Time and Sample Efficient Discovery of Markov Blankets and Direct Causal Relations

Ioannis Tsamardinos Constantin F. Aliferis Alexander Statnikov
Department of Biomedical Informatics, 2209 Garland Ave, Nashville, TN 37232 USA
{ioannis.tsamardinos,constantin.aliferis,alexander.statnikov}@vanderbilt.edu

ABSTRACT

Data Mining with Bayesian Network learning has two important characteristics: under broad conditions learned edges between variables correspond to causal influences, and second, for every variable T in the network a special subset (Markov Blanket) identifiable by the network is the minimal variable set required to predict T. However, all known algorithms learning a complete BN do not scale up beyond a few hundred variables. On the other hand, all known sound algorithms learning a local region of the network require an exponential number of training instances to the size of the learned region.

The contribution of this paper is two-fold. We introduce a novel local algorithm that returns all variables with direct edges to and from a target variable T as well as a local algorithm that returns the Markov Blanket of T. Both algorithms (i) are sound, (ii) can be run efficiently in datasets with thousands of variables, and (iii) significantly outperform in terms of approximating the true neighborhood previous state-of-the-art algorithms using only a fraction of the training size required by the existing methods. A fundamental difference between our approach and existing ones is that the required sample depends on the generating graph connectivity and not the size of the local region; this yields up to exponential savings in sample relative to previously known algorithms. The results presented here are promising not only for discovery of local causal structure, and variable selection for classification, but also for the induction of complete BNs.

Categories and Subject Descriptors

I.2.6 [**Artificial Intelligence**]: Learning

Keywords

Novel data mining algorithms, robust and scalable statistical methods, Bayesian networks

1. INTRODUCTION

SIGKDD '03, August 24-27, 2003, Washington, DC, USA.

Under certain broad conditions, as shown by the pioneering work in [8, 7, 1], induced Bayesian Networks (BN) from data have causal semantics: every edge from a variable X to a variable Y, $X \rightarrow Y$, means that X probabilistically and *directly* (i.e., with no other variable of the BN intervening) causes Y. Such causal knowledge is essential if one desires to manipulate the data process, e.g. to create a new drug based on causal knowledge of human gene interactions, or to restore a faulty device to normal operation, or increase sales to web site visitors, etc.

Knowledge of the BN representing the joint distribution is not only useful for causal discovery, but also for prediction, classification, and diagnosis. For every variable of interest T, the set of parents, children, and spouses (i.e., parents of common children) of T in a BN has special properties: given the values of these variables, the probability distribution of T is completely determined and knowledge of any other variable in the network becomes superfluous. This set of variables is called the Markov Blanket. Because of the above property, the Markov Blanket is inextricably connected to the variable selection problem, i.e., the problem of choosing a minimum set of predictors that optimally classify T. In particular, in [10] we prove that under certain broad conditions the Markov Blanket is the solution to the variable selection problem.

Several algorithms are currently available that can induce the BN that captures the data with accuracy given datasets consisting of a few hundred training instances [2]. However, these algorithms fail to scale to more than a few hundred variables.

A reasonable compromise to learning the full BN is to discover only the local structure (neighborhood) around a target variable of interest T (or a set of targets). We call the former *global learning* and the latter *local learning*. Ideally for causal discovery and manipulation of T the local structure of interest is the set of direct causes of T. However, it is hard to statistically distinguish between direct causes and direct effects of T. In this work, as a first step, we deal with the problem of identifying both the direct causes and direct effects of T and not distinguishing between the two.

The first contribution of this paper is the presentation of the first local-learning algorithm for identifying the parents and children of T in any BN that faithfully (see Section 2) represents the joint distribution of the data. Proof of uniqueness of this set is provided. When the conditions for causal discovery hold, the parents and children of T correspond to the direct causes and direct effects of T respec-

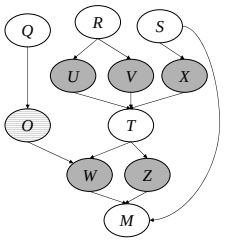

Figure 1: An example of a Bayesian Network. The $PC(T)$ are the variables in gray, while $MB(T)$ additionally includes the textured-filled variable O.

tively. The algorithm, called Max-Min Parents and Children or **MMPC** , is sound in the sample limit and scales up to datasets with thousands of variables. For finite sample, the quality of the output degrades gracefully. It is shown experimentally that the quality of the output matches or surpasses the one of the **PC** , a prototypical global BN learning algorithm that unlike **MMPC** does not scale up. This observation provides hope that local algorithms can be used with no degradation in quality to selectively learn parts of the BN, when learning the full BN is impractical.

