Towards Scalable and Data Efficient Learning of Markov Boundaries

Jose M. Peña [a,*], Roland Nilsson [a], Johan Björkegren [b], Jesper Tegnér [a,b]

[a] *IFM, Linköping University, SE-58183 Linköping, Sweden*

[b] *CGB, Karolinska Institutet, SE-17177 Stockholm, Sweden*

Abstract

We propose algorithms for learning Markov boundaries from data without having to learn a Bayesian network first. We study their correctness, scalability and data efficiency. The last two properties are important because we aim to apply the algorithms to identify the minimal set of features that is needed for probabilistic classification in databases with thousands of features but few instances, e.g. gene expression databases. We evaluate the algorithms on synthetic and real databases, including one with 139351 features.

Key words: Bayesian networks, feature subset selection, classification

1 Introduction

Probabilistic classification is the process of mapping an assignment of values to some random variables $\mathbf{F}$, the features, into a probability distribution for a distinguished random variable C, the class. Feature subset selection (FSS) aims to identify the minimal subset of $\mathbf{F}$ that is needed for probabilistic classification. Solving the FSS problem is important for two main reasons. First, it provides insight into the domain at hand and, second, it reduces the search space if the probabilistic classifier is to be learnt from data.

In this paper, we are interested in solving the FSS problem as follows. Since a Markov boundary (MB) of C is defined as any minimal subset of $\mathbf{F}$ that renders the rest of $\mathbf{F}$ independent of C, then a MB of C is a solution to the FSS

* Corresponding author. E-mail address: jmp@ifm.liu.se

problem. If the probability distribution of $\{\mathbf{F}, C\}$ can be faithfully represented by a Bayesian network (BN) for $\{\mathbf{F}, C\}$, then the MB of C is unique and can easily be obtained because it consists of the union of the parents and children of C and the parents of the children of C [12]. In this paper, we are interested in solving the FSS problem for databases with thousands of features but few instances. Such databases are common in domains like bioinformatics and medicine, e.g. gene expression databases [16]. Unfortunately, having to learn a BN for $\{\mathbf{F}, C\}$ in order to learn a MB of C can be painfully time consuming for such high-dimensional databases [22]. This is particularly true for those algorithms for learning BNs from data that are (asymptotically) correct under the faithfulness assumption [22], which are the ones we are interested in. Fortunately, there exists an algorithm for learning a MB of C from data that scales to high-dimensional databases and that is correct under the faithfulness assumption, the incremental association Markov boundary algorithm (IAMB) [19]. IAMB is scalable because it does not learn a BN for $\{\mathbf{F}, C\}$. However, IAMB is data inefficient because it may require an unnecessarily large amount of learning data to identify a MB of C. This raises the first question addressed in this paper: Can we develop an algorithm for learning MBs from data that is scalable, data efficient, and correct under the faithfulness assumption ? The answer is yes. In Section 4, we present such an algorithm, the parents and children based Markov boundary algorithm (PCMB). This leads us to the second question addressed in this paper: Can we relax the faithfulness assumption and develop an algorithm that is correct, scalable and data efficient ? We prove that IAMB is still correct under the composition property assumption, which is weaker than the faithfulness assumption. The proof also applies to a stochastic variant of IAMB that we propose in order to overcome the data inefficiency of IAMB. We call it KIAMB. This algorithm has the following additional advantage over IAMB. If C has several MBs (something impossible under the faithfulness assumption but possible under the composition property assumption), then KIAMB does not only return a MB of C but any MB of C with non-zero probability. Therefore, KIAMB can discover different MBs of C when run repeatedly while IAMB cannot because it is deterministic. We report experiments showing that PCMB outperforms IAMB and that KIAMB outperforms both IAMB and PCMB considerably often. To show that these algorithms are scalable, part of the experiments are run on the Thrombin database which contains 139351 features [2]. Before going into the details of our contribution, we review some key concepts in the following section.

2 Preliminaries

The following definitions can be found in most books on Bayesian networks, e.g. [12,17,18]. Let $\mathbf{U}$ denote a set of discrete random variables. A Bayesian network (BN) for $\mathbf{U}$ is a pair (G, θ), where G is an acyclic directed graph (DAG) whose nodes correspond to the random variables in $\mathbf{U}$, and θ are parameters specifying a probability distribution for each node X given its parents in G, $p(X|Pa(X))$. A BN (G, θ) represents a probability distribution for $\mathbf{U}$, $p(\mathbf{U})$, through the factorization $p(\mathbf{U}) = \prod_{X \in \mathbf{U}} p(X|Pa(X))$. In addition to $Pa(X)$, two other abbreviations that we use are $PC(X)$ for the parents and children of X in G, and $ND(X)$ for the non-descendants of X in G. Hereinafter, all the probability distributions and DAGs are defined over $\mathbf{U}$, unless otherwise stated. We call the members of $\mathbf{U}$ interchangeably random variables and nodes.

Let $\mathbf{X} \perp\!\!\!\perp \mathbf{Y}|\mathbf{Z}$ denote that $\mathbf{X}$ is independent of $\mathbf{Y}$ given $\mathbf{Z}$ in a probability distribution p. Any probability distribution p that can be represented by a BN with DAG G satisfies certain independencies between the random variables in $\mathbf{U}$ that can be read from G via the d-separation criterion, i.e. if *d-sep*$(\mathbf{X}, \mathbf{Y}|\mathbf{Z})$ then $\mathbf{X} \perp\!\!\!\perp \mathbf{Y}|\mathbf{Z}$ with $\mathbf{X}$, $\mathbf{Y}$ and $\mathbf{Z}$ three mutually disjoint subsets of $\mathbf{U}$. The statement *d-sep*$(\mathbf{X}, \mathbf{Y}|\mathbf{Z})$ is true when for every undirected path in G between a node in $\mathbf{X}$ and a node in $\mathbf{Y}$ there exists a node Z in the path such that either (i) Z does not have two parents in the path and $Z \in \mathbf{Z}$, or (ii) Z has two parents in the path and neither Z nor any of its descendants in G is in $\mathbf{Z}$. A probability distribution p is said to be faithful to a DAG G when $\mathbf{X} \perp\!\!\!\perp \mathbf{Y}|\mathbf{Z}$ iff *d-sep*$(\mathbf{X}, \mathbf{Y}|\mathbf{Z})$. Let p denote a probability distribution and $X \in \mathbf{U}$, any $\mathbf{Y} \subseteq (\mathbf{U} \setminus \{X\})$ such that $X \perp\!\!\!\perp (\mathbf{U} \setminus \mathbf{Y} \setminus \{X\})|\mathbf{Y}$ is called a Markov blanket of X. Any minimal Markov blanket of X is called a Markov boundary (MB) of X, i.e. no proper subset of a MB of X is a Markov blanket of X. The following three theorems are proven in [12], [17] and [12], respectively.

Theorem 1 *Let* $\mathbf{X}$, $\mathbf{Y}$, $\mathbf{Z}$ *and* $\mathbf{W}$ *denote four mutually disjoint subsets of* $\mathbf{U}$. *Any probability distribution* p *satisfies the following four properties: Symmetry* $\mathbf{X} \perp\!\!\!\perp \mathbf{Y}|\mathbf{Z} \Rightarrow \mathbf{Y} \perp\!\!\!\perp \mathbf{X}|\mathbf{Z}$, *decomposition* $\mathbf{X} \perp\!\!\!\perp (\mathbf{Y} \cup \mathbf{W})|\mathbf{Z} \Rightarrow \mathbf{X} \perp\!\!\!\perp \mathbf{Y}|\mathbf{Z}$, *weak union* $\mathbf{X} \perp\!\!\!\perp (\mathbf{Y} \cup \mathbf{W})|\mathbf{Z} \Rightarrow \mathbf{X} \perp\!\!\!\perp \mathbf{Y}|(\mathbf{Z} \cup \mathbf{W})$, *and contraction* $\mathbf{X} \perp\!\!\!\perp \mathbf{Y}|(\mathbf{Z} \cup \mathbf{W}) \wedge \mathbf{X} \perp\!\!\!\perp \mathbf{W}|\mathbf{Z} \Rightarrow \mathbf{X} \perp\!\!\!\perp (\mathbf{Y} \cup \mathbf{W})|\mathbf{Z}$. *If* p *is strictly positive, then* p *satisfies the previous four properties plus the intersection property* $\mathbf{X} \perp\!\!\!\perp \mathbf{Y}|(\mathbf{Z} \cup \mathbf{W}) \wedge \mathbf{X} \perp\!\!\!\perp \mathbf{W}|(\mathbf{Z} \cup \mathbf{Y}) \Rightarrow \mathbf{X} \perp\!\!\!\perp (\mathbf{Y} \cup \mathbf{W})|\mathbf{Z}$. *If* p *is faithful to a DAG* G, *then* p *satisfies the previous five properties plus the composition property* $\mathbf{X} \perp\!\!\!\perp \mathbf{Y}|\mathbf{Z} \wedge \mathbf{X} \perp\!\!\!\perp \mathbf{W}|\mathbf{Z} \Rightarrow \mathbf{X} \perp\!\!\!\perp (\mathbf{Y} \cup \mathbf{W})|\mathbf{Z}$ *and the local Markov property* $X \perp\!\!\!\perp (ND(X) \setminus Pa(X))|Pa(X)$ *for each* $X \in \mathbf{U}$.

