Local-to-Global Bayesian Network Structure Learning

Tian Gao [1] **Kshitij Fadnis** [1] **Murray Campbell** [1]

Abstract

We introduce a new local-to-global structure learning algorithm, called graph growing structure learning (GGSL), to learn Bayesian network (BN) structures. GGSL starts at a (random) node and then gradually expands the learned structure through a series of local learning steps. At each local learning step, the proposed algorithm only needs to revisit a subset of the learned nodes, consisting of the local neighborhood of a target, and therefore improves on both memory and time efficiency compared to traditional global structure learning approaches. GGSL also improves on the existing local-to-global learning approaches by removing the need for conflict-resolving AND-rules, and achieves better learning accuracy. We provide theoretical analysis for the local learning step, and show that GGSL outperforms existing algorithms on benchmark datasets. Overall, GGSL demonstrates a novel direction to scale up BN structure learning while limiting accuracy loss.

1. Introduction

Bayesian networks have been used in classification (Aliferis et al., 2010), feature selection (Gao et al., 2015), latent variable discovery (Lazic et al., 2013; Gao & Ji, 2016a), and knowledge discovery (Spirtes et al., 1999; Gao & Ji, 2015) in various domains (Ott et al., 2004). However, due to its NP-hard nature (Chickering et al., 2012), exact BN structure learning on directed acyclic graphs (DAG) faces scalability issues.

In this paper, we consider a local-to-global approach to learn the Bayesian network structure, starting from the local graph structure of one node and then gradually expanding the graph based on already learned structures.

[1]IBM Thomas J. Watson Research Center, Yorktown Heights, NY 10598 USA. Correspondence to: Tian Gao <tgao@us.ibm.com>.

Proceedings of the 34^{th} International Conference on Machine Learning, Sydney, Australia, PMLR 70, 2017.

The predominant exact score-based structure learning algorithms adopt the global approach and focus on better scoring criteria (Acid et al., 2005; Brenner & Sontag, 2013) or more efficient search procedures (Chickering, 2002; Koivisto & Sood, 2004; Silander & Myllymaki, 2006; Jaakkola et al., 2010; Cussens, 2011; Yuan & Malone, 2013) to navigate the intractable search space of possible directed acyclic graphs over all the variables present. Despite such progress, the practical usage of these algorithms is still limited when there are large numbers of variables. Many approximations (Scanagatta et al., 2015), constraints (de Campos et al., 2009; Chen et al., 2016), and assumptions (Nie et al., 2014) are utilized to alleviate time and memory complexity.

Instead of searching the entire DAG space for all the variables at the same time, the local-to-global approach limits the size of the space by learning a local structure with only a limited number of variables. These variables consists of potential candidates for local structures, usually defined by the parent-child (PC) or the Markov Blanket (MB) (Pearl, 1988) set of a target node in a DAG. Many local learning algorithms (Koller & Sahami, 1996; Tsamardinos et al., 2003; Fu & Desmarais, 2008) iteratively query new variables to update and learn the local structure, either PC, MB set or both, for one specific target variable. Many constraint-based and score-based local learning algorithms have been proposed and shown to have promising performances in practice (Aliferis et al., 2010). Using learned local structures, some prior works have proposed to combine the local structures for the global structure. Several works (Margaritis & Thrun, 1999; Pellet & Ellisseeff, 2008) have proposed algorithms to identify MBs of every node in the graph first, and then connect the MBs in a maximally consistent way to learn the global structure of a BN. Both constraint-based (Tsamardinos et al., 2006) and score-based local-to-global structure learning methods (Niinimaki & Parviainen, 2012) have been proposed. This local-to-global approach has the benefit of improving the exact structure learning efficiency, as at each step only a small number of variables are expected to be used, although the accuracy has not been very competitive.

We aim to improve the accuracy of the local-to-global approach and propose a new local-to-global structure learning algorithm. The algorithm starts the local learning at one

variable first, and then iteratively applies the local learning procedure to its neighbors and so on, gradually expanding the learned graph with minimal repeated learning. GGSL efficiently grows local graphs to global graphs without considering the traditional the AND-rule, or a consistency check on learned local neighborhoods. It uses only the necessary variables for each local learning step to learn and resolve any possible conflicts among local structures, hence improving efficiency over global learning algorithms and improving accuracy over existing local-to-global algorithms with the AND-rule.

Notation: We use capital letters (such as X, Y) to represent variables, small letters (such as x, y) to represent values of variables, and bold letters (such as $\mathbf{V}, \mathbf{MB}$) to represent variable sets. $|\mathbf{V}|$ represents the size of a set $\mathbf{V}$. $X \perp\!\!\!\perp Y$ and $X \not\perp\!\!\!\perp Y$ represent independence and dependence between X and Y, respectively.

2. Technical Preliminaries

Let $\mathbf{V}$ denote a set of random variables. A Bayesian Network for $\mathbf{V}$ is represented by a pair (G, θ). The network structure G is a directed acyclic graph with nodes corresponding to the random variables in $\mathbf{V}$. If a directed edge exists from node X to node Y in G, X is a *parent* of Y and Y is a *child* of X. The parameters θ indicate the conditional probability distribution of each node $X \in \mathbf{V}$ given its parents. Moreover, let a *path* between two nodes X and Y in G be any sequence of nodes between them such that any successive nodes are connected by a directed edge, and no node appears in the sequence twice. A *directed path* of a DAG is a path with nodes $(V_1, ..., V_n)$ such that, for $1 \leq i < n$, V_i is a parent of V_{i+1}. If there is a directed path from X to Y, then X is an *ancestor* of Y and Y is a *descendant* of X. If X and Y have a common child and they are not adjacent, X and Y are *spouses* of each other. Three nodes X, Y, and Z form a *V-structure* if node Y has two incoming edges from X and Z, forming $X \rightarrow Y \leftarrow Z$, and X is not adjacent to Z. Y is a *collider* if Y has two incoming edges from X and Z in a path. A path $\mathbf{J}$ from node X to Y is *blocked* by a set of nodes $\mathbf{Z}$, if any of following holds true: 1) There is a non-collider node in $\mathbf{J}$ belonging to $\mathbf{Z}$. 2) There is a collider node C on $\mathbf{J}$ such that neither C nor any of its descendants belong to Z. Otherwise, $\mathbf{J}$ from X to Y is unblocked or active.

