Applied Artificial Intelligence
An International Journal

ISSN: 0883-9514 (Print) 1087-6545 (Online) Journal homepage: https://www.tandfonline.com/loi/uaai20

Bayesian Network Learning with the PC Algorithm: An Improved and Correct Variation

Michail Tsagris

To cite this article: Michail Tsagris (2019) Bayesian Network Learning with the PC Algorithm: An Improved and Correct Variation, Applied Artificial Intelligence, 33:2, 101-123, DOI: 10.1080/08839514.2018.1526760

To link to this article: https://doi.org/10.1080/08839514.2018.1526760

APPLIED ARTIFICIAL INTELLIGENCE
2019, VOL. 33, NO. 2, 101–123
https://doi.org/10.1080/08839514.2018.1526760

Taylor & Francis
Taylor & Francis Group

Bayesian Network Learning with the PC Algorithm: An Improved and Correct Variation

Michail Tsagris

Department of Computer Science, University of Crete, Herakleion, Greece

ABSTRACT

PC is a prototypical constraint-based algorithm for learning Bayesian networks, a special case of directed acyclic graphs. An existing variant of it, in the R package *pcalg*, was developed to make the skeleton phase order independent. In return, it has notably increased execution time. In this paper, we clarify that the PC algorithm the skeleton phase of PC is indeed order independent. The modification we propose outperforms *pcalg*'s variant of the PC in terms of returning correct networks of better quality as is less prone to errors and in some cases it is a lot more computationally cheaper. In addition, we show that *pcalg*'s variant does not return valid acyclic graphs.

Introduction

Learning causal relationships in datasets with many variables (or features) is of high importance in many scientific fields. Bayesian networks (BN) have been applied for this purpose in many settings. In clinical set ups for example, BNs can be used for disease diagnostic purposes (Bucci, Sandrucci, and Vicario 2011; Suchánek, Marecki, and Bucki 2014; Zagorecki, Orzechowski, and Holownia 2013). In biology, they can be used to discover interaction networks (Isci et al. 2013) or analyze gene expression data (Friedman et al. 2000). Other applications include psychology (Glymour 2001), teaching purposes (Conati, Gertner, and Vanlehn 2002), data mining (Heckerman 1997), environmental modeling (Aguilera et al. 2011) and criminology (Baumgartner, Ferrari, and Palermo 2008) to name a few. Finance, and insurance are linked with operational risk management (or modeling), which is another area where BNs have been used (Cowell, Verrall, and Yoon 2007).

BNs are probabilistic graphical models representing causal relationships between variables. Network visualization is offered and intuitively (and causally in the case of BNs) one can interpret relationships among variables. Two main classes of algorithms for BN learning are constraint-based and score-based methods. Constraint-based learning algorithms, such as PC (Spirtes and

CONTACT Michail Tsagris mtsagris@yahoo.gr

Color versions of one or more of the figures in the article can be found online at www.tandfonline.com/uaai.

Glymour 1991; Spirtes, Glymour, and Scheines 2000) and FCI (Spirtes, Glymour, and Scheines 2000) employ conditional independence (CI) tests to discover the structure of the network, and then orient the edges by repetitively applying orientation rules. Score-based methods on the other hand (Chickering 2002; Cooper and Herskovits 1992; Heckerman, Geiger, and Chickering 1995) assign a score on the whole network and perform a search in the space of BNs to identify a high-scoring network. Furthermore, hybrid algorithms exist, such as MMHC (Tsamardinos, Brown, and Aliferis 2006) which first performs CI tests and then uses a scoring method on the reduced space.

In this work, we focus on the PC[1] algorithm for BN learning and a modification of it called PC-stable (SPC) (Colombo and Maathuis 2014). SPC is a popular modification of the skeleton phase of the PC, available in the R package *pcalg* (Kalisch et al. 2012), which was suggested as a means of making the skeleton of the original PC order independent. As analyzed later in this work, the original PC is already order independent and hence this variation did not make any improvement and is computationally more expensive. What is more is that it does not always produce a valid partially oriented DAG (PDAG).

The aforementioned observations encouraged us to revisit the original PC algorithm (Spirtes, Glymour, and Scheines 2000). We have re-implemented, in the R package *MXM* (Lagani et al. 2017), the original PC algorithm and modified its orientation rules[2] to ensure the validity of the returned graph and finally attempted to resolve some conflicts which occur.

Extensive simulation studies depict that our modification of the PC algorithm, termed MPC hereafter, leads to better results in comparison to SPC. The comparison takes place using a variety of CI tests for continuous, categorical, and mixed data and the results show that MPC leads not only to better results, but also, unlike SPC, returns a valid PDAG. In terms of computational cost, the implementation of the skeleton phase of the MPC (which is the same as PC) is much cheaper than SPC. For categorical data for example, it is more than two orders of magnitude faster. Time efficiency, validity of the output graph, and less errors in conjunction with its functionalities (works with many types of data) make MPC a more practical than PC-stable, algorithm to use.

Section 2 contains brief information about BNs and CI tests. Section 3 summarizes the two algorithms, along with some algorithmic details and our modifications in the orientation rules. Finally in Section 4, extensive experiments are presented and Section 5 concludes the paper.

Preliminaries

Directed Acyclic Graphs

A graphical model or probabilistic graphical model is a probabilistic model for which a graph expresses the conditional (in)dependencies between

random variables. A directed graph is a graphical model where arrows indicate a direction. When no cycles are allowed, for example $V_1 \rightarrow V_2 \rightarrow V_3$, the graph is termed directed acyclic graph (DAG).

The variables are denoted by nodes or vertices in the graph. The parents of a node V_i are the nodes whose direction (arrows) points toward V_i, and respectively, the node V_i is the child of these nodes. For example in Figure 2 (a), the nodes X, Z and W are the parents of the node Y and the node Y is the child of the nodes X, Z and W.

Bayesian Networks

A BN Pearl (1988); Spirtes, Glymour, and Scheines (2000) $B = \langle G, P \rangle$ consists of a directed acyclic graph G over vertices (variables) $\mathbf{V}$ and a joint probability distribution P. P is linked to G through the Markov condition, which states that each variable is conditionally independent of its non-descendants given its parents. By using this condition, the joint distribution P can be factorized as

$$P(V_1, \ldots, V_n) = \prod_{i=1}^{d} P(V_i | Pa(V_i))$$

where d is the total number of variables in G and Pa(V_i) denotes the parent set of V_i in G. If all conditional independencies in P are entailed by the Markov condition in G, the BN is called faithful.

Causal sufficiency, i.e. no latent confounders between the measured variables in $\mathbf{V}$, is a necessary assumption made by PC. A causal BN is a BN where edges are interpreted causally. Specifically, an edge $X \rightarrow Y$ exists if X is a direct cause of Y in the context of the variables $\mathbf{V}$. For every directed edge, $V_i \rightarrow V_j$, V_i denotes the parent and V_j the child. A collider is a triplet (V_i, V_k, V_j) where $V_i \rightarrow V_k \leftarrow V_j$. If there is no edge between V_i and V_j the node V_k is called unshielded collider. This translates to independence between V_i and V_j condition on V_k if G and P are faithful to each other (Spirtes, Glymour, and Scheines, 2000).