Theorem 2 *If a probability distribution* p *is faithful to a DAG* G, *then (i) for each pair of nodes* X *and* Y *in* G, X *and* Y *are adjacent in* G *iff* $X \not\perp\!\!\!\perp Y|\mathbf{Z}$

Table 1
IAMB.

```
     IAMB(T)

     /* add true positives to MB */
 1   MB = ∅
 2   repeat
 3     Y = arg max_{X∈(U\MB\{T})} dep(T, X|MB)
 4     if T ⫫̸ Y|MB then
 5       MB = MB ∪ {Y}
 6   until MB does not change
     /* remove false positives from MB */
 7   for each X ∈ MB do
 8     if T ⫫ X|(MB \ {X}) then
 9       MB = MB \ {X}
10   return MB
```

for all $\mathbf{Z}$ *such that* $X, Y \notin \mathbf{Z}$*, and (ii) for each triplet of nodes* X*,* Y *and* Z *in* G *such that* X *and* Y *are adjacent to* Z *but* X *and* Y *are non-adjacent,* $X \rightarrow Z \leftarrow Y$ *is a subgraph of* G *iff* $X \not\perp\!\!\!\perp Y|\mathbf{Z}$ *for all* $\mathbf{Z}$ *such that* $X, Y \notin \mathbf{Z}$ *and* $Z \in \mathbf{Z}$*.*

Theorem 3 *If a probability distribution* p *satisfies the intersection property, then each* $X \in \mathbf{U}$ *has a unique MB,* $MB(X)$*. If* p *is faithful to a DAG* G*, then* $MB(X)$ *is the union of* $PC(X)$ *and the parents of the children of* X *in* G*.*

3 Previous Work on Scalable Learning of MBs

In this section, we review three algorithms for learning MBs from data, namely the incremental association Markov boundary algorithm (IAMB) [19], the max-min Markov boundary algorithm (MMMB) [20], and HITON-MB [1]. To our knowledge, these algorithms and some minor variants of them are the only algorithms for learning MBs from data that have experimentally been shown to scale to databases with thousands of features. However, we show that IAMB is data inefficient and that MMMB and HITON-MB do not guarantee the correct output under the faithfulness assumption. In the algorithms, $X \not\perp\!\!\!\perp Y|\mathbf{Z}$ ($X \perp\!\!\!\perp Y|\mathbf{Z}$) denotes (in)dependence with respect to a learning database D, and $dep(X, Y|\mathbf{Z})$ is a measure of the strength of the dependence with respect to D. In particular, the algorithms run a χ^2 independence test with the G^2 statistic in order to decide on $X \not\perp\!\!\!\perp Y|\mathbf{Z}$ or $X \perp\!\!\!\perp Y|\mathbf{Z}$ [17], and they use the negative p-value of the test as $dep(X, Y|\mathbf{Z})$. The three algorithms are based on the assumption that D is faithful to a DAG G, i.e. D is a sample from a probability distribution p faithful to G, and thus each node has a unique MB.

Table 1 outlines IAMB. The algorithm receives the target node T as input and returns $MB(T)$ in MB as output. The algorithm works in two steps. First, the nodes in $MB(T)$ are added to MB (lines 2-6). Since this step is based on the heuristic at line 3, some nodes not in $MB(T)$ may be added to MB as well. These nodes are removed from MB in the second step (lines 7-9). Tsamardinos et al. prove in [19] that IAMB is correct under the faithfulness assumption.

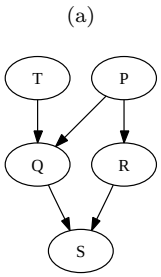
Theorem 4 *Under the assumptions that the independence tests are correct and that the learning database D is an independent and identically distributed sample from a probability distribution p faithful to a DAG G, IAMB(T) returns $MB(T)$.*

The assumption that the independence tests are correct means that they decide (in)dependence iff the (in)dependence holds in p. We elaborate further on this assumption in Section 6. In order to maximize accuracy in practice, IAMB performs a test if it is reliable and skips it otherwise. Following the approach in [17], IAMB considers a test to be reliable when the number of instances in D is at least five times the number of degrees of freedom in the test. This means that the number of instances required by IAMB to identify $MB(T)$ is at least exponential in the size of $MB(T)$, because the number of degrees of freedom in a test is exponential in the size of the conditioning set and the test to add to MB the last node in $MB(T)$ will be conditioned on at least the rest of the nodes in $MB(T)$. However, depending on the topology of G, it can be the case that $MB(T)$ can be identified by conditioning on sets much smaller than those used by IAMB, e.g. if G is a tree (see Sections 3.2 and 4). Therefore, IAMB is data inefficient because its data requirements can be unnecessarily high. Tsamardinos et al. are aware of this drawback and describe in [19] some variants of IAMB that alleviate it, though they do not solve it, while still being scalable and correct under the assumptions in Theorem 4: The second step can be run after each node addition at line 5, and/or the second step can be replaced by the PC algorithm [17]. Finally, as Tsamardinos et al. note in [19], IAMB is similar to the grow-shrink algorithm (GS) [10]. The only difference is that GS uses a simpler heuristic at line 3: $Y = arg\,max_{X \in (\mathbf{U} \setminus MB \setminus \{T\})} dep(T, X|\emptyset)$. GS is correct under the assumptions in Theorem 4, but it is data inefficient for the same reason as IAMB.

Table 2
MMPC and MMMB.

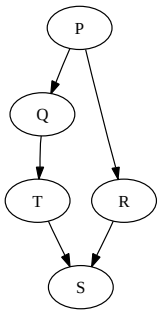
MMPC(T)

```
   /* add true positives to PC */
1  PC = ∅
2  repeat
3    for each X ∈ (U \ PC \ {T}) do
4      Sep[X] = arg min_{Z⊆PC} dep(T, X|Z)
5    Y = arg max_{X∈(U\PC\{T})} dep(T, X|Sep[X])
6    if T ⊥̸ Y|Sep[Y] then
7      PC = PC ∪ {Y}
8  until PC does not change
   /* remove false positives from PC */
9  for each X ∈ PC do
10   if T ⊥⊥ X|Z for some Z ⊆ PC \ {X} then
11     PC = PC \ {X}
12 return PC
```



MMMB(T)

```
   /* add true positives to MB */
1  PC = MMPC(T)
2  MB = PC
3  CanMB = (PC ∪_{X∈PC} MMPC(X)) \ {T}
   /* add more true positives to MB */
4  for each X ∈ CanMB \ PC do
5    find any Z such that T ⊥⊥ X|Z and T, X ∉ Z
6    for each Y ∈ PC do
7      if T ⊥̸ X|Z ∪ {Y} then
8        MB = MB ∪ {X}
9  return MB
```



3.2 MMMB

MMMB aims to reduce the data requirements of IAMB while still being scalable and correct under the faithfulness assumption. MMMB takes a divide-and-conquer approach that breaks the problem of identifying $MB(T)$ into two subproblems: First, identifying $PC(T)$ and, second, identifying the rest of the parents of the children of T in G. MMMB uses the max-min parents and children algorithm (MMPC) to solve the first subproblem. Table 2 outlines MMPC. The algorithm receives the target node T as input and returns $PC(T)$ in PC as output. MMPC is similar to IAMB, with the exception that MMPC considers any subset of the output as the conditioning set for the tests that it performs and IAMB only considers the output. Tsamardinos et al. prove in [20] that, under the assumptions in Theorem 4, the output of MMPC is $PC(T)$. However, this is not always true. The flaw in the proof is the assumption that if $X \notin PC(T)$, then $T \perp\!\!\!\perp X|\mathbf{Z}$ for some $\mathbf{Z} \subseteq PC(T)$ and, thus, any node not in $PC(T)$ that enters PC at line 7 is removed from it at line 11. This is not always true for the descendants of T. This is illustrated by running MMPC(T) with D faithful to the DAG (a) in Table 2. Neither P nor R enters PC at line 7 because $T \perp\!\!\!\perp P|\emptyset$ and $T \perp\!\!\!\perp R|\emptyset$. Q enters PC because $T \not\perp\!\!\!\perp Q|\mathbf{Z}$ for all $\mathbf{Z}$ such that $T, Q \notin \mathbf{Z}$. S enters PC because $T \not\perp\!\!\!\perp S|\emptyset$ and $T \not\perp\!\!\!\perp S|Q$. Then, $PC = \{Q, S\}$ at line 9. Neither Q nor S leaves PC at line 11. Consequently, the output of MMPC includes S which is not in $PC(T)$ and,

Table 3
CMMPC.

```
      CMMPC(T)

1     PC = ∅
2     for each X ∈ MMPC(T) do
3       if T ∈ MMPC(X) then
4         PC = PC ∪ {X}
5     return PC
```

thus, MMPC does not guarantee the correct output under the faithfulness assumption.