The *Local Markov Condition* (Pearl, 1988) states a node in a BN is independent of its non-descendant nodes, given its parents. It enables the recovery of a distribution $\mathcal{P}$ (in term of independence relationships) from a known DAG G. A DAG G and a joint distribution $\mathcal{P}$ are *faithful* to each other if all and only the conditional independencies true in $\mathcal{P}$ are entailed by G (Pearl, 1988). The faithfulness condition enables us to recover a DAG G from a distribution $\mathcal{P}$ to completely characterize $\mathcal{P}$.

A *Markov Blanket* of a target variable T, $\mathbf{MB}_T$, is the minimal set of nodes conditioned on which all other nodes are independent of T, denoted as $X \perp\!\!\!\perp T|\mathbf{MB}_T, \forall X \in \{\mathbf{V} \setminus T \setminus \mathbf{MB}_T\}$. Given independently and identically distributed (i.i.d.) samples D from an unknown distribution $\mathcal{P}$, represented by a faithful but unknown DAG G^0 to $\mathcal{P}$, *local structure learning* is to find the PC or MB of a target node in G^0. To avoid symbol confusion, we use G to represent any learned DAG and use G^0 to represent the ground truth DAG. Under the faithfulness assumption between G^0 and $\mathcal{P}$, the PC and MB of a target node is uniquely identifiable (Pearl, 1988). For example, in Figure 1a, nodes A and D form $\mathbf{PC}_B$. $\mathbf{MB}_B$ contains its parent node A, its child D, and its spouse C. All other nodes E, F, and H are independent of B, given $\mathbf{MB}_B$, due to blocked paths.

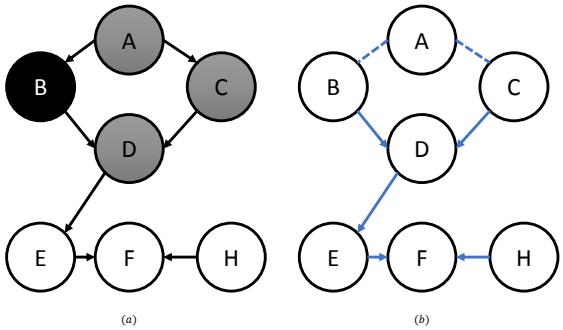

Figure 1. Sample Bayesian Network. a) Dark node B is the target node and the shaded nodes A, C, and D are the Markov Blanket of B. b) A learned DAG is shown, where the dashed edges can have different orientations while still being Markov Equivalent to the DAG in a).

Score-based structure learning algorithms rely on some score criteria s to learn a best-fitting DAG G for data D. Score $s(G, D)$ of a BN DAG structure G measures the goodness of fit of G on D. Let G be any BN structure and G' be the same structure as G but with an edge from a node T to a node X. Let $\mathbf{Pa}_X^G$ be the parent set of X in G. Score s is *locally consistent* if, as the size of the data D goes to infinity, the following two properties hold true: 1) if $X \not\perp\!\!\!\perp T|\mathbf{Pa}_X^G$, then $s(G, D) < s(G', D)$, and 2) if $X \perp\!\!\!\perp T|\mathbf{Pa}_X^G$, then $s(G, D) > s(G', D)$. In addition, s is *score equivalent* if Markov equivalent DAGs have the same score. s is *decomposable* if it is a sum of each node's individual score that depends on only this node and its parents. Commonly used Bayesian score criteria, such as BDeu, are decomposable, consistent, locally consistent (Chickering, 2002), and score equivalent (Heckerman et al., 1995). We assume the Markov condition, faithfulness condition, and the infinite data size hold in the theoretical analysis part of

the paper.

Lastly, one of the main concepts in the topology-based MB algorithms is the symmetry constraint, or the `AND-rule`.

Lemma 1. ***AND-Rule***. *For a node X to be adjacent to T in G, both of the following statements hold true: X must be in the PC set of T and T must be in the PC set of X, i.e., $X \in \mathbf{PC}_T^G$ and $T \in \mathbf{PC}_X^G$.*

The local-to-global BN structure learning algorithms generally use the AND-rule to enforce consistency between different learned local structures to obtain a global DAG (Margaritis & Thrun, 1999). Local structure learning algorithms also employ it to guarantee soundness (Niinimaki & Parviainen, 2012).

3. Local-to-Global BN Structure Learning

We will first introduce local BN structure learning and provide some new theoretical guarantees, then propose a novel procedure to expand the local graph to the global graph, including some consistency guarantee and the proposed GGSL algorithm.

3.1. Local Structure Learning

The local-to-global learning approach first uses local structure learning algorithms, either constraint-based (Tsamardinos et al., 2006) or score-based (Niinimaki & Parviainen, 2012), to discover the PC set or the Markov Blanket of the target. The arguably state-of-art algorithms to find the local structure of Bayesian network use a score-based framework (Gao & Ji, 2017), shown in Algorithm 1, `LocalLearn`.

In Algorithm 1, subroutine `BNStructLearn` learns an optimal DAG over a set of variables in the data, and can use any exact global BN structure learning algorithm. Subroutine `findPC` and `findSpouse` extract a variable T's PC set (by finding parent set $\mathbf{P}$ and children set $\mathbf{C}$) and spouse set given the adjacency matrix of a graph G. `LocalLearn` first sequentially learns the PC set by repeatedly using `BNStructLearn` on a set of nodes $\mathbf{Z}$ containing the target node T, its current PC set $\mathbf{PC}_T$, and one new query variable X. Then it uses a similar procedure to learn the spouse set and update the PC set. $\mathbf{PC}_T^G$ is guaranteed to contain all the true positive PC nodes of T.