Typically, multiple BNs encode the same set of CIs.[3] Such BNs are called Markov equivalent, and the set of all Markov equivalent BNs forms a Markov equivalence class. This class can be represented by a complete partially directed acyclic graph (CPDAG), which in addition to directed edges also contains undirected edges. Undirected edges may be oriented either way in some BNs in the Markov equivalence class (although not all combinations are possible), while directed and missing edges are shared among all equivalent networks.

Constrained-based Algorithms for Learning the Structure of a BN

Constrained-based algorithms (e.g. PC and MMHC) infer a BN by applying CI tests between pairs of variables. These tests can be information based (BIC), ad-hoc (Bayes factor) or statistical, i.e. they produce a p-value in order to make a decision to remove or not an edge between two nodes. In this paper, we focus on the PC algorithm and use statistical CI tests.

Conditional Independence Tests

Let X and Y be two random variables, and $\mathbf{Z}$ be a (possibly empty) set of random variables. Statistically speaking, X and Y are conditionally independent given $\mathbf{Z}$ ($X \perp\!\!\!\perp Y|Z$), if $P(X, Y|\mathbf{Z}) = P(X|\mathbf{Z}) \cdot P(Y|\mathbf{Z})$ holds for all values of X, Y and $\mathbf{Z}$. Equivalently, CI of X and Y given $\mathbf{Z}$ implies $P(X|Y, \mathbf{Z}) = P(X|\mathbf{Z})$ and $P(Y|X, \mathbf{Z}) = P(Y|\mathbf{Z})$. Such statements can be tested using CI tests.

Pearson Correlation

An example of commonly employed CI test is the partial correlation test Baba, Shibata, and Sibuya (2004) for continuous variables assuming linear relationships among the variables. The test statistic for the partial Pearson correlation is given by

$$T_{pearson} = \frac{1}{2}\left|\log\frac{1 + r_{X,Y|\mathbf{z}}}{1 - r_{X,Y|\mathbf{Z}}}\right|\sqrt{n - |\mathbf{Z}| - 3}, \tag{1}$$

where n is the sample size, $|\mathbf{Z}|$ the number of conditioning variables in the set $\mathbf{Z}$ and $r_{X,Y|z}$ is tha partial Pearson correlation of X and Y conditioning on $\mathbf{Z}$.

Spearman Correlation

In the case of Spearman correlation, the test statistic (1) becomes

$$T_{spearman} = \frac{1}{2}\left|\log\frac{1 + r_{X,Y|\mathbf{z}}}{1 - r_{X,Y|\mathbf{Z}}}\right|\frac{\sqrt{n - |\mathbf{Z}| - 3}}{1.029563}.$$

Asymptotically, both test statistics follow the standard normal distribution under the zero correlation assumption. In the R package *MXM* though, they are calibrated against a t distribution with $n - |\mathbf{Z}| - 3$ degrees of freedom, whose performance is better for small sample sizes.

Robust Pearson Correlation

In Shevlyakov and Smirnov (2011) many ways of obtaining robust correlations were presented. We have chosen one way to robustify (1). We calculate the residuals of 2 MM regression models (Yohai 1987), one for each variable X or Y being the response variable

$$\mathbf{e}_1 = Y - (\alpha_1 + \beta_1 X + \mathbf{Z}b_1) \text{ and } \mathbf{e}_2 = X - (\alpha_1 + \beta_1 Y + \mathbf{Z}b_2)$$

The conditional Pearson correlation is given via the correlation of these residuals. P-values are obtained as before using the Pearson's test statistic (1).

G^2 Test of Independence

The G^2 (and the χ^2) test of independence Agresti (2002)

$$G^2 = 2\sum_k \sum_{i,j} O_{ij|k} \log \frac{O_{ij|k}}{E_{ij|k}}$$

is used for categorical variables. The $O_{ij|k}$ are the observed frequencies of the $i-th$ and $j-th$ values of X and Y respectively in the $k-th$ set of values of $\mathbf{Z}$ and the $E_{ij|k}$ are their corresponding expected frequencies. Under the CI assumption, the G^2 test statistic follows the χ^2 distribution with $(|X|-1)(|X|-1)(|\mathbf{Z}|-1)$ degrees of freedom.

Permutation Based P-Values

All four aforementioned test statistics produce asymptotic p-values. A second method of obtaining p-values for either test statistic is via permutations. In the continuous variables for example, the idea is to distort the pairs multiple times and each time calculate the relevant test statistic (based on Pearson or Spearman). The p-value is then computed as the proportion of times the values of the permuted test statistics exceed the value of the test statistic in the original data. In the categorical variables, the scheme is more complicated and care must be taken. In the R package *MXM*, the choice of this scheme is offered by selecting the number of permutations to be performed.

Distance Correlation

For continuous variables, the relationships may not be linear. For this reason, we have also included the (partial) distance correlation (Szekely and Rizzo et al. 2014; Székely, Rizzo, and Bakirov 2007)

$$dcor(X,Y) = \frac{dcov(X,Y)}{\sqrt{dvar(Y)}, \sqrt{dvar(Y)}},$$

where $dcov(X,Y)$ is the distance covariance between X and Y and $dvar(.)$ denotes the distance variance. The p-value for zero correlation is calculated via permutations.

Symmetric Conditional Independence Tests for Mixed Data

In the case of mixed data, e.g. continuous, binary, ordinal, we used the symmetric test suggested by Tsagris et al. (2017). In order to check whether $(X \perp\!\!\!\perp Y|\mathbf{Z})$ holds true, one should perform two likelihood ratio tests, for

$Y \sim f(\mathbf{Z}, X)$ assessing $(Y \perp\!\!\!\perp X|\mathbf{Z})$ and for $X \sim f(\mathbf{Z}, Y)$ assessing $(X \perp\!\!\!\perp Y|\mathbf{Z})$. Regression models however are not symmetric. Y can be binary for example requiring a logistic regression and X can be Gaussian, requiring a linear model. The two resulting p-values are then combined in a meta-analytic way which handles the inherited correlation between the two tests.

A faster alternative method is to perform one regression model only, which was shown to be asymptotically equivalent to performing both models (Tsagris et al. 2017). In that case, (Tsagris et al. 2017) proposed a priority rule for implementing a regression model. For example, when the pairs consist of a continuous and a nominal, ordinal, etc., variable, the faster model that should be applied is the linear one. In case of a nominal-ordinal pair, the multinomial logistic regression model should be fitted as is faster than the ordinal logistic regression model.

The family of symmetric CI tests includes repeated measurements and clustered data (data from family members for example). These types of data are handled by generalized linear mixed models or by generalized estimating equations (Demidenko 2013). This means, that our implementation of PC constructs PDAGs for many types of data.