The second contribution of the paper is a local-learning algorithm for identifying the Markov Blanket of T. Again, proof of uniqueness of this set is provided. The algorithm, called Max-Min Markov Blanket or **MMMB**, is based on **MMPC**, is sound in the sample limit, and scales up to datasets with thousands of variables. **MMMB** is compared with all previously known algorithms for inducing Markov Blankets, namely the Incremental Association Markov Blanket (**IAMB**) algorithm [11], the Grow-Shrink (**GS**) algorithm [5], and the Koller-Sahami algorithm (**KS**) [4]. **MMMB** trades-off computation time for required training sample size, while still being efficient enough to scale up to thousands of variables. **MMMB** yields up to exponential savings in sample relative to the sample required by most other local Markov Blanket induction methods. The fundamental difference between **MMMB** and the other approaches is that the sample requirements of the former depend on the connectivity and topology of the generating BN and not the size of the learned Markov Blanket.

2. BACKGROUND

Bayesian Networks (BN) [6] are mathematical objects that compactly represent a joint probability distribution J among a set of random variables $\mathbf{\Phi}$ (also called nodes) using a directed acyclic graph G annotated with conditional probability tables of the probability distribution of a node given any instantiation of its parents. The graph of an example BN is shown in Figure 1. The probability distribution J and the graph G of a BN are connected by the **Markov Condition** property: a node is conditionally independent of its non-descendants, given its parents.

Two nodes X and T are **conditionally independent** given the set of variables $\mathbf{Z}$, if and only if $P(T|X,\mathbf{Z}) = P(T|\mathbf{Z})$, denoted as $\mathbf{Ind}(X;T|\mathbf{Z})$, while conditional dependence is denoted as $\mathbf{Dep}(X;T|\mathbf{Z}) \equiv \neg\mathbf{Ind}(X;T|\mathbf{Z})$.

Definition 1. Faithfulness Condition. A BN N and a joint distribution J are **faithful to** one another, *iff* every conditional independence entailed by the graph of N and the Markov Condition is also present in J [8] (in the terminology of Pearl [7] N is a perfect map of J). A BN N is **faithful** if it is faithful to its corresponding distribution J.

d-separation is a criterion that allows computation of the entailed independencies in a BN from the Markov Condition [7]. d-separation is defined on the basis of blocked paths:

Definition 2. Collider node; Blocked path. A node W of a path p is a **collider** if p contains two incoming edges into W (e.g. W in Figure 1 is a collider in the path $O \rightarrow W \leftarrow T$). A path p from node X to node Y is **blocked** by a set of nodes $\mathbf{Z}$, if any of the following is true [6]:
1. There is a non-collider node on p that belongs in $\mathbf{Z}$.
2. No collider nodes of p and none of their descendants belong in $\mathbf{Z}$.

Definition 3. d-separation. Two nodes X and Y are **d-separated** by $\mathbf{Z}$ (denoted as $\mathbf{Dsep}(X;Y|\mathbf{Z})$) if and only if every path from X to Y is blocked by $\mathbf{Z}$.

Theorem 1. *If a BN N is faithful to a distribution J, then* $\mathbf{Dsep}(X;T|\mathbf{Z}) \Leftrightarrow \mathbf{Ind}(X;T|\mathbf{Z})$ *[7].*

Because of the theorem and the faithfulness assumption, the terms d-separation and conditional independence are used interchangeably in the rest of the paper. In BN learning conditional independencies between variables are established using statistical tests as described [8].