Lemma 2. ***Preservation of True Positive PCs*** *(Niinimaki & Parviainen, 2012). Let G^0 be the faithful DAG of distribution $\mathcal{P}$ over $\mathbf{V}$, and G be the DAG learned by exact BN structure learning algorithms over the subset of variables $\mathbf{Z_L} \subseteq \mathbf{V}$ at the last iteration of Step 1 of Algorithm 1. Let $\mathbf{PC}_T^G$ be the learned PC set of the target T in G and $\mathbf{PC}_T^0$ be the PC set of T in G^0. Under the faithfulness and infinite data assumption, $\mathbf{PC}_T^0 \subseteq \mathbf{PC}_T^G$.*

Algorithm 1 `LocalLearn`

Input: dataset D, target node T
{step 1: find the PC set }
$\mathbf{PC}_T \leftarrow \emptyset$, $\mathbf{O} \leftarrow \mathbf{V} \setminus \{T\}$;
while $\mathbf{O}$ is nonempty **do**
  choose $X \in \mathbf{O}$, $\mathbf{O} \leftarrow \mathbf{O} \setminus \{X\}$;
  $\mathbf{Z} \leftarrow \{T, X\} \cup \mathbf{PC}_T$;
  $G \leftarrow$ `BNStructLearn` $(\mathbf{Z}, D_Z)$;
  $\mathbf{PC}_T, \mathbf{P}_T, \mathbf{C}_T \leftarrow$ `findPC`(G, T) ;
end while
{step 2: remove false PC nodes and find spouses}
$\mathbf{S}_T \leftarrow \emptyset$, $\mathbf{O} \leftarrow \mathbf{V} \setminus \mathbf{PC}_T$;
while $\mathbf{O}$ is nonempty **do**
  choose $X \in \mathbf{O}$, $\mathbf{O} \leftarrow \mathbf{O} \setminus \{X\}$;
  $\mathbf{Z} \leftarrow \{T, X\} \cup \mathbf{PC}_T \cup \mathbf{S}_T$;
  $G \leftarrow$ `BNStructLearn` $(\mathbf{Z}, D_Z)$;
  $\mathbf{PC}_T, \mathbf{P}_T, \mathbf{C}_T \leftarrow$ `findPC`(G, T) ;
  $\mathbf{S}_T \leftarrow$ `findSpouse`(G, T) ;
end while
Return: $\mathbf{MB} \leftarrow \mathbf{P}_T \cup \mathbf{C}_T \cup \mathbf{S}_T$;

However, $\mathbf{PC}_T^G$ may contain false positive PC nodes (Aliferis et al., 2010), as shown in Figure 1 of (Niinimaki & Parviainen, 2012). The iterative nature of Algorithm 1 can potentially violate the faithfulness assumption during the learning, due to absent variables. Previous analysis (Niinimaki & Parviainen, 2012; Gao & Ji, 2017) conjectured the soundness and completeness of `LocalLearn`. Here we provide a new theoretical proofs of these results in the `LocalLearn` .

Lemma 3. ***Preservation of Dependence Relationships between T and the Learned PC Set***. *Let G^0 be the global faithful DAG of $\mathcal{P}$ for the entire variable set $\mathbf{V}$, and G be the DAG learned by exact BN structure learning algorithms over a subset of variables of $\mathbf{V}$, $\mathbf{V}^G$, present at the last iteration of Step 1 of Algorithm 1. Then in G^0 every variable in the learned PC set $\mathbf{PC}_T^G$ is dependent of the target T, conditioned on any subset of the ground truth PC set: i.e., $X \not\perp\!\!\!\perp T|\mathbf{Z}, \forall X \in \mathbf{PC}_T^G, \forall \mathbf{Z} \subseteq \mathbf{PC}_T^0 \setminus \{X\}$.*

Proof. If $X \in \mathbf{PC}_T^0$, then the lemma holds automatically. Else if $X \notin \mathbf{PC}_T^0$, assuming $\exists X \in \mathbf{S}$ such that $X \perp\!\!\!\perp T|\mathbf{S} \setminus X$, where $\mathbf{S} = \mathbf{V}^G \setminus \{T\}$. Then, one of the following two cases must hold in G: $T \to X$ or $X \to T$. If $T \to X$, since each node in $\mathbf{Children}_X \cup \mathbf{Spouses}_X$ is either 1) not saved during the iterative procedure, or 2) saved as a node of $\mathbf{PC}_T^G$, in which case it forms a fully connected subgraph with X and T and can be changed to a $\mathbf{Parent}_X$. Then, $P(X|\mathbf{S} \setminus X) = P(X|\mathbf{Pa}_X^G)$ by MB definition. Since $P(X|\mathbf{S} \setminus X, T) = P(X|\mathbf{S} \setminus X)$ by assumption $X \perp\!\!\!\perp T|\mathbf{S} \setminus X$, then $T \notin \mathbf{Pa}_X$ since $\{\mathbf{S} \setminus X\}$ must contain $\mathbf{Pa}_X^G$. Then by local consistency removing the edge $T \to X$ will increase the score, which contradicts the as-

sumptions. If $X \to T$, since $P(T|X, \mathbf{S}\backslash X) = P(T|\mathbf{S}\backslash X)$ by assumption, then using a similar argument, $X \notin \mathbf{Pa}_T^G$ and removing the edge $X \to T$ will increase the score, which contradicts the assumption. Hence, $X \not\perp\!\!\!\perp T|\mathbf{S} \setminus X$. Since $\mathbf{Z} \subseteq \mathbf{PC}_T^0 \setminus \{X\} \subseteq \mathbf{S} \setminus \{X\}$, the lemma holds. □

Lemma 3 shows that the false PC nodes consist of only descendents of T, which is the same as Lemma 3 of (Gao & Ji, 2017) but without conjectured results:

Lemma 4. ***PC False Positive Identity***. *Let $\mathbf{PC}_T^G$ be the learned PC set of the target T in the learned graph G from Step 1 of Algorithm 1, and $\mathbf{PC}_T^0$ be the ground truth PC set in G^0. The false positives $\boldsymbol{F}$ in $\mathbf{PC}_T^G$ consist of only descendants of T in G^0, denoted as $\mathbf{Des}_T^0$, i.e., $\boldsymbol{F} \subseteq \mathbf{Des}_T^0$, $\boldsymbol{F} = \mathbf{PC}_T^G \setminus \mathbf{PC}_T^0$.*