PC, SPC and MPC Algorithms

The skeleton phase of both the PC and SPC algorithms begins with all pairwise unconditional associations and removes the edge between pairs, which are not statistically significantly related. Subsequently, CI tests are performed with the cardinality of the conditioning set (denoted by k) increasing by one at a time. At every step, the conditioning set consists of subsets of the neighbors of each variable. This process is continued until no edge can be removed.

A main difference between the PC and SPC algorithm is that in the SPC, the edge between any two variables for which the association is found to be statistically not significant, is not removed. For the specific value of k, all tests are performed and non-significant edges are deleted when increasing the value of k. Secondly, PC examines the pairs in an ordered fashion, whereas SPC goes through all variables that have at least k neighbors; thus, it performs more tests. The pseudocode of the skeleton phases of the PC and the SPC algorithm is given in Algorithms 1and 2, respectively.

We emphasize that the modification in the skeleton phase of the SPC algorithm is the same as the one proposed by Abellán, Gómez-Olmedo, and Moral et al. (2006) and Cano, Gómez-Olmedo, and Moral (2008). Cano, Gómez-Olmedo, and Moral (2008) mentions that the order of the tests performed in the original PC algorithm, even at the same cardinality step, are not order independent. According to Colombo and Maathuis (2014), the

order independence problem in constructing the skeleton of the BN is resolved by using the SPC.

There has been a misunderstanding for many years now. The skeleton phase of the original PC is independent of the order at which the variables are present in the dataset and this is the first time this clarification is given. (Spirtes, Glymour, and Scheines 2000, pg. 90) mention three heuristics to speed up the algorithm and perform fewer CI tests. The first heuristic is the so called lexicographic order; do all tests in the order of the variables. A reordering of the columns (variables) in the dataset will lead to a change in the order of the CI tests are performed. This was the motivation behind Colombo and Maathuis (2014) who developed an order independent skeleton phase of the PC algorithm. Based on this, they proposed their modification which makes the skeleton of the PC order independent, but at the cost of performing twice as many tests as the PC does with the third heuristic. The second heuristic accounts for this order dependence and performs the CI tests on the pairs of variables that are least dependent. The conditioning subsets (**S**) are chosen in a lexicographic order though. It is evident, that the order of the variable still affects the outcome.

Finally, the third heuristic is to perform the CI tests on pairs of variables that are least dependent conditional on those subsets that are most dependent on either variable of the pair.[4] It becomes evident that the order of appearance of the variables in the data does not matter when one uses the third heuristic. The sequence of the tests is based on the strength of association of the pairs of variables and the order of appearance makes no difference. No experiment is necessary to show that the first two heuristics are order dependent, whereas the third one is order-independent. Hence, we can state that **the skeleton phase of the (original) PC algorithm is order-independent**.

Algorithm 1 Skeleton phase of the PC algorithm

1: **Input**: Dataset on a set of n variables **V**

2: Let $k = 0$

3: **Repeat**

4: **Repeat**

5: Select an ordered pair of variables V_i and V_j that are adjacent in G, such that $\left|adj(G, V_i)\backslash\{V_j\}\right| \geq k$, and a subset **S** with $\left|adj(G, V_i)\backslash\{V_j\}\right| = k$, and if $(V_i \perp\!\!\!\perp V_j | \mathbf{S})$ delete edge $V_i - V_j$ from G and record $sepset(V_i, V_j) = sepset(V_j, V_i) = \mathbf{S}$.

6: Until all ordered pairs of adjacent variables V_i and V_j with $\left|adj(G, V_i)\backslash\{V_j\}\right| \geq k$ and all subsets **S** with $\left|adj(G, V_i)\backslash\{V_j\}\right| = k$ have been tested for CI.

7: k = k + 1

8: **Until** for each ordered pair of adjacent variables V_i and V_j, $\left|adj(G, V_i)\backslash\{V_j\}\right| \leq k$.

9: **Return** G, sepset.

Algorithm 2 Skeleton phase of the SPC algorithm

1: **Input**: Dataset on a set of n variables **V**

2: Let $k = 0$

3: **Repeat**

4: **For all** variables V_i in G **do**

5: Let $a(V_i) = adj(G, V_i)$

6: **End for**

7: **Repeat**

8: Select an ordered pair of variables V_i and V_j that are adjacent in G, such that $\left|a(G, V_i)\backslash\{V_j\}\right| \geq k$, and a subset **S** with $\left|a(G, V_i)\backslash\{V_j\}\right| = k$, and if $(V_i \perp\perp V_j|\mathbf{S})$ delete edge $V_i - V_j$ from G and record $sepset(V_i, V_j) = sepset(V_j, V_i) = \mathbf{S}$.

9: **Until** all ordered pairs of adjacent variables V_i and V_j with $\left|a(G, V_i)\backslash\{V_j\}\right| \geq k$ and all subsets **S** with $\left|a(G, V_i)\backslash\{V_j\}\right| = k$ have been tested for CI.

10: k = k + 1

11: **Until** for each ordered pair of adjacent variables V_i and V_j, $\left|a(G, V_i)\backslash\{V_j\}\right| \leq k$.

12: **Return** G, sepset.

Implementation Details of the PC Algorithm

There are a few implementation details that can make a difference in the number of tests performed, the order of pairs and in the quality of the constructed network.

(1) The first and most important feature is that we have utilized the often neglected third heuristic in our implementation in the R package *MXM*.

(2) This heuristic relies on the correct ordering of the pairs of variables. When the p-values are very small below a threshold value, 10^{-16} for example, R rounds them to 0. When it comes to ordering 2 or more p-values, R R Core Team (2016) orders 2 or more 0 values at random. This case is not at all rare, especially with the G^2 test. This problem is

also common in feature selection algorithms, which use an ordering of the p-values in order to select the most statistically significant variable. The answer to this problem is the use of the logarithm of the p-values. This of course is not the case for the SPC since it does not rely on any p-value ordering heuristic.

(3) When it comes to permutation based CI tests, equal (logged) p-value is a very frequent phenomenon. For this reason, we order the p-values first and then, for the equal p-values, we order their test statistic value divided by the degrees of freedom of the test. For the partial correlation test, this division is not different from taking the test statistic, as at each step of the algorithm the number of conditioning variables is the same. For the G^2 test though this makes a difference, as the degrees of freedom are usually different.

(4) The CI tests return a p-value and using a significance level α ((Colombo and Maathuis (2014) suggest values of 0.01 or less), a decision on the edge removal is made. In all of our experiments, we have chosen $\alpha = 0.01$. The α (type I error) denotes the probability of falsely assuming that two variables are not independent when in fact they are. In general, there is a trade-off between the type I error and the type II error (not detecting dependence when in fact there is one, termed β). But, with large samples, the power of the CI test (1 - β) is high. Thus, it is advisable to have a low significance level. This way, false positive added edges remain bounded at the 1%[5] of the total number of edges, in other words, statistical errors can be kept at very low levels if a large sample size is available.

Orientation Rules of the PC Algorithm

The orientation phase of the PC is to apply the following four rules as dictated by Spirtes, Glymour, and Scheines (2000) and Colombo and Maathuis (2014).