By performing such independence tests and considering the d-separation relations they entail, one can reconstruct the BN that captures the data generating process. This is the main idea behind constraint-based BN learning approaches [8, 3, 5]. The following theorem in [8] is foundational for the both the **PC** and the **MMPC** algorithms of Section 3:

Theorem 2. *If a BN N is faithful to a joint probability distribution J then:*
1. There is an edge between the pair of nodes X and Y in N iff X and Y are conditionally dependent given any other set of nodes.
2. If for the triplet of nodes X, Y, and Z in N, X is adjacent to Y, Y is adjacent to X, and Z is not adjacent to X, $X \rightarrow Y \leftarrow Z$ is a subgraph of N iff X and Z are dependent conditioned on every other set of nodes that contains Y.

The first part of the theorem allows one to infer the existence of edges; the second part to discover their direction.

Let $PC_N(T)$ denote the parents and children of T in network N. Then:

Theorem 3. *If BN C and BN N are both faithful to the same joint distribution J, then $PC_C(T) = PC_N(T)$ (proof in [12]).*

Because of the theorem, it makes sense to drop the index and denote the set simply as $PC(T)$ when referring to faithful BNs.

Another significant set of nodes is the Markov Blanket of T, denoted as $MB(T)$. $MB(T)$ can be defined either probabilistically [7] (as the minimal set conditioned on which every other node is independent of T) or graph theoretically [6] (as the set of parents, children, and spouses of T). *The two definitions coincide in distributions faithful to a BN.* Here we adopt the probabilistic definition and prove its graph theoretic properties:

Definition 4. Markov Blanket The Markov Blanket of T, $MB(T)$, is a minimal set conditioned on which all other nodes are independent of T, i.e. $\forall X \in \mathbf{\Phi} \setminus (MB(T) \cup \{T\}), \mathbf{Ind}(X; T|MB(T))$.

Theorem 4. *If BN N is faithful, then for every variable T, $MB(T)$ is unique and is the set of parents, children, and spouses of T (proof in [10]).*

Thus, the Markov Blanket has a precise graph correspondence in faithful BNs. As a corollary of Theorem 4 the following is true (proof in [12]):

Theorem 5. *If BN C and BN N are both faithful to the same joint distribution J, then the set of parents, children, and spouses of a variable T in C is the same as the set of parents, children, and spouses of T in N, and is the $MB(T)$.*

Having proven the uniqueness of $PC(T)$ and $MB(T)$ in faithful distributions we are now ready to define the problem that we address in the rest of the paper:

Definition 5. Local Structure Identification Problems. Given a probability distribution J faithful to some BN N and a node of interest T, identify the sets (i) $PC(T)$, and (ii) $MB(T)$.

3. MAX-MIN PARENTS AND CHILDREN

The Max Min Parents and Children (**MMPC**) algorithm discovers $PC(T)$ using a two-phase scheme (Figure 2). In phase I, the forward phase, variables enter sequentially a candidate $PC(T)$ set, from now on denoted as **CPC** (candidate parents and children), by use of a heuristic function. The heuristic is admissible in the sense that all members of PCT and possibly some non-members will enter **CPC**. In phase II, the backward phase, **MMPC** removes all false positives that entered in the first phase. In the end **CPC**=$PC(T)$.

MMPC first includes in **CPC** the variable with the highest univariate association with T. **MMPC** chooses to include next into **CPC** the variable that exhibits the maximum association with T conditioned on the subset of **CPC** that achieves the minimum association possible for this variable. Intuitively the heuristic is the following: select the variable that, despite our best efforts to make it independent of T (i.e., considering the minimum association conditioned on all possible subsets of **CPC**) has the highest such minimum association with T among all other candidate variables. Association of X with T given $\mathbf{Z}$ in the pseudo-code is denoted with $assoc(X; T|\mathbf{Z})$.

The heuristic is admissible since in a faithful BN a parent or child of T always has a non-zero association with (is dependent on) T given any subset of variables and so it will enter **CPC** eventually and will not be removed thereafter.

In the second phase, **MMPC** examines whether each variable of **CPC** can be d-separated from T by conditioning on all possible subsets of **CPC**. If for some conditioning set X is d-separated from T, then X is removed from **CPC**; otherwise it is retained. Conditioning on all subsets is feasible in practice because the first phase typically admits very few false positives.