Table 2 outlines MMMB. The algorithm receives the target node T as input and returns $MB(T)$ in MB as output. The algorithm works in two steps. First, PC and MB are initialized with $PC(T)$ and $CanMB$ with $(PC(T) \cup_{X \in PC(T)} PC(X)) \setminus \{T\}$ by calling MMPC (lines 1-3). $CanMB$ contains the candidates to enter MB. Second, the parents of the children of T in G that are not yet in MB are added to it (lines 4-8). This step is based on the following observation. The parents of the children of T in G that are missing from MB at line 4 are those that are non-adjacent to T in G. These parents are in $CanMB \setminus PC$. Therefore, if $X \in CanMB \setminus PC$ and $Y \in PC$, then X and T are non-adjacent parents of Y in G iff $T \not\perp\!\!\!\perp X | \mathbf{Z} \cup \{Y\}$ for any $\mathbf{Z}$ such that $T \perp\!\!\!\perp X | \mathbf{Z}$ and $T, X \notin \mathbf{Z}$. Tsamardinos et al. prove in [20] that, under the assumptions in Theorem 4, the output of MMMB is $MB(T)$. However, this is not always true even if MMPC were correct under the faithfulness assumption. The flaw in the proof is the observation that motivates the second step of MMMB, which is not true. This is illustrated by running MMMB(T) with D faithful to the DAG (b) in Table 2. Let us assume that MMPC is correct under the faithfulness assumption. Then, $MB = PC = \{Q, S\}$ and $CanMB = \{P, Q, R, S\}$ at line 4. P enters MB at line 8 if $\mathbf{Z} = \{Q\}$ at line 5, because $P \in CanMB \setminus PC$, $S \in PC$, $T \perp\!\!\!\perp P | Q$ and $T \not\perp\!\!\!\perp P | \{Q, S\}$. Consequently, the output of MMMB can include P which is not in $MB(T)$ and, thus, MMMB does not guarantee the correct output under the faithfulness assumption even if MMPC were correct under this assumption.

In practice, MMMB performs a test if it is reliable and skips it otherwise. MMMB follows the same criterion as IAMB to decide whether a test is reliable or not. MMMB is data efficient because the number of instances required to identify $MB(T)$ does not depend on the size of $MB(T)$ but on the topology of G.

In [22], Tsamardinos et al. identify the flaw in MMPC and propose a corrected MMPC (CMMPC). The output of MMPC must be further processed in order to obtain $PC(T)$, because it may contain some descendants of T in G other than its children. Fortunately, these nodes can be easily identified: If X is in the output of MMPC(T), then X is a descendant of T in G other than one of its children iff T is not in the output of MMPC(X). CMMPC, which is

Table 4
HITON-PC and HITON-MB.

HITON-PC(T)

```
1   PC = ∅
2   CanPC = U \ {T}
3   repeat
    /* add the best candidate to PC */
4     Y = arg max_{X∈CanPC} dep(T, X|∅)
5     PC = PC ∪ {Y}
6     CanPC = CanPC \ {Y}
    /* remove false positives from PC */
7     for each X ∈ PC do
8       if T ⊥⊥ X|Z for some Z ⊆ PC \ {X} then
9         PC = PC \ {X}
10  until CanPC is empty
11  return PC
```

HITON-MB(T)

```
    /* add true positives to MB */
1   PC = HITON-PC(T)
2   MB = (PC ∪_{X∈PC} HITON-PC(X)) \ {T}
    /* remove false positives from MB */
3   for each X ∈ MB do
4     for each Y ∈ PC do
5       if T ⊥⊥ X|Z for some Z ⊆ {Y} ∪ (U \ {T, X, Y}) then
6         MB = MB \ {X}
7   return MB
```

outlined in Table 3, implements this observation. The algorithm receives the target node T as input and returns $PC(T)$ in PC as output. As shown above, however, correcting MMPC does not make MMMB correct. Independently of Tsamardinos et al., we identify and fix the flaws in both MMPC and MMMB in [13]. We discuss our work in Section 4.

3.3 HITON-MB

Like MMMB, HITON-MB aims to reduce the data requirements of IAMB while still being scalable and correct under the faithfulness assumption. Like MMMB, HITON-MB identifies $MB(T)$ by first identifying $PC(T)$ and, then, identifying the rest of the parents of the children of T in G. HITON-MB uses HITON-PC to solve the first subproblem. Table 4 outlines HITON-PC. The algorithm receives the target node T as input and returns $PC(T)$ in PC as output. HITON-PC is similar to MMPC, with the exception that the former interleaves the addition of the nodes in $PC(T)$ to PC (lines 4-5) and the removal from PC of the nodes that are not in $PC(T)$ but that have been added to PC by the heuristic at line 4 (lines 7-9). Note also that this heuristic is simpler than the one used by MMPC, because the conditioning set is always the empty set. Aliferis et al. prove in [1] that, under the assumptions in Theorem 4, the output of HITON-PC is $PC(T)$. However, this is not always true. The flaw in the proof is the same as that in the proof of correctness of MMPC. Running HITON-PC(T) with D faithful to the DAG (a) in Table 2

can produce the same incorrect result as MMPC(T). Obviously, the flaw in HITON-PC can be fixed in the exactly the same way as the flaw in MMPC was fixed above.

Table 4 outlines HITON-MB. The algorithm receives the target node T as input and returns $MB(T)$ in MB as output. HITON-MB is similar to MMMB. The algorithm works in two steps. First, PC and MB are initialized with $PC(T)$ and $(PC(T) \cup_{X \in PC(T)} PC(X)) \setminus \{T\}$, respectively, by calling HITON-PC (lines 1-2). Second, the nodes in MB that are neither in $PC(T)$ nor have a common child with T in G are removed from MB (lines 3-6). This step is based on the following observation. If $X \in MB$ and $Y \in PC$, then X must be removed from MB iff $T \perp\!\!\!\perp X | \mathbf{Z}$ for some $\mathbf{Z}$ such that $T, X \notin \mathbf{Z}$. Aliferis et al. prove in [1] that, under the assumptions in Theorem 4, the output of HITON-MB is $MB(T)$. However, this is not always true even if HITON-PC were correct under the faithfulness assumption. The flaw in the proof is the observation that motivates the second step of HITON-MB, which is not true. This is illustrated by running HITON-MB(T) with D faithful to the DAG (b) in Table 2. Let us assume that HITON-PC is correct under the faithfulness assumption. Then, $PC = \{Q, S\}$ and $MB = \{P, Q, R, S\}$ at line 3. P and R are removed from MB at line 6 because $Q \in PC$ and $T \perp\!\!\!\perp P | Q$ and $T \perp\!\!\!\perp R | Q$. Then, $MB = \{Q, S\}$ at line 7. Consequently, the output of HITON-MB does not include R which is in $MB(T)$ and, thus, HITON-MB does not guarantee the correct output under the faithfulness assumption even if HITON-PC were correct under this assumption.

In practice, HITON-MB performs a test if it is reliable and skips it otherwise. HITON-MB follows the same criterion as IAMB and MMMB to decide whether a test is reliable or not. HITON-MB is data efficient because the number of instances required to identify $MB(T)$ does not depend on the size of $MB(T)$ but on the topology of G.

4 Improving Data Efficiency

This section addresses the same question that motivated MMMB and HITON-MB: Can we develop an algorithm for learning MBs from data that is scalable, data efficient, and correct under the faithfulness assumption ? The answer is yes. We call this new algorithm the parents and children based Markov boundary algorithm (PCMB) and prove that, unlike MMMB and HITON-MB, it is correct under the faithfulness assumption. Like IAMB, MMMB and HITON-MB, PCMB is based on the assumption that the learning database D is faithful to a DAG G and, thus, each node has a unique MB.

PCMB takes a divide-and-conquer approach that breaks the problem of iden-

Table 5
GetPCD, GetPC and PCMB.

GetPCD(T)

1 $PCD = \emptyset$
2 $CanPCD = \mathbf{U} \setminus \{T\}$
3 **repeat**
  /* remove false positives from $CanPCD$ */
4 **for each** $X \in CanPCD$ **do**
5 $Sep[X] = arg\,min_{\mathbf{Z} \subseteq PCD} dep(T, X|\mathbf{Z})$
6 **for each** $X \in CanPCD$ **do**
7 **if** $T \perp\!\!\!\perp X|Sep[X]$ **then**
8 $CanPCD = CanPCD \setminus \{X\}$
  /* add the best candidate to PCD */
9 $Y = arg\,max_{X \in CanPCD} dep(T, X|Sep[X])$
10 $PCD = PCD \cup \{Y\}$
11 $CanPCD = CanPCD \setminus \{Y\}$
  /* remove false positives from PCD */
12 **for each** $X \in PCD$ **do**
13 $Sep[X] = arg\,min_{\mathbf{Z} \subseteq PCD \setminus \{X\}} dep(T, X|\mathbf{Z})$
14 **for each** $X \in PCD$ **do**
15 **if** $T \perp\!\!\!\perp X|Sep[X]$ **then**
16 $PCD = PCD \setminus \{X\}$
17 **until** PCD does not change
18 **return** PCD

GetPC(T)

1 $PC = \emptyset$
2 **for each** $X \in GetPCD(T)$ **do**
3 **if** $T \in GetPCD(X)$ **then**
4 $PC = PC \cup \{X\}$
5 **return** PC

PCMB(T)

  /* add true positives to MB */
1 $PC = GetPC(T)$
2 $MB = PC$
  /* add more true positives to MB */
3 **for each** $Y \in PC$ **do**
4 **for each** $X \in GetPC(Y)$ **do**
5 **if** $X \notin PC$ **then**
6 find $\mathbf{Z}$ st $T \perp\!\!\!\perp X|\mathbf{Z}$ and $T, X \notin \mathbf{Z}$
7 **if** $T \not\perp\!\!\!\perp X|\mathbf{Z} \cup \{Y\}$ **then**
8 $MB = MB \cup \{X\}$
9 **return** MB

tifying $MB(T)$ into two subproblems: First, identifying $PC(T)$ and, second, identifying the rest of the parents of the children of T in G. PCMB uses the functions GetPCD and GetPC to solve the first subproblem. $X \not\perp\!\!\!\perp Y|\mathbf{Z}$, $X \perp\!\!\!\perp Y|\mathbf{Z}$ and $dep(X, Y|\mathbf{Z})$ are the same as in Section 3. Table 5 outlines GetPCD. The algorithm receives the target node T as input and returns a superset of $PC(T)$ in PCD as output. The algorithm tries to minimize the number of nodes not in $PC(T)$ that are returned in PCD. The algorithm repeats three steps until PCD does not change. First, some nodes not in $PC(T)$ are removed from $CanPCD$, which contains the candidates to enter PCD (lines 4-8). This step is based on the observation that $X \in PC(T)$ iff $T \not\perp\!\!\!\perp X|\mathbf{Z}$ for all $\mathbf{Z}$ such that $T, X \notin \mathbf{Z}$. Second, the candidate most likely to be in $PC(T)$ is added to PCD and removed from $CanPCD$ (lines 9-11). Since this step is based on the heuristic at line 9, some nodes not in $PC(T)$ may be added to PCD as well. Some of these nodes are removed from PCD in the third step (lines 12-16). This step is based on the same observation as the first step.