Rule 0. For every triplet of variables (V_i, V_j, V_k) such that V_i and V_j and V_j and V_k are adjacent in G, but V_i and V_k are not, orient $V_i - V_j - V_k$ as $V_i \rightarrow V_j \leftarrow V_k$ if $V_j \notin sepset(V_i, V_k)$.

Rule 1 . Orient $V_j - V_k$ as $V_j \rightarrow V_k$ if there is a directed edge $V_i \rightarrow V_j$ such that V_i and V_k are not adjacent in G.

Rule 2. Orient $V_i - V_k$ as $V_i \rightarrow V_k$ is there is a directed path $V_i \rightarrow V_j \rightarrow V_k$.

Rule 3. Orient $V_i - V_j$ as $V_i \rightarrow V_j$ whenever there are two directed paths $V_i - V_k \rightarrow V_j$ and $V_i - V_l \rightarrow V_j$, such that V_k and V_l are not adjacent in G.

Rule 0 is to be applied first. Then, according to Spirtes, Glymour, and Scheines (2000) the order of the other three rules is independent, as in the

sample limit (sample size going to infinity) and under no statistical errors, the output will be a Markov equivalence DAG. In the finite sample size case, though statistical errors exist and can lead to the wrong skeleton. Application of the third heuristic requires some attention and conflicts within the rules almost always appear. Cycles prevention is another issue not heavily addressed in SPC. In the next sub-section, we try to address some of these issues and point out some algorithmic and CI tests related details.

Modification of the PC Rules

The modification of the orientation rules that gives rise to the MPC algorithm. More formally, MPC consists of the PC skeleton phase and the modifications in the rules, presented below.

(1) 1. As mentioned in the Preliminaries Section, a DAG and hence a BN does not allow cycles. Unfortunately, the four aforementioned rules do not include cycles prevention in their protocol. When it comes to applying one rule, we check if cycles are created. If the answer is yes, the rule is canceled and the edge is left un-oriented. None of the aforementioned variations of the PC Abellán, Gómez-Olmedo, and Moral et al. (2006); Cano, Gómez-Olmedo, and Moral (2008); Colombo and Maathuis (2014) addresses this issue. Figure 1 shows an

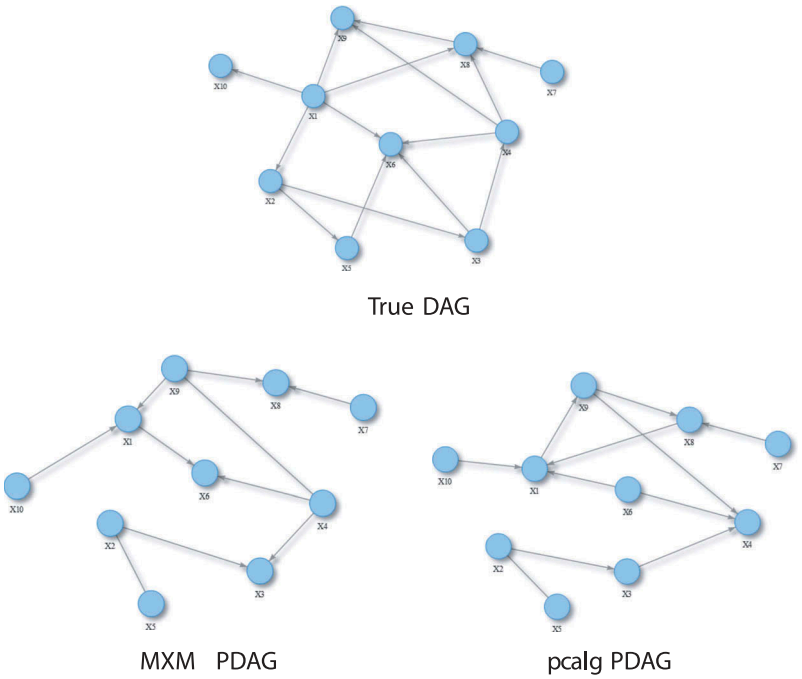

Figure 1. An example where *pcalg* produced a partially oriented **cyclic** directed graph.

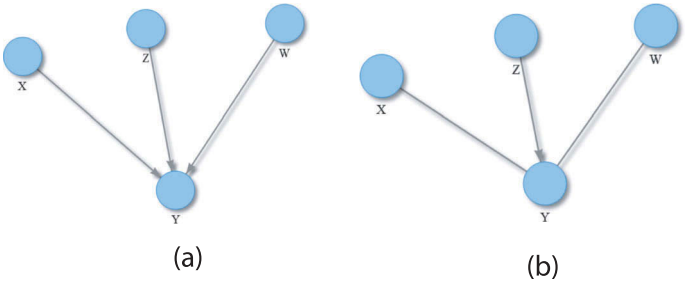

Figure 2. (a) Y is a collider for X and Z, for W and Z, but falsely considered as collider for X and W. (b) After discovering the mistake, the edges X → Y and W → Y loose their arrow.

example where SPC produces a cycle,[6] namely $X_1 \rightarrow X_9 \rightarrow X_8 \rightarrow X_1$. The difference in the skeleton between MPC and SPC is the edge connecting the nodes X_1 and X_8.

(2) 2. Unfaithful colliders often emerge. For example, let pairs X - Y, W - Y and Z - Y. The first triplet X → Y ← Z holds true ($X \perp\!\!\!\perp Z|Y$), the second one Z → Y ← W ($X \perp\!\!\!\perp W|Y$) holds true as well, but at the same time, the triplet X → Y ← W that has been created does not hold true ($X \perp\!\!\!\perp W|Y$). In this case, similarly to Isozaki (2014) the MPC disorients X - Y and W - Y (see Figure 2).

(3) 3. Colliders can be created when applying Rule 1. If that is the case, the rule is canceled for that particular node. Figure 3 gives such an example. In (a), Rule 1 would turn Y - Z into Y → Z and W - Z into W → Z. This would create a collider, Z for Y and W. We will direct one edge only (the first seen). Suppose we turned Y - Z into Y → Z. Then, Z - W is not allowed to be turned into Z → W because W would become a collider. In this case, Z - W would remain as is. In (b) orienting $Y - Z$ and $W - Z$ would create a new collider as well. In this case, both edges will be left un-oriented.

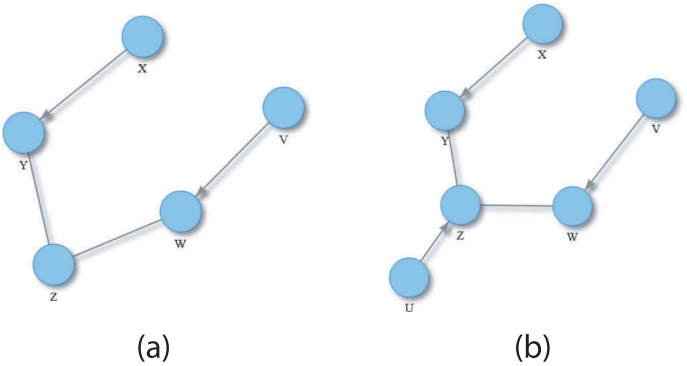

Figure 3. Rule 1 would turn Y - Z into Y → Z and W - Z into W → Z in both cases.