The conditional test of independence in our implementation of **MMPC** are G^2 tests as described and used in [8]. Specifically, if the two-tailed p-value returned by the test is less than the standard 5% threshold the variables are assumed independent, otherwise they are considered dependent. As a measure of association, i.e., function *assoc* in the pseudocode, we used again the p-value returned by the G^2 test: the smaller the p-value, the higher the association. Any other reliable test of independence and measure of association can be used.

MMPC(Data D; Target T)
Phase I (forward)
1 **CPC** = $\emptyset$
2 Repeat
3 For every variable X find
4 $minassocset(X)$=subset **s** of **CPC** that minimizes $assoc(X; T|\mathbf{s})$
5 End For
6 F=variable of $\mathbf{\Phi} \setminus (\{T\} \cup \mathbf{CPC})$ that maximizes $assoc(F; T|minassocset(F))$
7 If $\mathbf{Dep}(F; T|minassocset(F))$
8 **CPC** = **CPC** $\cup F$
9 End If
10 Until **CPC** has not changed
11 **PhaseII** (backward)
12 For all $X \in$ **CPC**
13 If $\exists \mathbf{s} \subseteq \mathbf{CPC}, s.t. \mathbf{Ind}(X; T|\mathbf{s})$
12 **CPC** = **CPC** $\setminus \{X\}$ /* Remove X from **CPC** */
13 End If
14 End For
15 Return **CPC**

Figure 2: The Max-Min Parents and Children (MMPC) Algorithm.

The number of the training instances to reliable estimate the G^2 statistic is exponential to the size of the conditioning set $\mathbf{Z}$. Following the practice used in [8] in our experiments running **MMPC** we do not perform independence tests unless there are at least five training instances on the average per parameter to be estimated.

The implications of this restriction is that at Lines 4 and 13 in Figure 2 only a limited number of subsets of **CPC** actually participate to the calls to functions **Ind** and *assoc*. When sample is limited, this method may allow false positives to enter **CPC**. For example, if the available sample is enough to condition on only one variable in the BN of Figure 1, then nodes M and S will enter **CPC** and remain at the final output since we need to condition on at least two variables to d-separate them from T. Had such a restriction not been imposed, the tests of independence for large conditioning sets will invariably return a large p-value, indicating that there is not enough sample to reject the null hypothesis of independence.

In our experiments with BNs from real decision support systems, most variables not in $PC(T)$ can be d-separated conditioned on only a few (less than 4) variables.

The worst-case complexity of **MMPC** is prohibitive for all but the smallest problems. However, in practice (see timing results in Section 6), **MMPC** 's heuristic is powerful enough so that **CPC** is of the order of $|PC(T)|$. With this assumption, the order of the complexity becomes $O(|\Phi| PC(T)^k)$. For a full analysis of the time complexity and a set of optimizations that speed up the algorithm see [12].

MMMB(Data D; Target T)
Phase I:Get a superset of $MB(T)$
1 **PCofT** = MMPC(D, T)
2 **CMB** = **PCofT** $\bigcup_{C \in \mathbf{PCofT}}$ **MMPC**(D, T)

Phase II: Remove False Positives
3 For every potential spouse $X \in$ **CMB** $\setminus$ **PCofT**
4 Find **s** such that **Ind**$(X; T|\mathbf{s})$
5 For every potential child variable $Y \in$ **PCofT**
6 If **Dep**$(X; T|Y \cup \mathbf{s})$
7 Mark X for inclusion in **CMB**
8 EndIf
9 End For
10 Remove X from **CMB** unless it is marked
10 End For
11 Return **CMB**

Figure 3: The Max-Min Markov Blanket (MMMB) Algorithm.

4. MAX-MIN MARKOV BLANKET

We now turn our attention to the problem of discovering the $MB(T)$. The difference of $MB(T)$ and $PC(T)$ is that the former additionally includes the spouses of T. Thus, one could first discover $PC(T)$ by running **MMPC**, then attempt to identify the spouses of T. The spouses of T are the parents of the common children of T which suggests that they should belong to $\bigcup_{C \in PC(T)} PC(C)$ (i.e., the union of the parents and children of the parents and children set). However, this union set also includes the children of the children of T, the parents of the parents of T, and the children of the parents of T.