Theorem 5 *Under the assumptions that the independence tests are correct and that the learning database D is an independent and identically distributed sample from a probability distribution p faithful to a DAG G, GetPCD(T) returns a superset of $PC(T)$ that does not include any node in $ND(T) \setminus Pa(T)$.*

Proof: First, we prove that the nodes in $PC(T)$ are included in the output PCD. If $X \in PC(T)$, then $T \not\perp\!\!\!\perp X|\mathbf{Z}$ for all $\mathbf{Z}$ such that $T, X \notin \mathbf{Z}$ owing to Theorem 2. Consequently, X enters PCD at line 10 and does not leave it thereafter.

Second, we prove that the nodes in $ND(T) \setminus Pa(T)$ are not included in the output PCD. It suffices to study the last time that lines 12-16 are executed. At line 12, $Pa(T) \subseteq PCD$ owing to the paragraph above. Therefore, if PCD still contains some $X \in ND(T) \setminus Pa(T)$, then $T \perp\!\!\!\perp X | \mathbf{Z}$ for some $\mathbf{Z} \subseteq PCD \setminus \{X\}$ owing to the local Markov property. Consequently, X is removed from PCD at line 16. □

The output of GetPCD must be further processed in order to obtain $PC(T)$, because it may contain some descendants of T in G other than its children. These nodes can be easily identified: If X is in the output of GetPCD(T), then X is a descendant of T in G other than one of its children iff T is not in the output of GetPCD(X). GetPC, which is outlined in Table 5, implements this observation. The algorithm receives the target node T as input and returns $PC(T)$ in PC as output. We prove that GetPC is correct under the faithfulness assumption.

Theorem 6 *Under the assumptions that the independence tests are correct and that the learning database D is an independent and identically distributed sample from a probability distribution p faithful to a DAG G, GetPC(T) returns $PC(T)$.*

Proof: First, we prove that the nodes in $PC(T)$ are included in the output PC. If $X \in PC(T)$, then $T \in PC(X)$. Therefore, X and T satisfy the conditions at lines 2 and 3, respectively, owing to Theorem 5. Consequently, X enters PC at line 4.

Second, we prove that the nodes not in $PC(T)$ are not included in the output PC. Let $X \notin PC(T)$. If X does not satisfy the condition at line 2, then X does not enter PC at line 4. On the other hand, if X satisfies the condition at line 2, then X must be a descendant of T in G other than one of its children and, thus, T does not satisfy the condition at line 3 owing to Theorem 5. Consequently, X does not enter PC at line 4. □

Finally, Table 5 outlines PCMB. The algorithm receives the target node T as input and returns $MB(T)$ in MB as output. The algorithm works in two steps. First, MB is initialized with $PC(T)$ by calling GetPC (line 2). Second, the parents of the children of T in G that are not yet in MB are added to it (lines 3-8). This step is based on the following observation. The parents of the children of T in G that are missing from MB at line 3 are those that are non-adjacent to T in G. Therefore, if $Y \in PC(T)$, $X \in PC(Y)$ and $X \notin PC(T)$, then X and T are non-adjacent parents of Y in G iff $T \not\perp\!\!\!\perp X | \mathbf{Z} \cup \{Y\}$ for any $\mathbf{Z}$ such that $T \perp\!\!\!\perp X | \mathbf{Z}$ and $T, X \notin \mathbf{Z}$. Note that $\mathbf{Z}$ can be efficiently obtained at line 6: GetPCD must have found such a $\mathbf{Z}$ and could have cached it for later retrieval. We prove that PCMB is correct under the faithfulness assumption.

Theorem 7 *Under the assumptions that the independence tests are correct*

and that the learning database D is an independent and identically distributed sample from a probability distribution p faithful to a DAG G, PCMB(T) returns $MB(T)$.

Proof: First, we prove that the nodes in $MB(T)$ are included in the output MB. Let $X \in MB(T)$. Then, either $X \in PC(T)$ or $X \notin PC(T)$ but X and T have a common child Y in G owing to Theorem 3. If $X \in PC(T)$, then X enters MB at line 2 owing to Theorem 6. On the other hand, if $X \notin PC(T)$ but X and T have a common child Y in G, then X satisfies the conditions at lines 3-5 owing to Theorem 6 and the condition at line 7 owing to Theorem 2. Consequently, X enters MB at line 8.

Second, we prove that the nodes not in $MB(T)$ are not included in the output MB. Let $X \notin MB(T)$. X does not enter MB at line 2 owing to Theorem 6. If X does not satisfy the conditions at lines 3-5, then X does not enter MB at line 8. On the other hand, if X satisfies the conditions at lines 3-5, then it must be due to either $T \rightarrow Y \rightarrow X$ or $T \leftarrow Y \leftarrow X$ or $T \leftarrow Y \rightarrow X$. Therefore, X does not satisfy the condition at line 7 owing to the faithfulness assumption. Consequently, X does not enter MB at line 8. □

In practice, PCMB performs a test if it is reliable and skips it otherwise. PCMB follows the same criterion as IAMB, MMMB and HITON-MB to decide whether a test is reliable or not. PCMB is data efficient because the number of instances required to identify $MB(T)$ does not depend on the size of $MB(T)$ but on the topology of G. For instance, if G is a tree, then PCMB does not need to perform any test that is conditioned on more than one node in order to identify $MB(T)$, no matter how large $MB(T)$ is. PCMB scales to databases with thousands of features because it does not require learning a complete BN. The experiments in the following section confirm it. Like IAMB, MMMB and HITON-MB, if the assumptions in Theorem 7 do not hold, then PCMB may not return a MB of T but an approximation. We discuss this issue further in Section 5.

4.1 Experimental Evaluation

In this section, we compare the performance of IAMB and PCMB through experiments on synthetic and real databases. We do not consider GS because IAMB outperforms it [19]. We do not consider MMMB and HITON-MB because we are not interested in any algorithm that does not guarantee the correct output under the faithfulness assumption. In order to ensure that IAMB converges to a local optimum, our implementation of it interleaves the first and second steps until convergence, i.e. if some node removal occurs at line 9, then IAMB jumps to line 2 after the second step is completed. This does not

Table 6
Results of the experiments with the Alarm and Pigs databases.

Database	Instances	Algorithm	Precision	Recall	Distance	Time
Alarm	100	IAMB	0.85±0.06	0.46±0.03	0.54±0.06	0±0
Alarm	100	PCMB	0.79±0.04	0.49±0.05	0.51±0.04	0±0
Alarm	200	IAMB	0.87±0.04	0.60±0.03	0.42±0.04	0±0
Alarm	200	PCMB	0.95±0.03	0.56±0.05	0.38±0.06	0±0
Alarm	500	IAMB	0.91±0.03	0.72±0.04	0.30±0.04	0±0
Alarm	500	PCMB	0.94±0.01	0.72±0.04	0.26±0.03	0±0
Alarm	1000	IAMB	0.93±0.03	0.80±0.01	0.22±0.02	0±0
Alarm	1000	PCMB	0.99±0.01	0.79±0.01	0.18±0.02	0±0
Alarm	2000	IAMB	0.92±0.04	0.83±0.01	0.22±0.04	0±0
Alarm	2000	PCMB	1.00±0.00	0.83±0.02	0.14±0.01	0±0
Alarm	5000	IAMB	0.92±0.02	0.86±0.01	0.18±0.02	0±0
Alarm	5000	PCMB	1.00±0.00	0.86±0.02	0.11±0.02	1±0
Alarm	10000	IAMB	0.92±0.04	0.90±0.01	0.14±0.03	0±0
Alarm	10000	PCMB	1.00±0.00	0.90±0.02	0.08±0.02	2±0
Alarm	20000	IAMB	0.94±0.00	0.92±0.00	0.10±0.00	1±0
Alarm	20000	PCMB	1.00±0.00	0.92±0.00	0.05±0.00	4±0
Pigs	100	IAMB	0.82±0.01	0.59±0.01	0.48±0.02	0±0
Pigs	100	PCMB	0.83±0.01	0.82±0.02	0.29±0.02	0±0
Pigs	200	IAMB	0.80±0.00	0.82±0.00	0.37±0.00	0±0
Pigs	200	PCMB	0.97±0.01	0.96±0.01	0.07±0.01	1±0
Pigs	500	IAMB	0.82±0.00	0.84±0.00	0.34±0.00	0±0
Pigs	500	PCMB	0.98±0.00	1.00±0.00	0.02±0.00	1±0

invalidate Theorem 4. Our implementation of IAMB and PCMB breaks ties at random. Both IAMB and PCMB are written in C++ and all the experiments below are run on a Pentium 2.4 GHz, 512 MB RAM and Windows 2000.