The four orientation rules do not include many conflict resolution strategies. Similarly, to the SPC, we treated the triplets in a lexicographical order. The difference is that we do not overwrite the directions, but perform them in a first-come-first-serve fashion. We should note that conflict-resolution strategies still remain an open area of research.

To sum up, SPC relies on a modified skeleton phase of the original PC. MPC on the other hand has left the skeleton phase of the original PC the same and changed the application of the orientation rules toward two directions: a) preventing cycles and b) preventing the creation of non existing colliders.

Experimental Validation and Comparisons

We conducted extensive experiments on simulated data in order to investigate the quality of estimation of the MPC and SPC. Both algorithms were mainly compared on synthetic BNs with either continuous, categorical or mixed data.

Data Generation

Let X be a variable in G and $Pa(X)$ be the parents of X in G.

In case $Pa(X)$ is empty, X is sampled from the standard normal distribution. If $Pa(X)$ is not empty, then $X = f(Pa(X)) = \beta_0 + \sum_i \beta_i Pa_i(X) + \epsilon_X$, where $f(Pa(X))$ is a linear function depending on X in this case, but in general it can by any function. The following procedure is used to generate data for X.

(1) Generate samples for each variable in $Pa(X)$ recursively, until samples for each variable are available.
(2) Sample the coefficients β of $f(Pa(X))$ uniformly at random from $[-1, -0.1] \cup [0.1, 1]$.
(3) Generate $\epsilon_X \sim N(0, 1)$.
(4) Compute X using $f(Pa(X))$.

In order to generate ordinal variables (for the mixed data case scenario), we first generated a continuous variable as previously described, and then discretized it into 2–4 categories appropriately (retaining the ordinal scale). Each category contains at least 15% of the observations, while the remaining ones are randomly allocated to all categories. This is identical to having a latent continuous variable (the one generated), but observing its discretized proxy variable with some noise added. Note that, as the discretization is random, any normality of the input continuous variable is not preserved. Finally, ordinal variables in the parent sets

are not treated as nominal variables, but simply as continuous ones and thus only one coefficient is used for them for the purpose of data generation.

After completing the dataset set generation we shuffle the columns of the matrix (change the order of the variables) and hence the same columns and rows of the adjacency matrix representing the BN. This makes the estimation procedure more difficult, because the order in which Rule 0 is applied is lexicographical and the order with which we have generated the data would benefit from the application of this rule.

Many evaluation criteria were employed in order to accurately evaluate the performance of the two algorithms. For the skeleton phase, we compared the computational efficiency and the number of tests performed by both algorithms, along with the Hamming distance (HD). The HD between two strings of equal length is the number of positions at which the corresponding symbols are different. In our case, the two strings (one for the estimated skeleton and one for the true skeleton of the BN) are binary, indicating the presence or absence of an edge between two pairs.

The quality of the learned BNs was assessed using the structural Hamming distance (SHD) Tsamardinos, Brown, and Aliferis (2006) of the estimated PDAG from the true PDAG. This is defined as the number of operations required to make the estimated graph equal to the true graph. The true PDAG is simply the Markov equivalence graph of the true BN; that is some edges have been un-oriented as their direction cannot be statistically decided. The transform from the DAG to the PDAG is carried out using Chickering's algorithm Chickering (1995). The number of times the estimated network is a valid PDAG (no cycles are present) is another crucial measure reported.

Hamming Distance of the Skeleton

Figure 4 clearly demonstrates that the HD of the MPC skeleton is similar to the HD of SPC for all combinations of sample size and number of variables.

Computational Time and Number of Tests Performed during the Skeleton Phase

We evaluated the MPC and SPC algorithm in terms of computational time and number of CI tests performed. For the continuous data case, the generated BNs contained a various number of nodes, $p = (50, 100, 150, 200, 300, 500, 700, 1000)$, with 3 and 5 neighbors on average. For each case, we created 30 random BNs, and simulated Gaussian data with various sample sizes, $n = (100, 200, 500, 1000, 2000, 5000)$. In total, this amounts to 2880 datasets.

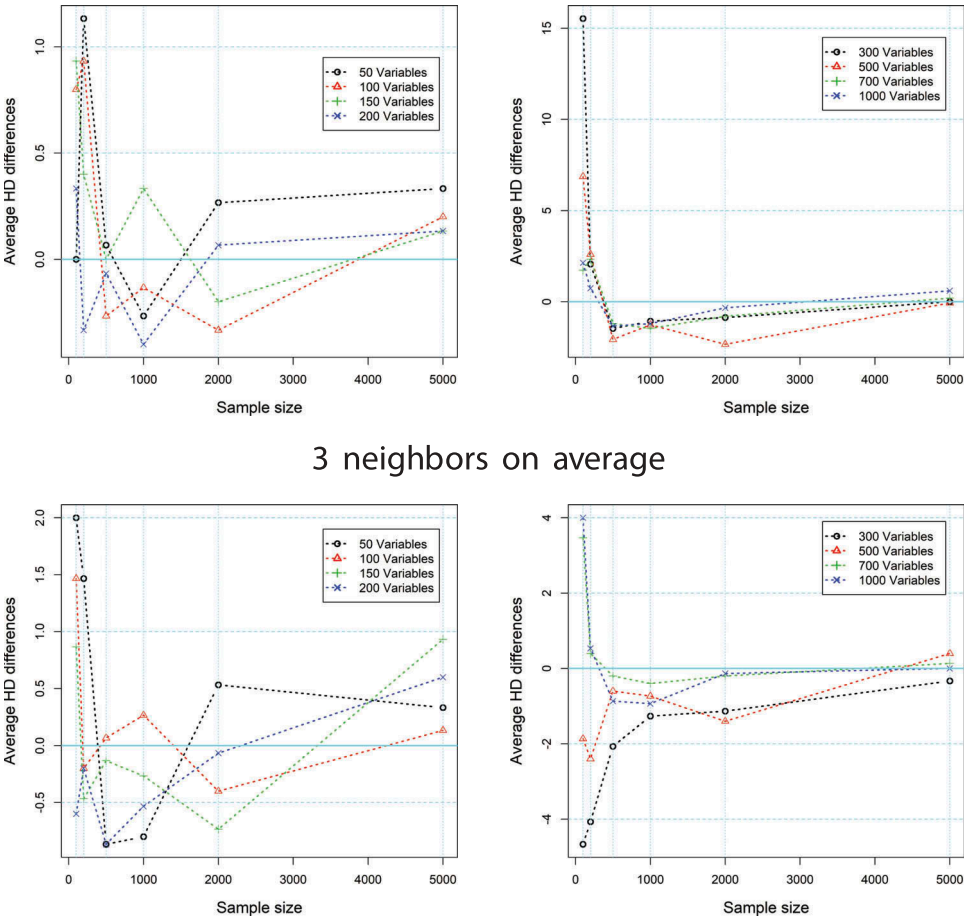

Figure 4. Differences in the average HDs between MPC (*MXM* R package) and SPC (*pcalg* R package) are presented for a range of sample sizes.