Proof. By Lemma 2, $\mathbf{PC}_T^G$ consists of the entire $\mathbf{PC}_T^0$ and some false positives $\mathbf{F}$. We show $\mathbf{F} \subseteq \mathbf{Des}_T^0$. According to Lemma 3, a node $X \in \mathbf{F}$ is conditionally dependent of T given any $\mathbf{Z} \subseteq \mathbf{PC}_T^0 \setminus \{X\}$ in G^0. In the last iteration of Step 1, $\mathbf{PC}_T^G$ must contain the true positive parents of T $\mathbf{Pa}_T^0$, as $\mathbf{Pa}_T^0 \subseteq \mathbf{PC}_T^0 \subseteq \mathbf{PC}_T^G$ by Lemma 2. Therefore, $X \not\perp\!\!\!\perp T|\mathbf{Pa}_T^0$. However, by the Markov condition, all the non-descendant nodes X are independent of T given $\mathbf{Pa}_T^0$ in G^0. Hence non-descendants X cannot be in $\mathbf{PC}_T^G$ by the last iteration. Thus, $\mathbf{F} \subseteq \mathbf{Des}_T^0$. □

For the sake of complete discussion, we include the following property, showing the existence of unblocked paths:

Lemma 5. ***Coexistence Between Descendants and Spouses in Score-Based PC Search***. *In the learned G from Step 1 of Algorithm 1, the only false positives $\boldsymbol{F}$ in $\mathbf{PC}_T^G$ belong to the descendants of T, $\mathbf{Des_T^0}$, due to an unblocked path between T and its descendants via a V-structure $T \to$ Child $\leftarrow$ Spouse in G^0.*

Proof. Lemma 4 shows the first part of the lemma is true, and we just need to show the second part holds. Assuming false positive PC nodes $\mathbf{F}$ exist, let $X \in \mathbf{F} \subseteq \mathbf{Des}_T^0$, then Lemma 4 shows that $X \not\perp\!\!\!\perp T|\mathbf{Z}, \forall \mathbf{Z} \subseteq \mathbf{PC}_T^0$. For $\mathbf{F}$ to exist, $X \not\perp\!\!\!\perp T|\mathbf{PC}_T^0$ must be true. Since $\mathbf{PC}_T^0$ must be present in all paths from T to X in G^0, in the last iteration of the score-based PC search the dependence between T and X occurs only if $\mathbf{PC}_T^0$ unblocks some paths from T to X. This can only happen when there is a collider node in $\mathbf{PC}_T^0$. Hence, the only way X can exist in $\mathbf{PC}_T^G$ is through an unblocked path that contains a V-structure $T \to$ child $\leftarrow$ spouse in G^0. □

We show the consistency results of `LocalLearn`:

Theorem 1. *Under the infinite data and faithfulness assumption, `LocalLearn` finds all and only the Markov Blanket nodes of the target node.*

Proof. Step 1 of Algorithm 1 returns all of the true positive parents, children, and some descendants, if they exist, by Lemma 4 and Lemma 5. The tasks left are to add the true positive spouses and remove false positive PC nodes, i.e. the non-child descendants, from $\mathbf{PC}_T$.

First, we show `LocalLearn` will find all of the true positive spouses. In Step 2, `LocalLearn` learns a structure with the target, the current PC set, the current spouse set, and one query variable $X \in \mathbf{O} = \mathbf{V} \setminus \mathbf{PC}_T$. At each step, $\mathbf{PC}_T$ is a set that has been found to be dependent or conditionally dependent of the target. Since $\mathbf{PC}_T$ includes and will always include the true positive PC set by Lemma 2, true positive spouses are conditionally dependent of T given $\mathbf{PC}_T$. Following a similar logic as Lemma 2, $\mathbf{S}_T^0$ must directly connect to the true positive children of T. Because Step 2 of Algorithm 1 queries every variable in $\mathbf{V}$ and $\mathbf{S}_T^0$ must be dependent of nonadjacent T given $\mathbf{PC}_T^0$, to capture the correct independence relationships with the locally consistent score, $\mathbf{S}_T^0$ must be included in $\mathbf{S}_T$.

Secondly, we show false positive PCs and spouses will be removed. Since all the true positive PC nodes and spouses are present in the last iteration, the false positive PC nodes (the non-MB descendants by Lemma 4), if exist, should be adjacent to the true positive PC nodes and spouses in G. However, the DAG obtained from G by removing the edge from false positives PC to T would score higher by capturing the same independence relationships (i.e., the descendants are dependent of children and spouse nodes of T and independent of T given the true positive MB set) and the dependence relationships between the true positive MB set and T, but with fewer edges. Therefore, all false positive PC nodes will be removed from $\mathbf{PC}_T$. Similarly, since all the true positive PC nodes and spouses are present in the last iteration, if there exist false positive spouses $\mathbf{F} \subseteq \mathbf{S}_T^G \setminus \mathbf{S}_T^0$, $\mathbf{F}$ would be parents to some true positive children nodes $\mathbf{C}_T^0$ in G and $\mathbf{F} \not\perp\!\!\!\perp T|\mathbf{C}$. If so, $\mathbf{F}$ should be adjacent to $\mathbf{S}_T^0$ as well in G as every path between $\mathbf{C}_0$ and $\mathbf{F}$ must go through $\mathbf{S}_T^0$. However, the DAG obtained from G by removing the edge from $\mathbf{C}$ to $\mathbf{F}$ would score higher than G by capturing the same independence relationships but with less edges. Therefore, there will not be any false positive spouse nodes.

Thus, $\mathbf{PC}_T$ and $\mathbf{S}$ will contain all and only the true PC and spouses. Their union contains all and only the MB nodes. Therefore, `LocalLearn` is sound and complete. □

Note that the learned parent set $\mathbf{P}$ and children set $\mathbf{C}$ from `LocalLearn` themselves may not be sound or complete, due to Markov equivalence, even though their joint set is sound and complete.

3.2. Local-to-Global Learning

Using the local structures, one alternative approach to global structure learning is to learn the local structures of all nodes and then combine them to construct the global graph structure from the existing local graphs. Many algorithms (Margaritis & Thrun, 1999; Niinimaki & Parviainen, 2012) first use the AND-rule to resolve the potential conflicts among different local structures between adjacent variables, and then use the Meek rules (Meek, 1995) to obtain the final DAG. The AND-rule seems arbitrary in the learning results and can introduce errors if one of the neighbors learns a wrong local graph, hence affecting the accuracy of the final global graph. One naive way to solve such conflicts is to re-run the subroutine `BNStructLearn` on the variable set containing both neighbor sets to redetermine the existence of the edges. However, the procedure is inefficient as it repeats learning for local structures of every edge after repeating for every node, relearning the same parts of the graph multiple times.

