(5)	6.1(1)	4.5556(2)	8.3585(2)	4.9513(7)	3.4
$Pearson_{sn}$	3.778(7)	6.625(5)	4.886(5)	11.3907(6)	4.0882(2)	5
second-order	2.423(1)	6.4211(4)	4.619(3)	7.7565(1)	3.7409(1)	**2**

Table 4 - 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	6	0	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 5 - Low values threshold τ in ARACNE implementation for all Datasets

Dataset Name	low values threshold τ
alarm	0.01
insurance	0.05
win95pts	0.025
pigs	0.01
link	0.005

Table 6 - Evaluation of LAMA against other algorithms on Alarm Dataset

	LAMA						Global					
	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 7 - Evaluation of LAMA against other algorithms on Insurance Dataset

	LAMA						Global					
	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	85.455
MMHC	43	9	4	91.489	82.692	86.869	43	9	2	95.556	82.692	88.660
Model Avg	45	7	7	86.538	86.538	86.538	NA	NA	NA	NA	NA	NA

Table 8 - Evaluation of LAMA against other algorithms on Win95pts Dataset

	LAMA						Global					
	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	88.889	57.143	69.565	38	74	3	92.683	33.929	49.673
Greedy	99	13	143	40.909	88.393	55.932	94	18	106	47.000	83.929	60.256
MMHC	92	20	56	62.162	82.143	70.769	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 9 - Evaluation of LAMA against other algorithms on Pigs Dataset

	LAMA						Global					
	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

Table 10 - Evaluation of LAMA against other algorithms on Link Dataset

	LAMA						Global					
	Hit(TP)	Miss(FN)	Error(FP)	Precision	Recall	F-Score	Hit(TP)	Miss(FN)	Error(FP)	Precision	Recall	F-Score
ARACNE	422	703	338	55.526	37.511	44.775	444	681	280	61.326	39.467	48.026
PC	466	659	277	62.719	41.422	49.893	70	1055	26	72.917	6.222	11.466
Greedy	413	712	342	54.702	36.711	43.936	783	342	1374	36.300	69.600	47.715
MMHC	474	651	321	59.623	42.133	49.375	621	504	418	59.769	55.200	57.394
Model Avg	408	717	413	49.695	36.267	41.932	NA	NA	NA	NA	NA	NA

Table 11 - Significance of LAMA with respect to Precision normalized by the Precision of global situation for four state-of-art structure learning algorithms.

Dataset	ARACNE	PC	Greedy	MMHC
alarm	97.850%	100%	99.952%	95.602%
insurance	96.324%	95.286%	107.583%	95.744%
win95pts	77.689%	95.906%	87.040%	84.265%
pigs	97.855%	101.354%	105.532%	99.653%
link	90.542%	86.014%	150.694%	99.756%

Table 12 - Significance of LAMA with respect to Recall normalized by the Recall of global situation for four state-of-art structure learning algorithms.

Dataset	ARACNE	PC	Greedy	MMHC
alarm	125.808%	108.572%	88.637%	97.727%
insurance	132.001%	116.130%	72.341%	100%
win95pts	152.831%	168.419%	105.319%	102.223%
pigs	100%	97.123%	96.284%	96.959%
link	95.044%	665.734%	52.746%	76.328%

Table 13 - Significance of LAMA with respect to F-Score normalized by the F-Score of global situation for four state-of-art structure learning algorithms.

Dataset	ARACNE	PC	Greedy	MMHC
alarm	111.982%	104.693%	94.355%	96.665%
insurance	117.168%	107.196%	95.957%	97.980%
win95pts	113.964%	140.046%	92.824%	92.000%
pigs	98.914%	99.206%	100.872%	98.288%
link	93.231%	435.139%	92.080%	86.028%