As for the categorical data, we simulated datasets with different sample size from the INSURANCE network Binder et al. (1997) which contains 27 variables only (and 52 edges). The SPC algorithm for continuous data has been implemented in C++, whereas for categorical data has been implemented in R. On the contrary, our PC algorithm implementation, for both continuous and categorical data, is in C++. Henceforth, the time comparisons with categorical data are not fair and this is why we chose to simulate from a BN with few variables and vary the sample size.

SPC performs more tests, thus reporting only the time would not result in a fair comparison. For this reason, for each algorithm we report the total time required divided by the number of tests carried out. Figure 5 presents the ratios of the normalized times required by MPC and SPC with both

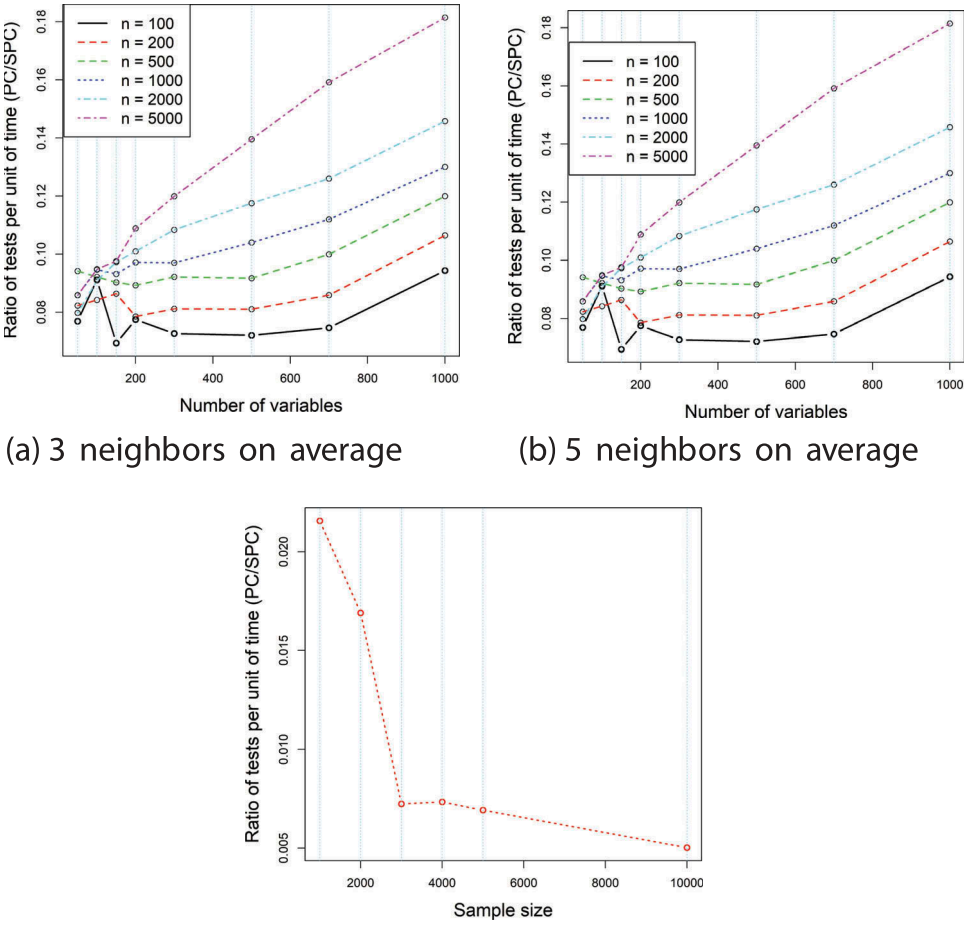

Figure 5. For each algorithm, MPC and SPC, we have calculated the normalized computational cost required to construct the skeleton. That is the total time divided by the number of tests executed. The ratio of the normalized times between MPC and SPC appears in these two graphs for BNs with continuous data and (a) 3 and (b) 5 neighbors on average. The bottom graph contains the same information with categorical data obtained from a real BN.

continuous and categorical data. Overall, we can see that the skeleton of the MPC is much more computationally efficient than SPC.

Structural Hamming Distance of the PDAGs with Continuous Data

The generated BNs contained a various number of nodes, $p = (50, 100, 150, 200, 300, 500, 700, 1000)$, with 3 and 5 neighbors on average. For each case we created 30 random BNs, and simulated Gaussian data with various sample sizes, $n = (100, 200, 500, 1000, 2000, 5000)$. Figure 6 clearly

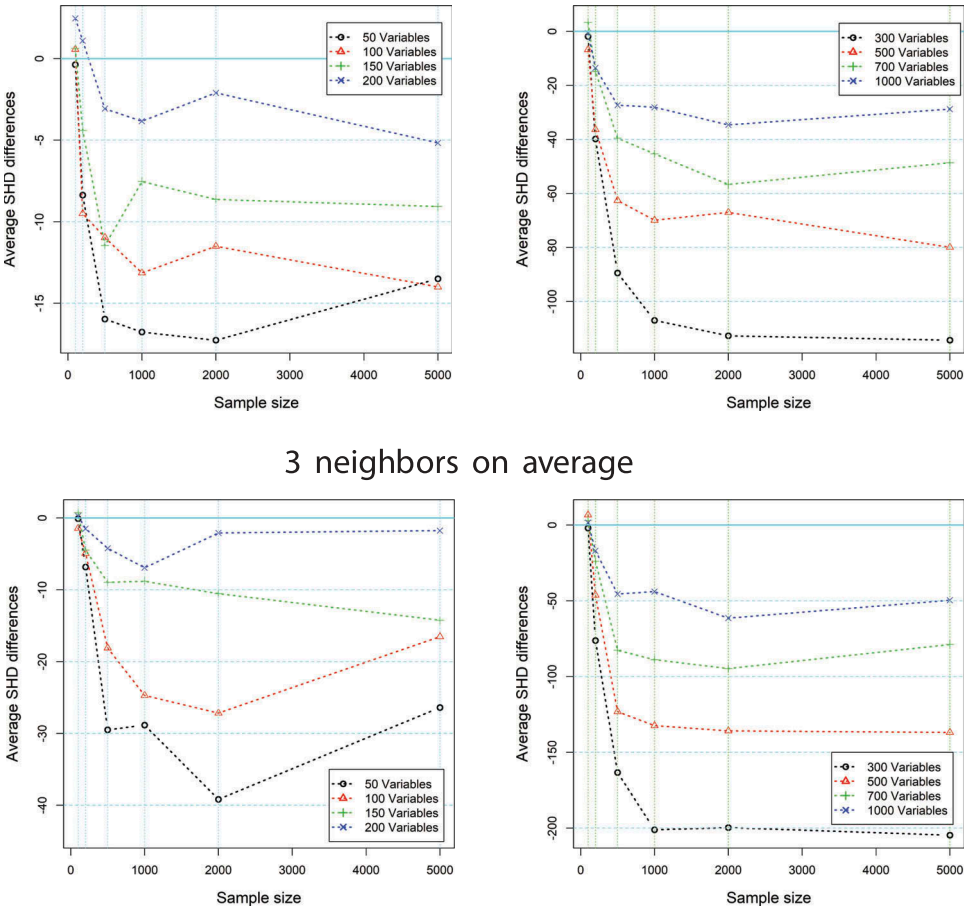

Figure 6. Differences in the average SHDs between MPC (SHD(MPC) - SHD(SPC)) are presented for a range of sample sizes. Negative values are in favor of MPC.

demonstrates that almost always the SHD of the MPC is lower than the SHD of SPC for all combinations of sample size and number of variables.