We propose to anchor the learning at one target variable T, and then grow the graph by gradually expanding it. It is made possible as the `LocalLearn` does not require neighbors' local structure to learn T's neighborhood correctly, unlike previous algorithms. We only use the necessary variables at each iteration to expand the graph while keeping other parts of the learned graph fixed. The proposed graph growing structure learning (GGSL) algorithm is shown in Algorithm 2. Starting from an empty graph, GGSL iteratively updates the global graph by using `LocalLearn` to learn the local structure of one target variable T at a time. Each local learning uses the set consisting of the current local variables of the target variable in the learned graph G and variables that are not in the local structures of any variable. Then GGSL updates G with learned results using `updateGraph`. It runs until all but one variable is not learned and has four main steps:

Step 1: GGSL chooses one target variable T at each iteration, first chosen randomly and then based on query set $\mathbf{Q}$, which contains adjacent nodes of the already queried variables in the graph G. Q can be maintained as a regular queue (first in, first out). Queried variable set $\mathbf{A}$ keeps all the learned Ts and prevents repeated learning.

Step 2: GGSL uses `LocalLearn` shown in Algorithm 1 to find the local structure of the target variable over the query set $\mathbf{Z}$. This step resolves the potential edge conflicts through efficient learning, avoiding the simple AND-rule used by other local-to-global structure learning algorithms The main difference between each run of `LocalLearn` is that the previous T's will not be considered, unless they are in the local structure of the current target variable. Hence the max possible variable set size decreases over iterations, improving the memory efficiency.

Algorithm 2 Graph Growing Structure Learning

Input: data D, size m, variable set $\mathbf{V}$
$\mathbf{Q} \leftarrow \emptyset$; $\mathbf{A} \leftarrow \emptyset$;
$G \leftarrow zeros(|V|, |V|)$
repeat
  if $Q \neq \emptyset$ & $Q[0] \notin \mathbf{A}$ **then**
    $T \leftarrow Q.pop(0)$
  else
    $T \leftarrow$ the next unqueried variable in $\mathbf{V}$
  end if
  $G \leftarrow$ `GGSL`$(D, T, G, \mathbf{A}, \mathbf{V})$
  add adjacent nodes of T in G to Q
  $\mathbf{A} \leftarrow \mathbf{A} \cup T$
until $|S| = |V| - 1$
$G \leftarrow$ `PDAG-to-DAG`(G)
Return: G

Algorithm 3 `GGSL` Subroutine

Input: data D, target variable T, current DAG G, variable set $\mathbf{A}$, variable set V
$\mathbf{PC}, \mathbf{P}, \mathbf{C} \leftarrow$ `findPC`(G, T);
$\mathbf{Sp} \leftarrow$ `findSpouse`(G, T)
$\mathbf{Z} \leftarrow T \cup \mathbf{PC} \cup \mathbf{Sp} \cup \mathbf{V} \setminus \{T, \mathbf{PC}, \mathbf{Sp}, \mathbf{A}\}$
$D_Z \leftarrow D$ of the set $\mathbf{Z}$
$\mathbf{MB}, \mathbf{P}, \mathbf{C}, \mathbf{Sp} \leftarrow$ `LocalLearn`$(D_{\mathbf{Z}}, T)$
$G \leftarrow$ `updateGraph`$(G, T, \mathbf{P}, \mathbf{C}, \mathbf{Sp})$
Return: G

Step 3: Subroutine `updateGraph`, shown in Algorithm 3, performs the following check to enforce graph consistency: First, it checks if any directed parent in the existing G is learned as a child from the local learned children set $\mathbf{C}$. If so, it corrects the children into parents. Secondly, it checks if any directed child in the existing G is wrongly learned as a parent from $\mathbf{P}$. If so, it corrects the parents into children. Lastly, the algorithm checks if any $\mathbf{P}$ and $\mathbf{C}$ nodes can have a different edge direction without introducing new or destroying existing V-structures; if so, all these $\mathbf{P}$ and $\mathbf{C}$ are marked as undirected. Otherwise, they are marked as directed edges. There checks are needed to ensure the already-oriented edges remain unchanged, as the earlier runs of local structure learning uses more variables and hence are more likely to be correct. Lastly, to orient the newly learned spouse set $\mathbf{Sp}$ is straightforward, due to the definitive nature of V-structures.

Step 4: After repeating the first three steps for all but one variable, the last step of GGSL is to obtain a DAG given all the directed and undirected edges in the graph. Applying the conventional rules (Meek, 1995) to convert a partial directed acyclic graph (PDAG) to completely directed acyclic graph is sufficient, with an option to convert to DAG if desired.

Algorithm 4 `updateGraph`

Input: current DAG G, target T, learned parent $\mathbf{P}$, child set $\mathbf{C}$, spouse set $\mathbf{Sp}$
{Step 1. update the PC set}
for all C in $\mathbf{C}$ **do**
  if C conflicts with directed edge $C \to T$ in G **then**
    add C into $\mathbf{P}$ and remove C from $\mathbf{C}$
  end if
end for
for all P in $\mathbf{P}$ **do**
  if P conflicts with directed edge $T \to P$ in G **then**
    add P into $\mathbf{C}$ and remove P from $\mathbf{P}$
  end if
end for
if $|\mathbf{P}| \geq 2$ **then**
  orient $\mathbf{P}$ and $\mathbf{C}$ accordingly as a directed edge in G
else if directed edge T-A, $\forall A \in \{\mathbf{P} \cup \mathbf{C}\}$ cannot be reversed without destroying existing or introducing new V-structures **then**
  orient T-A accordingly as a directed edge in G
else
  orient T-A accordingly as a undirected edge in G
end if
{Step 2. update the spouse set}
Orient $\mathbf{Sp}$ and T to their children accordingly as directed edges in G
Return: G

Lemma 6. ***Requirement of Post-processing*** . *Subroutine* `PDAG-to-DAG` *is required in a local-to-global learning system to correctly orient DAG G to its Markov equivalent class in Algorithm 2.*

A simple example, shown in Figure 2 would justify Lemma 6. If the target T is chosen with the following order: F, E, D, C, and A, then the direction of edges $C - D$, $D - E$, and $E - F$ can be set in both directions and hence are labeled as undirected edges by Algorithm 2. When node C becomes the target, then edge $C - D$ is known. Other edges $D - E$ and $E - F$ have to be checked and corrected. Without subroutine `PDAG-to-DAG` to propagate the changes back, the learned DAG is not guaranteed to be the correct completely partially directed DAG (CPDAG).