The **MMMB** algorithm, shown in Figure 3 first identifies the $PC(T)$, then recursively calls the algorithm a second time to identify $PC(T) \bigcup_{C \in PC(T)} PC(C)$. This latter candidate Markov Blanket (**CMB**) set is a superset of $MB(T)$, and a method for filtering out the false positives is required for the algorithm to be correct in the sample limit.

Spouses X of T that are not adjacent to T have the following property: conditioned on any subset that includes a common child (or children) X and T are dependent (Theorem 2 part 2). This property is not shared by the false positives in **CMB** and thus, it can be used to filter the latter out. One problem with checking the property directly is that we do not know which nodes in $PC(T)$ are actually children. The second is that it is inefficient to condition on all possible subsets.

Fortunately, **MMMB** overcomes both of these problems. First, it identifies a subset **s** that d-separates X from T (Line 4) (**MMPC** has already identified such a subset and could cache it for retrieval). Now notice, that if there is a variable $Y \in PC(T)$, such that **Dep**$(X; T|Y \cup \mathbf{s})$ then Y has to be a child of T and X has to be a spouse of T. This is from the definition of the d-separation. The reverse also holds, so if there is no node Y for which the condition holds, X cannot be a spouse of T and it can be filtered out.

Theorem 6. *Algorithms* **MMPC** *and* **MMMB** *will return $PC(T)$ and $MB(T)$ respectively given that (i) the data D are faithful to a BN N, (ii) the statistical tests of independence and measure of associations are reliable (proof in the [12]).*

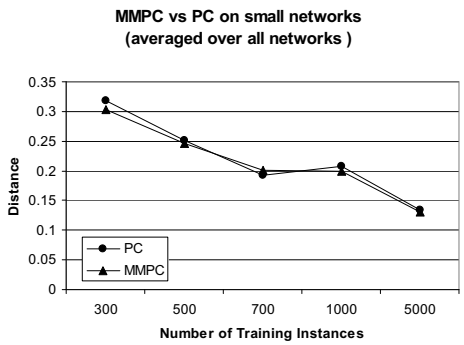

Figure 4: The comparison between MMPC and PC. Each point in the graph is the average distance over all small networks (except Pigs) and possible targets.

5. RELATED WORK

GS [5] and **IAMB** [10, 11] were the first local BN learning algorithms that could provably discover the $MB(T)$ in the sample limit. Unlike **MMPC**, instead of trying all subsets of the **CPC** or **CMB** in an attempt to d-separate all nodes not in the local neighborhood, they condition on the full **CMB**. Obviously, conditioning on the full **CMB** instead of all subsets of it significantly reduces the time complexity. However, the sample requirements of the algorithms also increase exponentially.

The **KS** algorithm [4] is one of the first that employed the concept of the Markov Blanket for feature selection. The algorithm accepts two parameters: (i) the number v of variables to retain and (ii) a parameter k which is the maximum number of variables the algorithm is allowed to condition on. For k=0 **KS** is equivalent to univariately ordering the variables and selecting the first v. The Koller-Sahami paper does not explicitly claim to identify the $MB(T)$; however, if one could guess the size of the $MB(T)$ and set the parameter v to this number then ideally (if their heuristics were perfect) the algorithm should output $MB(T)$. Viewed this way we treated the **KS** algorithm as an algorithm for approximating the $MB(T)$ using only v variables. Unlike the rest of the algorithms in this paper, **KS** comes with no theoretical guarantees of discovering the $MB(T)$ even in the sample limit.

PC learns the full network starting with a fully connected unoriented Bayesian Network graph and has three phases. In phase I the algorithm finds undirected edges by using the criterion that variable A has an edge to variable B *iff* for all subsets of variables there is no subset **S**, s.t. **Ind**$(A; B|\mathbf{S})$. In phases II and III the algorithm orients the edges by performing global constraint propagation.