Structural Hamming Distance of the PDAGs with Mixed Data

The simulated data now consist of continuous, binary and ordinal data with the analogy being, on average, $50\%, 25\%, 25\%$ respectively. In our experiments, we used the same number of variables as Tsagris et al. (2017), $p = (50, 100)$ but larger sample sizes, $n = (100, 200, 500, 1000, 2000)$. The average number of neighbors is the same as before (3 and 5). A similar, to the continuous data case, conclusion is drawn here as well. MPC almost always produces PDAGs with SHD lower than SPC (see Figure 7).

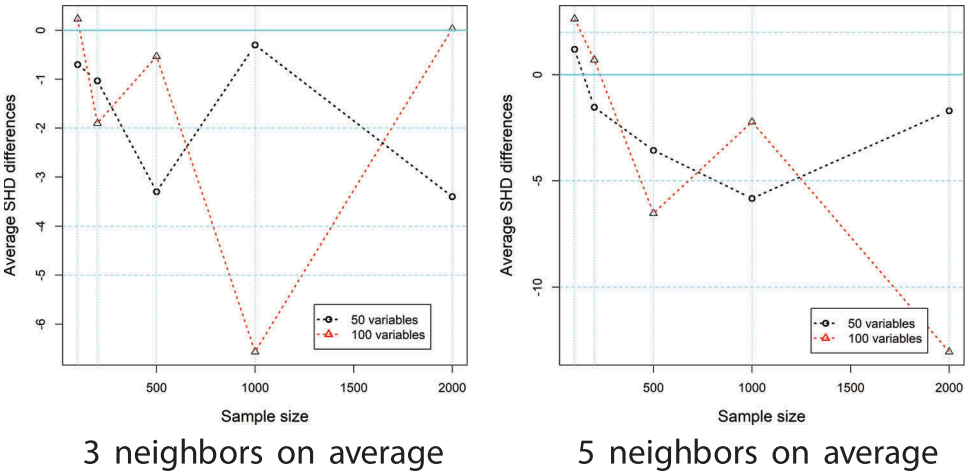

Figure 7. Mixed data scenario. The average SHD differences between MPC and SPC (SHD(MPC) - SHD(SPC)) are presented for a range of sample sizes. Negative values are in favor of MPC.

Proportion of Times SPC Returns an Acyclic Graph

SPC does not check for cycles during performance of the orientation rules. BNs are acyclic graphs, hence if this condition is not satisfied, the final graph is not a BN. Figure 8 depicts the issue which becomes more evident as the sample size increases. Even with 1000 variables, SPC will return an acyclic graph with a high probability when the sample size is small. But, as the sample size increases, even with 50 variables, this probability decays.

Realistic Bayesian Networks

As a final comparison, we used the HAILFINDER (Jensen and Jensen 1996) and the ALARM (Beinlich et al. 1989) BNs, which consists of 56 nodes & 66 edges and 37 nodes & 46 edges respectively. The R package *bnlearn* Scutari (2010) contains 20,000 categorical instantiations from these networks. Even though these datasets consist of both nominal and ordinal variables but we treated them as nominal.

We randomly permuted the variables of the data 30 times and each time we applied the four algorithms and calculated the SHD and the implementation time. Table 1 presents this information. The SHD produced by MMHC varies a lot, whereas the PC related algorithms exhibit very small variances.

For the HAILFINDER network, on average all PC-related algorithms have an SHD equal to 38–39. However, the SPC produces an acyclic graph only in 50% of the times. In addition, SPC is more than 1200 times slower than MPC. For the ALARM network, the image is similar. SPC produces a slightly less SHD on average, but it returns as acyclic graph in 73.33% of the times.