We show Algorithm 2 is sound and complete:

Theorem 2. ***Soundness and Completeness***. *Under the infinite data and faithfulness assumption, Algorithm 2 GGSL learns and directs all and only the correct edges in the underlying DAG G^0, up to the Markov equivalent class of G^0.*

Proof. For the first target variable T_1, GGSL finds the correct local structure of one target variable by Theorem 1. The updated DAG G is sound and complete for T_1. Starting from the second iteration $i \geq 2$, since the learned graph G is shown to be correct, if any variable in the target variable T_i's MB set has been queried or saved, they must exist in the $\mathbf{PC}$ and $\mathbf{Sp}$ variable, hence in $\mathbf{Z}$ of Algorithm 3. With the rest of variables complementing the $\mathbf{PC}$ and $\mathbf{Sp}$ variables, all the true positive PC set and spouse set of T_i must be included in the set $\mathbf{Z}$ of Algorithm 3. By Theorem 1 again, the local structure learned must be sound and complete. Hence, at each iteration of Algorithm 2, the updated DAG G must be sound and complete for all T_is as well. Since the `PDAG-to-DAG` subroutine is also proven to orient the correct DAG from PDAG (Meek, 1995), the result DAG G at the end of Algorithm 2 must be in the Markov equivalent class of G^0. □

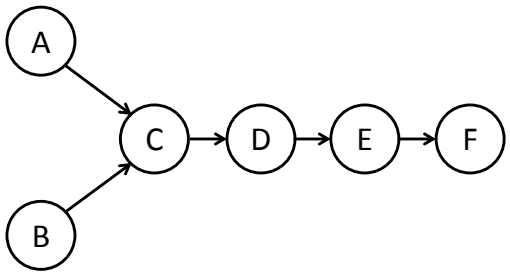

Figure 2. An example BN for Lemma 6. If the query order is $F \Rightarrow E \Rightarrow D \Rightarrow C \Rightarrow A$, the edge $E - F$ would be not guaranteed to orient correctly until C is queried and hence the post processing is needed.

3.3. Case Study

Applying Algorithm 2 to Figure 1a would result in the following procedure, assuming the query order for target T is $A \Rightarrow B \Rightarrow C \Rightarrow D \Rightarrow E \Rightarrow F$. When $T_1 = A$, `LocalLearn` finds B and C in the local structure of A with the query set of all variables, with undirected edges between them , and update G. With $T_2 = B$, the query set of variables contains its local structure A from G and the rest of variables. `LocalLearn` finds D and C as B's local structure, and updates G. With $T_3 = C$, `LocalLearn` finds the same A and D and does not make new update of G. With $T_4 = D$, `LocalLearn` finds E and update G with directed edge $D - E$ in G. Similarly, `LocalLearn` finds the directed edge $E - F$ and $H - F$ due to the definitive V-structure and update G accordingly, shown in Figure 1b, when $T_i = E$ and F. `PDAG-to-DAG` takes G and produces a possible DAG in Figure 1a.

3.4. Implementation Optimization

While Algorithm 2 is theoretically sound and complete, in practice further optimizations in the algorithmic procedure can be implemented. First, inside the iterative `LocalLearn` procedure, variables in the existing $\mathbf{PC}$ and $\mathbf{Sp}$ set of T_i should be queried first. These true positive MB set variables could potentially reduce the queried set size to `BNStructLearn` at each iteration by removing non-MB set of variables early in the process. Using this procedure is similar to the idea behind an improved version of `LocalLearn` (Gao & Ji, 2017), which is shown to improve accuracy and reduce computational time. Secondly, using the same concept, one can also keep track of which variables are removed from T_i's local structure after adding each queried variable X for target T_i's local learning procedure, hence forming a separation set, or sepset, of X from T. Querying variables from the sepset first when X becomes the target could also reduce the potential query set variable size to `LocalLearn`, as sepset variables are more likely to be adjacent variables of X.

3.5. Complexity and Performance Discussion

The exact global learning complexity varies depending on the algorithm, but is exponential in the worst case. For example, Dynamic programming approaches (Silander & Myllymaki, 2006) cost $O(N^2 2^N)$, where N is the total number of variables present. The most expensive step of local-to-global approach is each iteration of local learning using `LocalLearn`, which costs $O(N^3 2^N)$ using the same dynamic programming approach. Repeating for all the variable present, the local-to-global approach takes $O(N^4 2^N)$ in the worst case (Niinimaki & Parviainen, 2012; Gao & Ji, 2017). As one can see, in the case where the local structure of one target variable includes all other variable (such as in the Naive Bayesian model), the local-to-global approach would match the complexity of the global approach. However, by the iterative nature of the learning and the fact that the number of query variables decreases at later stages of learning, the expected running time is much lower for the local-to-global learning approach. If we assume a uniform distribution on the local neighbor size of each node in a network of N nodes, then the expected time complexity of the proposed GGSL approach is $O(\sum_{i=1}^{N} i^3 2^i) = O(N^3 2^N)$. If we assume l to be the maximum size of local neighbors, then the average complexity would be $O(l^3 2^l)$, which can lead to a big performance gain $O(2^{N-l})$.

Our algorithm is a score-based algorithm, as it can use any one of existing score-based optimal learner as the `BNStructLearn` subroutine. Theoretical optimality of the proposed algorithm holds only under standard assumptions. In real datasets, when the faithfulness assumption is violated or estimated probabilistic distributions are estimated incorrectly due to insufficient data, the performances of all the algorithms (global and local) are not guaranteed. During the learning procedure of the GGSL algorithm, even with a smaller query set of variables, the information about edge existence and orientation with sufficient data does not decrease compared to the global learning methods, if the local variables around each edge are all present. Hence, reducing the query set size would not affect performance with sufficient data, although the information loss does happen in practice. On the other hand, the estimation performance on the number of data samples is known to be sensitive to the number of parameters. The standard error of estimation is $\sigma / \sqrt[2]{N}$, where σ is the standard deviation. Due to the smaller set of variable present, the computation of Bayesian scores could be more accurate in practice when the sample size is limited. The trade-off between the information loss and estimation error varies among different datasets.