4.1.1 Synthetic Data

The experiments in this section focus on the accuracy and data efficiency of the algorithms, whereas the next section addresses their scalability. For this purpose, we consider databases sampled from two known BNs, the Alarm BN [7] and the Pigs BN [8]. These BNs have 37 and 441 nodes, respectively, and the largest MB consists of eight and 68 nodes, respectively. We run IAMB and PCMB with each node in each BN as the target random variable T and, then, report the average precision and recall over all the nodes for each BN. Precision is the number of true positives in the output divided by the number of nodes in the output. Recall is the number of true positives in the output divided by the number of true positives in the BN. We also combine precision and recall as $\sqrt{(1-\text{precision})^2 + (1-\text{recall})^2}$ to measure the Euclidean distance from perfect precision and recall. Finally, we also report the running time in seconds. The significance level for the independence tests is 0.01.

Table 6 summarizes the results of the experiments for different sample sizes. Each entry in the table shows average and standard deviation values over 10 databases (the same 10 databases for IAMB and PCMB). For the Alarm databases, both algorithms achieve similar recall but PCMB scores higher precision and, thus, shorter distance than IAMB. Therefore, PCMB usually returns fewer false positives than IAMB. The explanation is that PCMB per-

forms more tests than IAMB and this makes it harder for false positives to enter the output. Compare, for instance, the heuristic at line 3 in IAMB with the heuristic at line 9 in GetPCD and the double check at lines 2-3 in GetPC. For the Pigs databases where larger MBs exist, PCMB outperforms IAMB in terms of precision, recall and distance. For instance, PCMB correctly identifies the MB of the node 435 of the Pigs BN, which consists of 68 nodes, with 500 instances while IAMB performs poorly for this node and sample size (precision=1.00±0.00, recall=0.04±0.00 and distance=0.96±0.00). The explanation is that, unlike IAMB, PCMB does not need to condition on the whole MB to identify it. Consequently, we can conclude that PCMB is more accurate than IAMB because it is more data efficient. It is worth mentioning that we expect the two variants of IAMB mentioned in Section 3.1 to perform better than IAMB, as they carry out more tests, but worse than PCMB, as they still have to condition on the whole MB to identify it, e.g. they require a number of instances at least exponential in 68 for perfect precision and recall for the node 435 of the Pigs BN.

4.1.2 Real Data

The experiments in this section compare the ability of IAMB and PCMB to solve a real-world FSS problem involving thousands of features. Specifically, we consider the Thrombin database which was provided by DuPont Pharmaceuticals for KDD Cup 2001 and it is exemplary of the real-world drug design environment [2]. The database contains 2543 instances characterized by 139351 binary features. Each instance represents a drug compound tested for its ability to bind to a target site on thrombin, a key receptor in blood clotting. The features describe the three-dimensional properties of the compounds. Each compound is labelled with one out of two classes, either it binds to the target site or not. The task of KDD Cup 2001 was to learn a classifier from 1909 given compounds (learning data) in order to predict binding affinity and, thus, the potential of a compound as anti-clotting agent. There were 114 classifiers submitted to KDD Cup 2001, whose accuracy was evaluated by the organizers of the competition on the remaining 634 compounds (testing data). The accuracy of a classifier was computed as the average of the accuracy on true binding compounds and the accuracy on true non-binding compounds. Besides the huge number of features, the Thrombin database is challenging for two other reasons. First, the learning data are extremely imbalanced: Only 42 out of the 1909 compounds bind. Second, the testing data are not sampled from the same probability distribution as the learning data, because the compounds in the testing data were synthesized based on the assay results recorded in the learning data. Scoring higher than 60 % accuracy is impressive [2].

To solve the FSS problem for the Thrombin database, we run IAMB and

Table 7
Results of the experiments with the Thrombin database.

Algorithm	**Features**	**Accuracy**	**Time**
IAMB	6±0	54±0	2426±72
PCMB	3±1	63±0	7302±1012
No FSS	139351	50	Not available
Winner of KDD Cup 2001	4	68	Not available
NB with the features in the winner of KDD Cup 2001	4	67	Not available
Taking into account the imbalance of the learning data	6	77	Not available
Taking into account the distribution of the testing data	15	83	Not available

PCMB with the class random variable as the target random variable T. Unlike in the previous section, we cannot now assess the performance of IAMB and PCMB by comparing their outputs with $MB(T)$ because this is unknown. Instead, we assess the performance of IAMB (PCMB) as the accuracy on the testing data of a naive Bayesian classifier (NB) trained on the learning data corresponding to only the features selected by IAMB (PCMB): The higher the accuracy the better the features selected and, thus, the better the algorithm used to select them. In order to train the NBs, we use the MLC++ software with the default parameters, except for the Laplace correction that is switched on [9]. We also report the number of features selected and the running time in seconds. As in the section above, the significance level for the independence tests in PCMB is 0.01. For IAMB, however, better results are obtained when the significance level for the independence tests is 0.0001. This significance level seems to avoid better than 0.01 the spurious dependencies that may exist in the learning data due to the large number of features. In the case of PCMB, it seems that the criterion for a node to enter the output, which is considerably more stringent than that in IAMB, suffices to avoid the spurious dependencies.

Table 7 summarizes the results of the experiments. The table shows average and standard deviation values over 114 runs for IAMB and PCMB because ties, which are broken at random, are common due to the high dimensionality of the learning data. Clearly, PCMB returns smaller and more accurate MBs than IAMB. Specifically, PCMB scores higher than 60 % accuracy in all the runs, which is impressive according to [2]. For instance, the MB to which PCMB converges most often (39 out of the 114 runs) scores 63 % accuracy and contains only three features (12810, 79651 and 91839). Regarding running time, PCMB is slower than IAMB because, as we have discussed in the previous section, it performs more tests. All in all, our results illustrate that both algorithms are scalable. We note that no existing algorithm for learning BNs from data can handle such a high-dimensional database as the Thrombin database. Hence, the importance of developing algorithms for learning MBs from data that, like IAMB and PCMB, avoid learning a complete BN as an intermediate step.

Table 7 compiles some other results that we now describe further. We do not report the baseline accuracy of a NB with no FSS because our computer

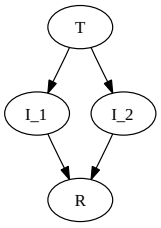
cannot run the MLC++ software with all the 139351 features. This illustrates the importance of FSS. In [23], 50 % accuracy is reported for a support vector machine with no FSS. The winner of KDD Cup 2001 was a tree augmented naive Bayesian classifier with four features scoring 68 % accuracy. A NB with these four features scores 67 % accuracy. In [23], a classifier that takes into account that the learning data are imbalanced reaches 77 % accuracy with six features, and another classifier that takes into account the distribution of the unlabelled testing data achieves 83 % accuracy with 15 features. Since the features used by these two classifiers are not reported in [23], we cannot calculate the accuracy of a NB with only those features. In any case, the 67 % accuracy of the NB with only the features in the winner of KDD Cup 2001 suffices to conclude that IAMB and PCMB return suboptimal solutions to the FSS problem for the Thrombin database. The causes of this suboptimal behavior lie, on one hand, in the data inefficiency of IAMB and, on the other hand, in the divide-and-conquer approach that PCMB takes, that is justified if the faithfulness assumption holds but that may hurt performance otherwise (see more evidence on Section 5.1). We believe that in order to improve the performance of IAMB and PCMB we have to relax the faithfulness assumption that underlies PCMB and avoid the data inefficiency of IAMB. We address this question in the next section.

5 Relaxing the Faithfulness Assumption

This section studies the following question: Can we relax the faithfulness assumption and develop an algorithm for learning MBs from data that is correct, scalable and data efficient ? We prove that IAMB is still correct under the composition property assumption, which is weaker than the faithfulness assumption (Theorem 1). We propose a stochastic variant of IAMB that can overcome the data inefficiency of IAMB while being scalable and correct under the composition property assumption. We show with experiments on the Thrombin database that this new algorithm can outperform IAMB and PCMB considerably often.