6. EXPERIMENTAL RESULTS

We evaluate the new algorithms using data sampled from probability distributions of BNs used in real Decision Support Systems that capture a wide variety of real life applications (medicine, weather forecasting, financial modelling, device troubleshooting, and animal breeding). Since the true structure of each network is known there exist a rigorous gold standard for assessing the performance of each algorithm. The generating networks are taken from the Bayesian

Network Repository maintained at the Hebrew University of Jerusalem, at URL http://www.cs.huji.ac.il/labs/compbio/Repository. The repository also cites appropriate references. The networks are: **ALARM**, a network created by medical experts for monitoring patients in the intensive care unit; **Hailfinder**, a system that forecasts severe summer hail in northeastern Colorado; **Pigs**, a BN for predicting the pedigree of breeding pigs; **Insurance**, a network for evaluating car insurance risks; **Win95PTS**, an expert system for printer troubleshooting in Windows 95.

To further show that our algorithms scale up we also simulated large BNs having approximately 5000 variables each. The BNs were created by tiling several copies of the smaller real BNs in a way that retains their structural and probabilistic properties, hoping that the simulated network will exhibit the same characteristics as the real BN tiles (the details are in [9]). More specifically, we randomly generated **ALARM-5K** by tiling 135 copies of ALARM. Similarly, for **Pigs-5K** (11 copies of original Pigs) and **Hailfinder-5K** (89 copies of original Hailfinder). We did not create tiled versions of Insurance and Win95PTS because they are available in a format not suitable for our tiling simulator.

Subsequently we generated training instances of sizes 300, 500, 700, 1000, and 5000 drawn from the probability distributions of these BNs. The algorithms were then applied to identify the $PC(T)$ or the $MB(T)$ (depending on the algorithm) for a set of targets T. For the small BNs we ran the algorithms using as targets all variables (except for the **Pigs** dataset which contains 400 variables from which we randomly selected 50 as targets). For the large BNs we ran the algorithms for 10 randomly selected variables from each network choosing randomly among variables with Markov Blanket size of greater than four (so as the cases were not among the easiest ones).

For each single experiment (i.e the application of an algorithm on a particular training sample and target) we measure the *sensitivity* and *specificity* for the given task, i.e., in discovering the $PC(T)$ or $MB(T)$. When the task is to discover the $PC(T)$ the sensitivity is the ratio of the number of correctly identified variables in the $PC(T)$ over the size of the true $PC(T)$. The specificity is the ratio of the number of correctly identified variables as *not* belonging in the $PC(T)$ over the true number of variables not in $PC(T)$. Similarly, specificity and sensitivity is defined for identifying the $MB(T)$.

These statistics provide two related measures for comparing two algorithms. None is sufficient by itself. This is because an algorithm may achieve perfect sensitivity by including all variables in the output, or achieve perfect specificity by excluding all variables from the output. Thus, a combined measure is required. The measure we used for the evaluation is the proximity of the sensitivity and specificity of the algorithm to perfect sensitivity and specificity expressed as the Euclidean distance:

$$d = \sqrt{(1 - sensitivity)^2 + (1 - specificity)^2}$$

Notice that the *smaller* the distances, the closer the output of an algorithm is to the true local neighborhood. For example, when the sensitivity and specificity are equal to 85%, 90%, and 95%, d is respectively 0.21, 0.14, and 0.07.

All algorithms were implemented in Matlab 6.5. The experiments were run on Intel Xeon and Intel Pentium IV computers running Windows 2000 with a clock speed ranging from 1.5 to 2.4 GHz and RAM ranging from 1-2 GB. We reimplemented all previous algorithms since there are no publicly available versions of the original **KS** and **GS** algorithms. For the **PC** we used our version that matches exactly the output of the publicly available version for the given tasks and can also be integrated with our scripts. For all algorithms that are using conditional independence tests we used the same G^2 statistic as described in Section 2.

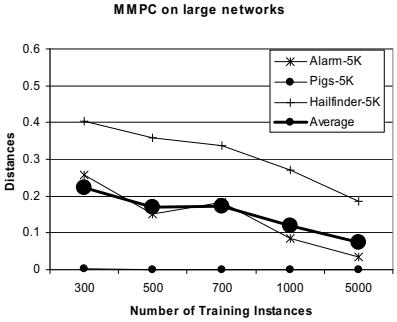

Figure 5: The performance of MMPC on large networks.