4. Experiments

We compare the proposed algorithms with both global and local-to-global learning methods. Specifically, we compare our results with global methods, Dynamic Programming (DP) structure learning (Silander & Myllymaki, 2006), Constrained Structure Learning (CSL) (de Campos et al., 2009), GOBNILP (Cussens et al., 2016), and local methods Score-based Local Learning (SLL+C) (Niinimaki & Parviainen, 2012) with three different `BNStructLearn` as above (DP, CSL, and GOBNILP), denoted as SLL+C-DP, SLL+C-CSL, and SLL+C-GOBNILP. We use the existing implementation of DP (in MATLAB), CSL (in C), and GOBNILP (in C), and implement our algorithms in MATLAB. We test the algorithms on benchmark BN datasets from the BN repository[1], using the datasets provided from existing works(Tsamardinos et al., 2006). We run the algorithms with 1000 samples of each dataset 10 times, and compare *BDeu* scores of each algorithm, along with the standard deviation, shown in Table 1. We also compare the algorithms on a synthetic 7-node network for DP as `BNStructLearn` so algorithms can return results within the time limit. We report the running time (the entire algorithmic time, including data access, score computation and structure search) of each algorithm[2], along with the standard deviation, shown in Table 2, with the maximum running time of 24 hours. The experiments are conducted on a machine with Intel i5-3320M 2.6GHz with 8 GB memory. Due to memory limitation, the DP method can fail to finish. CSL and GOBNILP have parameters that can control

[1] http://www.bnlearn.com/bnrepository/

[2] The time results are different from results on the GOBNILP website, which represent the times of finding the optimal structure given already computed scores.

Table 1. Learning Scores for Different BN Structure Learning Algorithms on Different Datasets.

DATA SET	VARIABLE SIZE	DP	SLL+C-DP	GGSL-DP
7BN	7	-13854.6± 133	-14224.6± 143	**-13781.2 ± 133**
ALARM	37	OOM	-14774.1± 210	**-11557.8± 379**
CHILDREN	20	OOM	-13548.0±172	**-12690.0± 106**
HAILFINDER	56	OOM	-62281.5±213	**-54551.6 ± 404**
CHILDREN3	60	OOM	-38713.2±255	**-37271.2± 315**
DATA SET	VARIABLE SIZE	CSL	SLL+C-CSL	GGSL-CSL
ALARM	37	**-10989.8** ± 196	-11437.0.1±268	-11033.4± 382
CHILDREN	20	-12690.0 ± 104	-12811.4±153	**-12600.0± 106**
HAILFINDER	56	**-54375.6** ±111	-58138.1±523	-57794.1± 623
CHILDREN3	60	-37407.5± 228	-38634.8± 249	**-37258.3± 340**
DATA SET	VARIABLE SIZE	GOBNILP	SLL+C-GOB	GGSL-GOB
ALARM	37	DNF	**-10337.1± 410**	-10575.0±269
CHILDREN	20	-12690.0 ± 104	-12811.4±153	**-12600.0± 106**
HAILFINDER	56	DNF	-54192.5± 781	**-53411.7± 844**
CHILDREN3	60	DNF	-38303.0±402	**-36950.1± 382**

OOM: OUT OF MEMORY. DNF: DID NOT FINISH

memory usages.

Table 2. Time Taken, in seconds, for Different BN Structure Learning Algorithms on Different Datasets.

Data set	DP	SLL+C-DP	GGSL-DP
7BN	0±0	0±0	0± 0
Alarm	OOM	2975±192	126± 36
Children	OOM	721±21	29± 3
Children3	OOM	39581±801	1119±369
Hailfinder	OOM	5083±279	885± 249
Data set	CSL	SLL+C-CSL	GGSL-CSL
Alarm	22220±614	512 ± 120	577± 25
Children	311 ±135	131±5	140± 15
Children3	52077±3622	1225±112	1374±395
Hailfinder	22640±42	1017±220	950±141
Data set	GOBNILP	SLL+C-GOB	GGSL-GOB
Alarm	$\geq 24hr$	369± 34	499±77
Children	73±2	63±2	71± 1
Children3	$\geq 24hr$	443± 132	569± 213
Hailfinder	$\geq 24hr$	394±62 41	459± 41

OOM: Out of Memory.

As one can see from Table 1, GGSL improves the learning scores by a significant margin over the SLL+C algorithm, with different `BNStructLearn` in all four datasets tested, except one case in ALARM with GOBNILP. It can even compete with global structure learning approaches in some cases. GGSL outperforms the global learning methods in CHILDREN datasets with CSL and GOBNILP. Efficiency-wise, from Table 2, using DP, GGSL is more efficient than SLL+C and can achieve one than one order of speedup. Using CSL and GOBNILP, GGSL has more than one order of magnitude speed-ups in 3 out of 4 datasets when compared with global learning method. However, GGSL's running time is generally slower to SLL+C algorithm using CSL and GOBNILP. It is faster than SLL+C on HAILFINDER with CSL, and is slightly slower in the other testing cases. We speculate that the difference in speed gains across different algorithms is mainly the code base of `BNStructLearn`, where the extra checking Step 3 in GGSL can take proportionally longer time if `BNStructLearn` is implemented in C.

5. Discussion and Conclusion

We have proposed a novel graph expanding learning algorithm to learn BN structure. We strengthen the existing local structure learning analysis, justifying its soundness when the traditional faithfulness condition fails with absent variables, and propose a new local-to-global approach to combine the local structures efficiently. Experiments have shown that the proposed GGSL improves the accuracy over existing local-to-global algorithms and improves efficiency over existing global algorithms, both by a significant margin. In addition, GGSL can work with any exact score-based BN learning algorithm and achieve consistent performance gain. The iterative nature of GGSL can have many applications, such as online BN structure learning with streaming data. Future work could study how GGSL would work with constraint-based (van Beek & Hoffmann, 2015; Gao & Ji, 2016b) and approximated BN structure learning algorithms as the `BNStructLearn` routines.

Acknowledgements

We thank Dennis Wei and anonymous reviewers for inspiration and helpful comments.

References

Acid, Silvia, de Campos, Luis M, and Castellano, Javier G. Learning bayesian network classifiers: Searching in a space of partially directed acyclic graphs. *Machine Learning*, 59(3):213–235, 2005.