Theorem 8 *Under the assumptions that the independence tests are correct and that the learning database D is an independent and identically distributed sample from a probability distribution p satisfying the composition property, IAMB(T) returns a MB of T.*

Proof: First, we prove that MB is a Markov blanket of T when the loop in lines 2-6 is left. Let us suppose that this is not the case, i.e. $T \not\perp\!\!\!\perp (\mathbf{U} \setminus MB \setminus \{T\})|MB$ when the loop in lines 2-6 is left. Then, there exists $X \in (\mathbf{U} \setminus MB \setminus \{T\})$ such that $T \not\perp\!\!\!\perp X|MB$ due to the composition property assumption. This contradicts the assumption that the loop in lines 2-6 is left



$p(T)$

$T=0$	$T=1$	$T=2$	$T=3$
1/6	2/6	2/6	1/6

$p(I_1|T)$

T	$I_1=0$	$I_1=1$	$I_1=2$	$I_1=3$
0, 1	$(1-\eta)/2$	$(1-\eta)/2$	$\eta/2$	$\eta/2$
2, 3	$\eta/2$	$\eta/2$	$(1-\eta)/2$	$(1-\eta)/2$

$p(I_2|T)$

T	$I_2=0$	$I_2=1$	$I_2=2$	$I_2=3$
0, 2	$(1-\eta)/2$	$\eta/2$	$(1-\eta)/2$	$\eta/2$
1, 3	$\eta/2$	$(1-\eta)/2$	$\eta/2$	$(1-\eta)/2$

$p(R|I_1, I_2)$

I_1	I_2	$R=0$	$R=1$	$R=2$	$R=3$
0, 1	0, 2	$1-\eta$	$\eta/3$	$\eta/3$	$\eta/3$
0, 1	1, 3	$\eta/3$	$1-\eta$	$\eta/3$	$\eta/3$
2, 3	0, 2	$\eta/3$	$\eta/3$	$1-\eta$	$\eta/3$
2, 3	1, 3	$\eta/3$	$\eta/3$	$\eta/3$	$1-\eta$

Fig. 1. BN for the integer transmission example ($\eta \in (0, 1/2)$).

due to the assumption that the independence tests are correct.

Second, we prove that MB remains a Markov blanket of T after each node removal in line 9. It suffices to note that the independence tests are assumed to be correct and that $T \perp\!\!\!\perp (\mathbf{U} \setminus MB \setminus \{T\})|MB$ and $T \perp\!\!\!\perp X|(MB \setminus \{X\})$ yield $T \perp\!\!\!\perp (\mathbf{U} \setminus (MB \setminus \{X\}) \setminus \{T\})|(MB \setminus \{X\})$ due to the contraction property.

Third, we prove that MB is a minimal Markov blanket of T in line 10. Let us suppose that this is not the case, i.e. there exists $\mathbf{M} \subset MB$ in line 10 such that $\mathbf{M}$ is a Markov blanket of T. Let $X \in (MB \setminus \mathbf{M})$ and $\mathbf{Y} \subseteq (\mathbf{U} \setminus \mathbf{M} \setminus \{T\} \setminus \{X\})$. Then, $T \perp\!\!\!\perp (\mathbf{U} \setminus \mathbf{M} \setminus \{T\})|\mathbf{M}$ and, thus, $T \perp\!\!\!\perp (\mathbf{Y} \cup \{X\})|\mathbf{M}$ due to the decomposition property and, thus, $T \perp\!\!\!\perp X|(\mathbf{M} \cup \mathbf{Y})$ due to the weak union property. This contradicts the assumption that $\mathbf{M} \subset MB$, because any $X \in (MB \setminus \mathbf{M})$ would have been removed from MB in line 9 due to the assumption that the independence tests are correct. □

The following result, which we borrow from [3], illustrates that the composition property assumption is much weaker than the faithfulness assumption. If a probability distribution $p(\mathbf{U}, \mathbf{H}, \mathbf{S})$ is faithful to a DAG G over $\{\mathbf{U}, \mathbf{H}, \mathbf{S}\}$, then $p(\mathbf{U}) = \sum_{\mathbf{h}} p(\mathbf{U}, \mathbf{H} = \mathbf{h}, \mathbf{S} = \mathbf{s})$ satisfies the composition property, though it may not be faithful to any DAG. In other words, G can include some hidden nodes $\mathbf{H}$ and some selection bias $\mathbf{S} = \mathbf{s}$. Moreover, this result holds not only for DAGs and the d-separation criterion but for any graph and any criterion that is based on vertex separation.

As mentioned before, false positives may enter MB at line 5 in IAMB because the heuristic at line 3 is greedy. An example, inspired by [21], follows. An integer between 0 and 3 is sent from a transmitter station T to a receiver station

Table 8
KIAMB.

```
     KIAMB(T, K)

     /* add true positives to MB */
 1   MB = ∅
 2   repeat
 3     CanMB = ∅
 4     for each X ∈ (U \ MB \ {T}) do
 5       if T ⫫̸ X|MB then
 6         CanMB = CanMB ∪ {X}
 7     CanMB2 = random subset of CanMB with size max(1, ⌊(|CanMB| · K)⌋)
 8     Y = arg max_{X∈CanMB2} dep(T, X|MB)
 9     MB = MB ∪ {Y}
10   until MB does not change
     /* remove false positives from MB */
11   for each X ∈ MB do
12     if T ⫫ X|(MB \ {X}) then
13       MB = MB \ {X}
14   return MB
```

R through two intermediary stations I_1 and I_2. T does not send the integer to I_1 but only wether it is in $\{0, 1\}$ or in $\{2, 3\}$. Likewise, T only communicates to I_2 whether the integer is in $\{0, 2\}$ or in $\{1, 3\}$. Fig. 1 depicts a BN for this example, where $\eta \in (0, 1/2)$ represents the noise in the transmission. Since $p(T, I_1, I_2, R)$ satisfies the faithfulness assumption, $\{I_1, I_2\}$ is the unique MB of T. If η is positive but small enough, then R has more information about the integer transmitted by T than I_1 or I_2 alone. Thus, owing to the greediness of the heuristic at line 3, we expect IAMB(T) to first add R, then add I_1 and I_2 in any order, and finally remove R. This sequence of node additions and removals is less data efficient and more prone to errors than directly adding I_1 and I_2 in any order, because the former sequence requires three independence tests to decide dependence while the latter requires only two. Therefore, the sequence that IAMB tries may not be the most data efficient and safe sequence available. This leads us to propose a stochastic variant of IAMB, called KIAMB, that can try different sequences when run repeatedly. Hopefully, some of these sequences are more data efficient and less prone to errors than the one used by IAMB, e.g. if they add fewer false positives.

Table 8 outlines KIAMB. KIAMB differs from IAMB in that it allows the user to specify the trade-off between greediness and randomness in the search through an input parameter $K \in [0, 1]$. IAMB corresponds to KIAMB with $K = 1$. Therefore, while IAMB greedily adds to MB the most dependant node in $CanMB$ which contains the candidates to enter MB, KIAMB adds to MB the most dependant node in $CanMB2$ which is a random subset of $CanMB$ with size $max(1, \lfloor(|CanMB| \cdot K)\rfloor)$ where $|CanMB|$ is the size of $CanMB$ (lines 7-9). The proof of Theorem 8 is also valid for the following theorem.

Theorem 9 *Under the assumptions that the independence tests are correct and that the learning database D is an independent and identically distributed sample from a probability distribution p satisfying the composition property, KIAMB(T, K) returns a MB of T for any value of K.*

We note that Theorems 8 and 9 say "a MB of T" and not "the MB of T" because, unlike the faithfulness assumption, the composition property assumption does not imply that T has an unique MB. A necessary condition for the existence of more than one MB of T is that p does not satisfy the intersection property (Theorem 3), which implies that p cannot be strictly positive (Theorem 1). A simple example of a probability distribution p that satisfies the composition property and has several MBs of T involves two other random variables X and Y such that $T = X = Y$: Both $\{X\}$ and $\{Y\}$ are MBs of T. A more elaborated example is the integer transmission scenario introduced above with $p(T)$ uniform and $\eta = 0$: Both $\{I_1, I_2\}$ and $\{R\}$ are MBs of T.[1] The following theorem extends Theorem 9 with the guarantee that KIAMB with $K = 0$ can discover any MB of T.

Theorem 10 *Under the assumptions that the independence tests are correct and that the learning database D is an independent and identically distributed sample from a probability distribution p satisfying the composition property, KIAMB($T, 0$) returns a MB of T. The MB of T returned is any MB of T with non-zero probability.*

Proof: MB is a MB of T in line 14 due to Theorem 9. Let us assume that T has several MBs. We now prove that MB is any MB of T in line 14 with non-zero probability. Let $\mathbf{M} \subseteq (\mathbf{U} \setminus \{T\})$ be any MB of T. First, we prove that if $MB \subset \mathbf{M}$ before the node addition in line 9, then $MB \subseteq \mathbf{M}$ with non-zero probability after the node addition. Let us assume that $MB \subset \mathbf{M}$ before the node addition in line 9. Then, $T \not\perp\!\!\!\perp (\mathbf{U} \setminus MB \setminus \{T\})|MB$ and $T \perp\!\!\!\perp (\mathbf{U} \setminus \mathbf{M} \setminus \{T\})|\mathbf{M}$. These two statements together yield $T \not\perp\!\!\!\perp (\mathbf{M} \setminus MB)|MB$ due to the contraction property and, thus, there exists $X \in (\mathbf{M} \setminus MB)$ such that $T \not\perp\!\!\!\perp X|MB$ due to the composition property assumption. Therefore, X is added to MB in line 9 with non-zero probability due to $K = 0$ and the assumption that the independence tests are correct.