MMPC, to the best of our knowledge, is the first local-learning algorithm for discovering the $PC(T)$ and so we can only compare it with global learning BN algorithms such as the **PC** algorithm. The public version of **PC** has a limit of 100 variables and so we did not run it on the **Pigs** dataset.

MMPC returned the $PC(T)$ for each target in the four small BNs. **PC** returned the whole BN from which the $PC(T)$ for every target T was extracted. The distance from perfect specificity and sensitivity, averaged out on all targets and all four networks is shown in Figure 4 for the given samples sizes. The performance of both algorithms is very close. As expected, both algorithms perform better (smaller distance) with larger available sample.

Figure 5 shows the performance of **MMPC** for the large BNs. We were unable to run **PC** or any other global BN learning method on these sets due to their sizes. Each point in the graph is the average of the 10 randomly selected targets. The **Pigs-5K** dataset proved to be the easiest for discovery and **MMPC** has almost perfect performance on it. **Hailfinder-5K** was the most difficult for **MMPC** to recover (close inspection of the network showed the existence of distributions that are close to violating faithfulness), while the performance for **ALARM-5K** is similar to the average performance.

In the same figure, we see that the average distance of **MMPC** when the sample is above 1000 instances, is about 0.1 . This distance corresponds to a sensitivity and specificity of 93%. Similarly, a distance of 0.2, which is the average performance for sample sizes between 300 and 700 corresponds to sensitivity and specificity of 86%.

The maximum time for **MMPC** on any target or sample size for each dataset is 11, 35, 12, and 30 seconds for **ALARM**, **Hailfinder**, **Insurance**, and **Win95PTS** respectively. On the same datasets **PC** took respectively a maximum (depending on the sample size) of 40, 263, 882, and 57668 seconds to discover the full network. For the large networks **ALARM-5K**, **Hailfinder-5K**, and **Pigs-5K** **MMPC** ranged between 38-126, 31-2294, and 47-180

Dataset	Sample Size 300	500	700	1000	5000
ALARM	**3.4%**	**8.4%**	**7.6%**	**9.1%**	**4.5%**
Pigs	**24.1%**	**20.2%**	**20.4%**	**19.7%**	**6.0%**
Hailfinder	**1.5%**	-3.4%	**4.1%**	**4.4%**	**5.3%**
Insurance	-1.1%	**3.3%**	**12.8%**	**11.5%**	**6.5%**
Win95PTS	-2.8%	-5.4%	-11.6%	-5.2%	**1.2%**
ALARM-5K	**9.0%**	**7.2%**	**12.4%**	**17.6%**	**18.2%**
Pigs-5K	**20.5%**	**14.7%**	**14.8%**	**11.4%**	**1.6%**
Hailfinder-5K	-4.0%	**13.8%**	**15.8%**	**18.0%**	**16.5%**

Figure 6: Summary of results for discovery of $MB(T)$. Cells with positive values indicate cases where MMMB is better than all other algorithms (33/40 cases).

seconds respectively.

The algorithms used for comparison with **MMMB** are the basic and advanced version of the Incremental Association Markov Blanket algorithm **IAMB** and **IAMBnPC** [11], the Koller-Sahami **KS** [4] for all three values of the k parameter, 0, 1, and 2 used in the original paper, the Grow-Shrink **GS** [5] algorithm, and the **PC** algorithm (a total of seven baseline algorithms). **IAMB**, **IAMBnPC**, and **GS** explicitly try to identify the $MB(T)$ and so their output is directly comparable to the true local structure. **KS** requires an extra parameter v, which is the number of variables to retain to approximate the $MB(T)$. We set the value of v to the size of the output of **MMMB** so as the two algorithms are directly comparable. As mentioned, **PC** returns the full network from which the $MB(T)$ can be extracted. **PC** was not run on any dataset larger than 100 nodes and **KS** was not run on the large BNs with 5000 variables. This is because the average time complexity of **KS** is $O(|\Phi|^2)$, which is prohibitive for such large networks. Even though all local algorithms have a worst-case time complexity of equal or worse order than **KS**, their average time complexity is much better since it depends on the size of the identified neighborhood which is typically much smaller than the full set Φ of variables. The quality of the output of the **KS** on the small networks was on the average much less than **MMPC**'s. This indicates that the results would be the same if it had been possible to run **KS** on the larger networks too.