Aliferis, Constantin F., Statnikov, Alexander, Tsamardinos, Ioannis, Mani, Subramani, and Koutsoukos, Xenofon D. Local causal and markov blanket induction for causal discovery and feature selection for classification part i: Algorithms and empirical evaluation. *Journal of Machine Learning Research*, pp. 171–234, Jan 2010.

Brenner, Eliot and Sontag, David. Sparsityboost: A new scoring function for learning bayesian network structure. *arXiv preprint arXiv:1309.6820*, 2013.

Chen, Eunice Yuh-Jie, Shen, Yujia, Choi, Arthur, and Darwiche, Adnan. Learning bayesian networks with ancestral constraints. In *Advances in Neural Information Processing Systems*, pp. 2325–2333, 2016.

Chickering, David Maxwell. Optimal structure identification with greedy search. *Journal of Machine Learning Research*, 2002.

Chickering, David Maxwell, Meek, Christopher, and Heckerman, David. Large-sample learning of bayesian networks is np-hard. *CoRR*, abs/1212.2468, 2012.

Cussens, James. Bayesian network learning with cutting planes. In *Proceedings of the Twenty-Seventh Conference Annual Conference on Uncertainty in Artificial Intelligence (UAI-11)*, pp. 153–160, Corvallis, Oregon, 2011. AUAI Press.

Cussens, James, Haws, David, and Studenỳ, Milan. Polyhedral aspects of score equivalence in bayesian network structure learning. *Mathematical Programming*, pp. 1–40, 2016.

de Campos, Cassio P., Zeng, Zhi, and Ji, Qiang. Structure learning of Bayesian networks using constraints. In *ICML '09: Proceedings of the 26th Annual International Conference on Machine Learning*, pp. 113–120, New York, NY, USA, 2009. ACM.

Fu, Shunkai and Desmarais, Michel C. Fast markov blanket discovery algorithm via local learning within single pass. In *Proceedings of the Canadian Society for computational studies of intelligence, 21st conference on Advances in artificial intelligence*, Canadian AI'08, Berlin, Heidelberg, 2008. Springer-Verlag.

Gao, Tian and Ji, Qiang. Local causal discovery of direct causes and effects. In *Advances in Neural Information Processing Systems*, pp. 2512–2520, 2015.

Gao, Tian and Ji, Qiang. Constrained local latent variable discovery. In *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence*, IJCAI'16, pp. 1490–1496. AAAI Press, 2016a.

Gao, Tian and Ji, Qiang. Efficient markov blanket discovery and its application. *IEEE Transactions on Cybernetics*, pp. 1–11, 2016b.

Gao, Tian and Ji, Qiang. Efficient score-based markov blanket discovery. *International Journal of Approximate Reasoning*, 80:277–293, 2017.

Gao, Tian, Wang, Ziheng, and Ji, Qiang. Structured feature selection. In *Proceedings of the IEEE International Conference on Computer Vision*, pp. 4256–4264, 2015.

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

Jaakkola, Tommi, Sontag, David, Globerson, Amir, and Meila, Marina. Learning bayesian network structure using lp relaxations. 2010.

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

Koller, Daphne and Sahami, Mehran. Toward optimal feature selection. In *ICML 1996*, pp. 284–292. Morgan Kaufmann, 1996.

Lazic, Nevena, Bishop, Christopher, and Winn, John. Structural expectation propagation (sep): Bayesian structure learning for networks with latent variables. In *Artificial Intelligence and Statistics*, pp. 379–387, 2013.

Margaritis, Dimitris and Thrun, Sebastian. Bayesian network induction via local neighborhoods. In *Advances in Neural Information Processing Systems 12*, pp. 505–511. MIT Press, 1999.

Meek, Christopher. Strong completeness and faithfulness in bayesian networks. In *Proceedings of the Eleventh conference on Uncertainty in artificial intelligence*, pp. 411–418. Morgan Kaufmann Publishers Inc., 1995.

Nie, Siqi, Mauá, Denis D, De Campos, Cassio P, and Ji, Qiang. Advances in learning bayesian networks of bounded treewidth. In *Advances in Neural Information Processing Systems*, pp. 2285–2293, 2014.

Niinimaki, Teppo and Parviainen, Pekka. Local structure disocvery in bayesian network. In *Proceedings of Uncertainy in Artifical Intelligence, Workshop on Causal Structure Learning*, pp. 634–643, 2012.

Ott, Sascha, Imoto, Seiya, and Miyano, Satoru. Finding optimal models for small gene networks. In *Pacific symposium on biocomputing*, volume 9, pp. 557–567, 2004.

Pearl, Judea. *Probabilistic reasoning in intelligent systems: networks of plausible inference*. Morgan Kaufmann Publishers, Inc., 2 edition, 1988.

Pellet, Jean-Philippe and Ellisseeff, Andre. Using markov blankets for causal structure learning. *Journal of Machine Learning Research*, 2008.

Scanagatta, Mauro, de Campos, Cassio P, Corani, Giorgio, and Zaffalon, Marco. Learning bayesian networks with thousands of variables. In *Advances in Neural Information Processing Systems*, pp. 1864–1872, 2015.

Silander, Tomi and Myllymaki, Petri. A simple approach for finding the globally optimal bayesian network structure. In *Proceedings of the Twenty-Second Annual Conference on Uncertainty in Artificial Intelligence (UAI)*, pp. 445–452, 2006.

Spirtes, Peter, Glymour, Clark N, and Scheines, Richard. *Computation, Causation, and Discovery*. AAAI Press, 1999.

Tsamardinos, Ioannis, Aliferis, Constantin, Statnikov, Alexander, and Statnikov, Er. Algorithms for large scale markov blanket discovery. In *In The 16th International FLAIRS Conference, St*, pp. 376–380. AAAI Press, 2003.

Tsamardinos, Ioannis, Brown, LauraE., and Aliferis, ConstantinF. The max-min hill-climbing bayesian network structure learning algorithm. *Machine Learning*, 65(1): 31–78, 2006.

van Beek, Peter and Hoffmann, Hella-Franziska. Machine learning of bayesian networks using constraint programming. In *International Conference on Principles and Practice of Constraint Programming*, pp. 429–445. Springer, 2015.

Yuan, Changhe and Malone, Brandon. Learning optimal bayesian networks: A shortest path perspective. 48:23–65, 2013.