Second, we prove that $MB = \mathbf{M}$ in line 14 with non-zero probability. The paragraph above guarantees that $MB = \mathbf{M}$ with non-zero probability when the loop in lines 2-10 is left, and the assumption that the independence tests are correct guarantees that none of the nodes in MB is removed from it in line 13. □

The theorem above does not hold for IAMB, e.g. IAMB always returns $\{R\}$ in the integer transmission example with $p(T)$ uniform and $\eta = 0$ because the

[1] To show that this modification of the integer transmission example satisfies the composition property, we reformulate it as follows. Let B_1 and B_2 be the first and second bits, respectively, of the binary code corresponding to the integer sent by T. Then, $T = R = \{B_1, B_2\}$, $I_1 = \{B_1\}$, $I_2 = \{B_2\}$, and $\mathbf{X} \perp\!\!\!\perp \mathbf{Y}|\mathbf{Z}$ iff $(\mathbf{X} \setminus \mathbf{Z}) \cap \mathbf{Y} = \emptyset$. Consequently, $\mathbf{X} \perp\!\!\!\perp \mathbf{Y}|\mathbf{Z} \wedge \mathbf{X} \perp\!\!\!\perp \mathbf{W}|\mathbf{Z} \Rightarrow (\mathbf{X} \setminus \mathbf{Z}) \cap \mathbf{Y} = \emptyset \wedge (\mathbf{X} \setminus \mathbf{Z}) \cap \mathbf{W} = \emptyset \Rightarrow (\mathbf{X} \setminus \mathbf{Z}) \cap (\mathbf{Y} \cup \mathbf{W}) = \emptyset \Rightarrow \mathbf{X} \perp\!\!\!\perp (\mathbf{Y} \cup \mathbf{W})|\mathbf{Z}$.

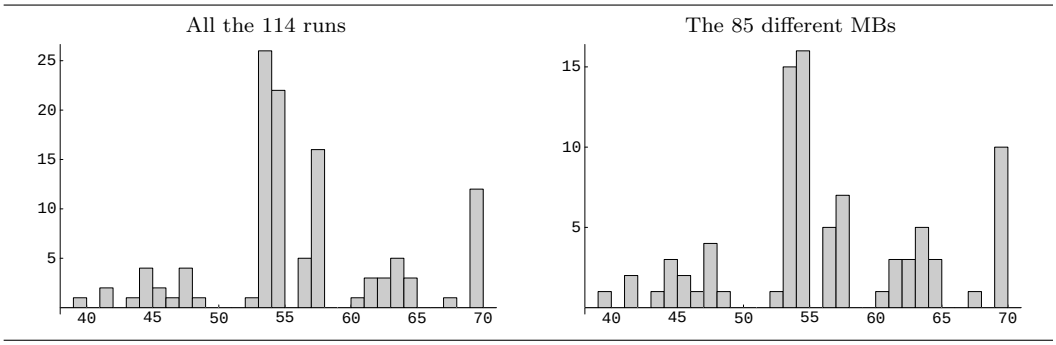


Fig. 2. Histograms of the accuracy of the runs of KIAMB (accuracy in the horizontal axis and number of runs in the vertical axis).

heuristic at line 3 is greedy. However, in some cases IAMB can return any MB of T by just breaking ties at random, e.g. in the $T = X = Y$ example. The theorem above guarantees that KIAMB with $K = 0$ discovers all the MBs of T if run repeatedly enough times. However, since T can have many MBs, it may be more realistic to say that running KIAMB repeatedly with $K \neq 1$ has the potential to discover, if not all, at least several MBs of T. This ability to generate alternative hypothesis is important in domains such as bioinformatics and medicine [16]. For instance, if the nodes in a MB of T represent genetic markers for a disease T, then the more MBs of T are identified the more biological insight into the disease T is gained.

5.1 Experimental Evaluation

In the previous section, we have argued that KIAMB can outperform IAMB because it can follow a sequence of node additions and removals that is more data efficient and, thus, less prone to errors. In this section, we confirm it by adding to the 114 runs of IAMB in Section 4.1.2 the results of 114 runs of KIAMB with $K = 0.8$. Preliminary experiments indicated that $K \in [0.7, 0.9]$ performs best. We run each algorithm 114 times to guarantee that our results are comparable with those of KDD Cup 2001, which had 114 participants. Our implementation of KIAMB reuses most of the code of IAMB in Section 4.1.2, including the interleaving of the first and second steps until convergence. This does not invalidate any of the theorems in the previous section. The rest of the experimental setting is the same as in Section 4.1.2.

The 114 runs of IAMB return 39 different MBs, all of them containing six features: 62 runs corresponding to 19 different MBs score 53 % accuracy, and 52 runs corresponding to 20 different MBs score 54 % accuracy. The 114 runs of KIAMB return 85 different MBs, containing from four to seven features. The left histogram in Fig. 2 summarizes the accuracy of the 114 runs of KIAMB, whereas the right histogram in the figure summarizes the accuracy of the 85 different MBs found in these runs. It is clear from the histograms that

KIAMB is able to identify many MBs that outperform those found by IAMB. To be exact, 49 of the 114 runs of KIAMB corresponding to 38 different MBs score higher than 54 % accuracy, which is the highest accuracy obtained by IAMB. Only 28 of the 114 participants in KDD Cup 2001 scored higher than 54 % accuracy. Furthermore, 12 of the 114 runs of KIAMB corresponding to 10 different MBs score 69 % accuracy, whereas the winner of KDD Cup 2001 scored 68 % accuracy. Our 10 MBs scoring 69 % accuracy contain six features. These are 3392, 63916, 79651, 135817, 138924 and, then, one of the following ones: 46937, 48386, 49864, 51230, 55132, 63853, 63856, 73697, 103235 or 108355. Regarding running time, IAMB takes 2426±72 seconds per run and KIAMB 2408±442. Therefore, our results confirm that KIAMB is scalable and that it can outperform IAMB considerably often.

The histograms in Fig. 2 also warn that KIAMB can perform worse than IAMB. To be exact, 17 of the 114 runs of KIAMB corresponding to 16 different MBs score lower than 53 % accuracy, which is the lowest accuracy obtained by IAMB. We note that 79 of the 114 participants in KDD Cup 2001 scored lower than 53 % accuracy. Therefore, KIAMB should be seen as a tool to generate alternative MBs, each of which should be validated before being accepted. The validation may resort to expert knowledge of the domain at hand. Or, as we have done in this paper, it may use some testing data. Obviously, having to hold some data out of the learning process for testing purposes is a drawback when the data available are scarce. However, we prefer it to running IAMB on all the data available which, as our experiments show, may produce a rather inaccurate MB. It is necessary warning that selecting the most accurate MB on some testing data out of the MBs obtained by running KIAMB repeatedly on some learning data may result in an overfitted MB if the number of repeated runs is too large [11]. We do not think this is an issue in our experiments because 114 is not such a large number of runs. In any case, the risk of overfitting in our experiments should be comparable to that in KDD Cup 2001, because we run KIAMB as many times as there were participants in KDD Cup 2001. Moreover, we do not simply compare the best MB obtained via KIAMB with the winner of KDD Cup 2001 but we compare the whole distributions of results, which makes our conclusions more robust against overfitting.

The histograms in Fig. 2 show that KIAMB can also outperform PCMB considerably often. As a matter of fact, 16 of the 114 runs of KIAMB corresponding to 14 different MBs score higher than 63 % accuracy, which is the accuracy obtained by all the 114 runs of PCMB in Section 4.1.2. Only five of the 114 participants in KDD Cup 2001 scored higher than 63 % accuracy. Finally, we report 114 runs of a stochastic version of PCMB, called KPCMB, in which the greedy heuristic at line 9 in GetPCD is modified to trade-off greediness and randomness through an input parameter $K \in [0, 1]$ in exactly the same way as IAMB was modified to develop KIAMB. We use $K = 0.8$ in the ex-

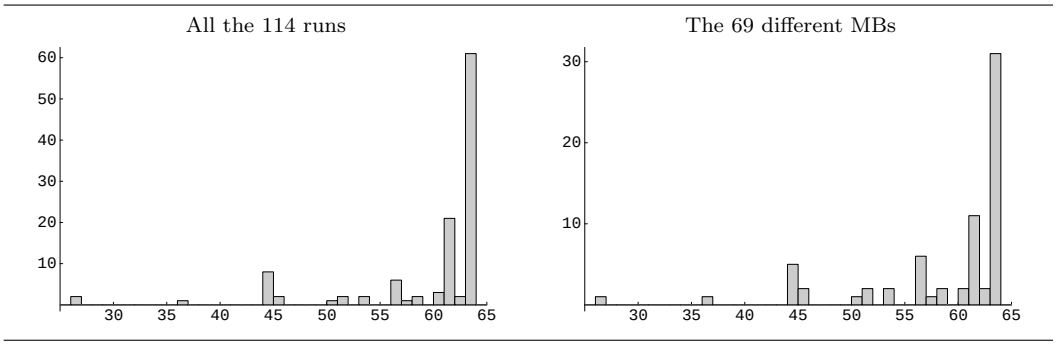


Fig. 3. Histograms of the accuracy of the runs of KPCMB (accuracy in the horizontal axis and number of runs in the vertical axis).

periments. The left histogram in Fig. 3 summarizes the accuracy of the 114 runs of KPCMB, whereas the right histogram in the figure summarizes the accuracy of the 69 different MBs found in these runs. Surprisingly, none of the runs of KPCMB scores higher than the 63 % accuracy of PCMB but 53 runs corresponding to 38 different MBs score lower than that. The reason of such poor performance lies in that KPCMB does not always return a MB of T, because there may exist some nodes not in the output of KPCMB that are dependent of T given the output. For instance, the worst run of KPCMB scores 26 % accuracy and returns the features 3392, 79651 and 135817, but $T \not\perp\!\!\!\perp 46937 | \{3392, 79651, 135817\}$. However, neither these four features are a MB of T because $T \not\perp\!\!\!\perp 63916 | \{3392, 46937, 79651, 135817\}$. Neither these five features are a MB of T because $T \not\perp\!\!\!\perp 138924 | \{3392, 46937, 63916, 79651, 135817\}$. Now, these six features are a MB of T. Actually, they are one of the 10 MBs scoring 69 % accuracy that are found by KIAMB. The reason why KPCMB may not return a MB of T lies in the divide-and-conquer approach that it takes, that is justified if the faithfulness assumption holds but that may hurt performance otherwise. In other words, the solutions to the subproblems that KPCMB obtains with the help of GetPCD and GetPC may not combine into a solution to the original problem of learning a MB of T. This illustrates the importance of developing algorithms for learning MBs from data that, like KIAMB, avoid the faithfulness assumption while being data efficient.