A summary of all results is shown in Figure 6. In the figure, for each cell let d_{other} be the minimum distance from perfect sensitivity and specificity achieved by all other algorithms, and d_{MMMB} the distance achieved by **MMMB** . The value in the cell contains the difference $d_{other} - d_{MMMB}$. For example, For sample size equal to 500 and for **Pigs** the distance of **MMMB** from perfect sensitivity and specificity is 20.2% smaller than the best other algorithm. Thus, positives values indicate that **MMMB** was better than all other algorithms for that task and vice versa.

MMMB is better than all other algorithms in 33 out of 40 cases. Most of the times **MMMB** did not outperform all other algorithms are in the **Win95PTS** network, which was the BN that also proved the most difficult for **MMPC**. A detailed analysis of the experiments is in [12].

As already mentioned, compared to the other local learning algorithms **MMMB** trades off computational effort for reduced required data sample. The maximum time observed in the experiments for **MMMB** over all targets and sample sizes is 21, 392, 49, 341, 43516, 2143, 123701, and 1176 seconds on **ALARM**, **Hailfinder**, **Insurance**, **Win95PTS**, **Pigs**, **ALARM-5K**, **Hailfinder-5K**, and **Pigs-5K** respectively.

7. CONCLUSIONS

We introduced a novel local algorithm (**MMPC**) that returns all variables with direct edges to and from a target variable T in faithful distributions. We also introduced a local algorithm (**MMMB**) that returns the Markov Blanket of T. We proved the algorithms' soundness, and compared them to state-of-the-art alternatives. The experiments showed the scalability of both **MMPC** and **MMMB** to datasets with thousands of variables and their superiority in approximating the local regions compared to a wide variety of baseline algorithms. **MMPC** and **MMMB** employ techniques that allow exponential savings in required training sample compared to the baseline algorithms. They can be used for local causal discovery, variable selection for classification, and for the induction of complete BNs. For a more thorough discussion of the algorithms see [12].

8. REFERENCES

[1] C. Glymour and G. F. Cooper, editors. *Computation, Causation, and Discovery*. AAAI Press/The MIT Press, 1999.

[2] D. Heckerman. A tutorial on learning bayesian networks. Technical Report MSR-TR-95-06, Microsoft Research, March 1995.

[3] C. Jie, R. Greiner, J. Kelly, D. A. Bell, and W. Liu. Learning bayesian networks from data: An information-theory based approach. *Artificial Intelligence*, 137:43–90, 2002.

[4] D. Koller and M. Sahami. Toward optimal feature selection. In *Thirteen International Conference in Machine Learning*, 1996.

[5] D. Margaritis and S. Thrun. Bayesian network induction via local neighborhoods. In *Advances in Neural Information Processing Systems 12 (NIPS)*, 1999.

[6] R. E. Neapolitan. *Probabilistic Reasoning in Expert Systems*. John Wiley and Sons, 1990.

[7] J. Pearl. *Probabilistic Reasoning in Intelligent Systems*. Morgan Kaufmann, San Mateo, CA, 1988.

[8] P. Spirtes, C. Glymour, and R. Scheines. *Causation, Prediction, and Search*. The MIT Press, second edition, 2000.

[9] A. Statnikov, I. Tsamardinos, and C. F. Aliferis. An algorithm for the generation of large Bayesian Networks. Technical Report DSL-03-01, Vanderbilt University, 2003.

[10] I. Tsamardinos and C. F. Aliferis. Towards principled feature selection: Relevancy, filters and wrappers. In *Ninth International Workshop on Artificial Intelligence and Statistics (AI&Stats 2003)*, 2003.

[11] I. Tsamardinos, C. F. Aliferis, and A. Statnikov. Algorithms for large scale markov blanket discovery. In *The 16th International FLAIRS Conference*, 2003.

[12] I. Tsamardinos, C. F. Aliferis, and A. Statnikov. Time and sample efficient discovery of Markov Blankets and direct causal relations. Technical Report DSL-03-02, Vanderbilt University, 2003.