6 Discussion

In this paper, we have reported the results of our research on learning MBs from data. We have presented algorithms for such a task, studied the conditions under which they are correct, scalable and data efficient, and evaluated them on synthetic and real databases. Specifically, we have introduced PCMB, an algorithm that is scalable, data efficient, and correct under the faithfulness assumption. Then, we have proven that IAMB is correct under the composition property assumption. Finally, we have introduced KIAMB, an algorithm

that aims to overcome the data inefficiency of IAMB while being scalable and correct under the composition property assumption. The experimental results have shown that PCMB outperforms IAMB, and that KIAMB can outperform IAMB and PCMB considerably often. The experimental results have also confirmed that these algorithms can scale to high-dimensional domains. The reason is that they do not require learning a BN first, which can be painfully time consuming in high-dimensional domains [22]. This is particularly true for those algorithms for learning BNs from data that are (asymptotically) correct under the faithfulness or composition property assumption [22], which are the ones we are interested in.

It is worth mentioning that the proofs of correctness of the algorithms in this paper assume that the independence tests are correct. If the tests are simply consistent, then the proofs of correctness become proofs of consistency, because the algorithms perform a finite number of tests. Kernel-based independence tests that are consistent for any probability distribution exist [5,6]. The probability of error for these tests decays exponentially to zero when the sample size goes to infinity.

It is also worth mentioning that, throughout this paper, we have assumed that all the random variables are discrete. However, the results in this paper remain valid when all the random variables except the target node are continuous. Furthermore, they also remain valid when all the random variables (including the target node) are continuous. It suffices to replace the χ^2 independence test by an appropriate independence test, e.g. Student's t-test or Fisher's z test. The case where all the random variables (including the target node) are continuous is particularly interesting if the learning database is assumed to be a sample from a Gaussian probability distribution, because any Gaussian probability distribution satisfies the composition property, no matter whether it is faithful to some DAG or not [18]. Therefore, IAMB and KIAMB are correct if the learning database is assumed to be a sample from a Gaussian probability distribution. Such an assumption is common in many domains, e.g. when learning genetic regulatory networks from gene expression databases [15].

We are currently working on a scalable divide-and-conquer algorithm similar to PCMB that is data efficient as well as correct under the composition property assumption. At the same time, we are applying the results in this paper to solve the FSS problem for gene expression databases with thousands of features but hundreds of instances at most. Since the existing algorithms for learning BNs from data can be painfully time consuming for such high-dimensional database [22], it is very important to develop algorithms for learning MBs from data that, like those in this paper, avoid learning a complete BN as an intermediate step. An alternative approach is to reduce the search space so as to reduce the computational cost of the existing algorithms for learning BNs from data.

For instance, [4,22] propose restricting the search for the parents of each node to a small set of candidate parents that are selected in advance. According to the experiments in the latter paper, the algorithm proposed in that paper performs better. However, both algorithms lack a proof of (asymptotic) correctness under the faithfulness assumption. Moreover, it seems unnecessarily time consuming to learn a complete BN to solve the FSS problem, because we are only interested in a very specific part of it, namely $MB(T)$. Based on this idea and the results in this paper, we have recently presented in [14] an algorithm that learns a BN for the nodes in the neighborhood of a given node. This algorithm allows us to cope with high-dimensional data by learning a local BN around a node of interest rather than a complete BN model of the data.

Acknowledgements

We thank the editors and anonymous referees for their comments. This work is funded by the Swedish Research Council (VR-621-2005-4202), the Swedish Foundation for Strategic Research, and Linköping University.

References

[1] Aliferis, C. F., Tsamardinos, I., Statnikov, A.: HITON, a Novel Markov Blanket Algorithm for Optimal Variable Selection. In Proceedings of the 2003 American Medical Informatics Association Annual Symposium (2003) 21-25.

[2] Cheng, J., Hatzis, C., Hayashi, H., Krogel, M. A., Morishita, S., Page, D., Sese, J.: KDD Cup 2001 Report. ACM SIGKDD Explorations 3 (2002) 1-18. See also http://www.cs.wisc.edu/∼dpage/kddcup2001/

[3] Chickering D. M., Meek C.: Finding Optimal Bayesian Networks. In Proceedings of the Eighteenth Conference on Uncertainty in Artificial Intelligence (2002) 94-102.

[4] Friedman, N., Nachman, I., Peér, D.: Learning Bayesian Network Structure from Massive Datasets: The "Sparse Candidate" Algorithm. In Proceedings of the Fifteenth Conference on Uncertainty in Artificial Intelligence (1999) 206-215.

[5] Gretton, A., Herbrich, R., Smola, A., Bousquet, O., Schölkopf, B.: Kernel Methods for Measuring Independence. Journal of Machine Learning Research 6 (2005) 2075-2129.

[6] Gretton, A., Smola, A., Bousquet, O., Herbrich, R., Belitski, A., Augath, M., Murayama, Y., Pauls, J., Schölkopf, B., Logothetis, N.: Kernel Constrained

Covariance for Dependence Measurement. In Proceedings of the Tenth International Workshop on Artificial Intelligence and Statistics (2005) 1-8.

[7] Herskovits, E. H.: Computer-Based Probabilistic-Network Construction. PhD Thesis, Stanford University (1991).

[8] Jensen, C. S.: Blocking Gibbs Sampling for Inference in Large and Complex Bayesian Networks with Applications in Genetics. PhD Thesis, Aalborg University (1997).

[9] Kohavi, R., Sommerfield, D., Dougherty, J.: Data Mining Using MLC++: A Machine Learning Library in C++. In Tools with Artificial Intelligence (1996) 234-245.

[10] Margaritis, D., Thrun, S.: Bayesian Network Induction via Local Neighborhoods. In Proceedings of Neural Information Processing Systems (2000) 505-511.

[11] Ng, A.: Preventing "Overfitting" of Cross-Validation Data. In Proceedings of the Fourteenth International Conference on Machine Learning (1997) 245-253.

[12] Pearl, J.: Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference. Morgan Kaufmann (1988).

[13] Peña, J. M., Björkegren, J., Tegnér, J.: Scalable, Efficient and Correct Learning of Markov Boundaries under the Faithfulness Assumption. In Proceedings of the Eighth European Conference on Symbolic and Quantitative Approaches to Reasoning under Uncertainty - Lecture Notes in Artificial Intelligence 3571, 136-147.

[14] Peña, J. M., Björkegren, J., Tegnér, J.: Growing Bayesian Network Models of Gene Networks from Seed Genes. Bioinformatics 21 (2005) ii224-ii229.

[15] Schäfer, J., Strimmer, K.: An Empirical Bayes Approach to Inferring Large-Scale Gene Association Networks. Bioinformatics 21 (2005) 754-764.

[16] Sebastiani, P., Gussoni, E., Kohane, I. S., Ramoni, M.: Statistical Challenges in Functional Genomics (with Discussion). Statistical Science 18 (2003) 33-60.

[17] Spirtes, P., Glymour, C., Scheines, R.: Causation, Prediction, and Search. Springer-Verlag (1993).

[18] Studený, M.: Probabilistic Conditional Independence Structures. Springer (2005).

[19] Tsamardinos, I., Aliferis, C. F., Statnikov, A.: Algorithms for Large Scale Markov Blanket Discovery. In Proceedings of the Sixteenth International Florida Artificial Intelligence Research Society Conference (2003) 376-380.

[20] Tsamardinos, I., Aliferis, C. F., Statnikov, A.: Time and Sample Efficient Discovery of Markov Blankets and Direct Causal Relations. In Proceedings of the Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (2003) 673-678.

[21] Tsamardinos, I., Aliferis, C. F., Statnikov, A.: Time and Sample Efficient Discovery of Markov Blankets and Direct Causal Relations. Technical report DSL TR-03-04, Vanderbilt University (2003).

[22] Tsamardinos, I., Brown, L. E., Aliferis, C. F.: The Max-Min Hill-Climbing Bayesian Network Structure Learning Algorithm. Machine Learning (2006).

[23] Weston, J., Pérez-Cruz, F., Bousquet, O., Chapelle, O., Elisseeff, A., Schölkopf, B.: Feature Selection and Transduction for Prediction of Molecular Bioactivity for Drug Design. Bioinformatics 19 (2003) 764-771.