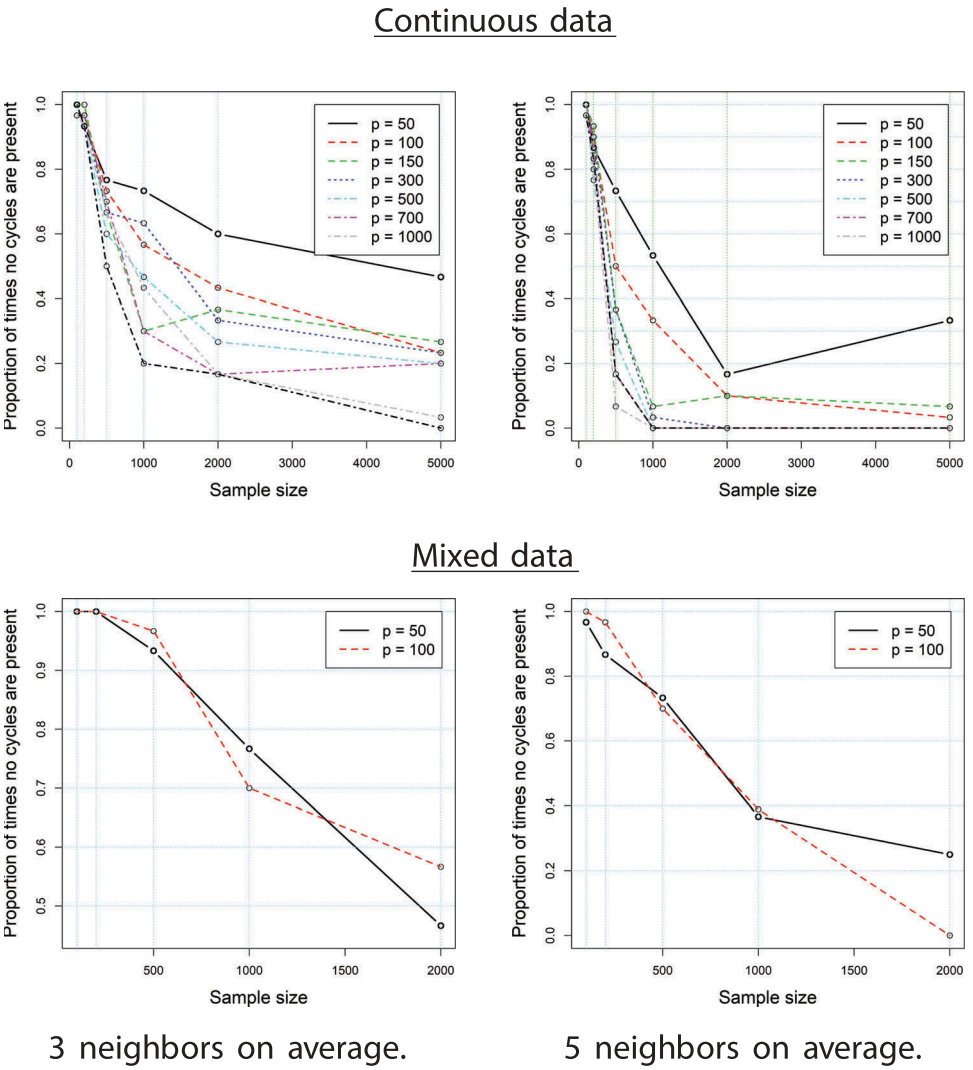

Figure 8. Percentage of times there were no cycles in the estimated PDAG when using the SPC. The lines correspond to different number of variables as the sample size increases. MPC does not appear because it always returns acyclic graphs.

We remind the reader that in Figure 7, the percentage of times SPC produces a DAG decays as the number of variables increases and the minimum number of variables we consider is 50.

Conclusions

In this paper, we showed that the skeleton phase of the original PC algorithm is indeed order independent. Extensive simulation studies showed that the returned skeleton of the MPC (which is the same as the skeleton phase of the

Table 1. Summary statistics regarding SHD and computational cost (in seconds) of the 2 algorithms applied to 30 random permutations of the variables of the data generated from the HAILFINDER and ALARM networks.

	SPC	MPC
SHD: (Minimum, Maximum)		
HAILFINDER	(38, 39)	(38, 40)
ALARM	(57, 57)	(60, 60)
Computational cost: (Minimum, Maximum)		
HAILFINDER	(206.42, 431.67)	(0.20, 0.27)
ALARM	(188.02, 204.84)	(0.18, 0.22)

original PC) and of the SPC have very small differences, which vanish as the sample size increases, yet the former performs half the test the latter performs and is computationally more efficient. When proper modifications are applied on the orientation rules a valid PDAG (no cycles) must be returned. This essential acyclicity property is not checked by the package *pcalg* Kalisch et al. (2012), even though this package is quite popular and has been used by other researchers (Harris and Drton 2013)

The comparisons between MPC and SPC (Colombo and Maathuis 2014) showed that, with continuous data, the first leads to PDAGs whose SHD is lower when dealing with continuous data. The same was with mixed data, continuous, binary and ordinal, but due to the increased computational cost required we did not try high dimensional settings. Despite the difference in the SHD being smaller, SPC produces partially oriented graphs, which are not acyclic; hence, they violate a basic and necessary condition of BNs. The BNs examined here contained continuous, categorical and ordinal data for which Pearson correlation, G^2-test and appropriate regression models respectively were used.

MXM also provides many functionalities to assess the skeleton of the BN. Confidence on the discovered edges can be calculated either theoretically (Triantafillou, Tsamardinos, and Roumpelaki 2014) or numerically via bootstrap and a lower limit in the confidence, as proposed by Scutari and Nagarajan (2013). Estimation of the false discovery rate (Tsamardinos and Brown 2008) and construction of ROC curves are some of the utility functions.

Our main future research direction is to focus on conflict resolution strategies. This is a key aspect of the MPC rules, which can lead to further improvements in the estimated PDAG. More types of data will be handled, making MPC practical and generic. In addition, we plan to examine further the coupling of the MMPC Tsamardinos, Brown, and Aliferis (2006) with the MPC rules. Another direction is to substitute the scoring search with rules that are order independent and produce PDAGs of similar or better quality than MMHC (Tsamardinos, Brown, and Aliferis 2006).

Notes

1. PC stands for Peter and Clark, named after Peter Spirtes and Clark Glymour, the names of the two researchers who invented it.
2. We will drop the word "orientation" hereafter and simply refer to them as rules or PC rules.
3. Two DAGs are called Markov equivalent if and only if they have the same skeletons and the same v-structures Verma and Pearl (1990).
4. In our implementation of the PC we have used this heuristic.
5. This is the worst case scenario upper bound which is never reached. The construction of the skeleton using the PC algorithm ensures this.
6. We highlight that SPC does not always produce acyclic graphs and we show this in the experimentation Section. This was a rare example, yet not impossible to happen. The R code used to generate these Figures is in the Appendix.

Acknowledgments

I would like to acknowledge Professor Tsamardinos Ioannis for our fruitful conversations which inspired me for this paper. Also, Stefanos Fafalios for his constructive comments and Konstantinos Tsirlis for reading an earlier draft.

The research leading to these results has received funding from the European Research Council under the European Union's Seventh Framework Programme (FP/2007-2013)/ERC Grant Agreement n. 617393.

Funding

The research leading to these results has received funding from the European Research Council under the European Union's Seventh Framework Programme (FP/2007-2013) / ERC Grant Agreement No. 617393.

ORCID

Michail Tsagris http://orcid.org/0000-0002-2049-3063

References

Abellán, J., M. Gómez-Olmedo, S. Moral. 2006. Some variations on the PC algorithm. In Third European Workshop on Probabilistic Graphcal Models, 12-15 September 2006, Prague, Czech Republic.

Agresti, A. 2002. Categorical data analysis. In *Wiley series in probability and statis- tics*, 2nd, New Jersey: Wiley-Interscience.

Aguilera, P., A. Fernández, R. Fernández, R. Rumí, and A. Salmerón. 2011. Bayesian networks in environmental modelling. *Environmental Modelling & Software* 26 (12):1376–88. doi:10.1016/j.envsoft.2011.06.004.

Baba, K., R. Shibata, and M. Sibuya. 2004. Partial correlation and conditional correlation as measures of conditional independence. *Australian & New Zealand Journal of Statistics* 46 (4):657–64. doi:10.1111/j.1467-842X.2004.00360.x.

Baumgartner, K., S. Ferrari, and G. Palermo. 2008. Constructing Bayesian networks for criminal profiling from limited data. *Knowledge-Based Systems* 21 (7):563–72. doi:10.1016/j.knosys.2008.03.019.

Beinlich, I. A., H. J. Suermondt, R. M. Chavez, and G. F. Cooper. 1989. The alarm monitoring system: A case study with two probabilistic inference techniques for belief networks. In: Hunter J., Cookson J., Wyatt J. (eds) AIME 89. Lecture Notes in Medical Informatics, vol 38. Springer, Berlin, Heidelberg.

Binder, J., D. Koller, S. Russell, and K. Kanazawa. 1997. Adaptive probabilistic networks with hidden variables. *Machine Learning* 29 (2):213–44. doi:10.1023/A:1007421730016.

Bucci, G., V. Sandrucci, and E. Vicario (2011). Ontologies and Bayesian networks in medical diagnosis. In *System Sciences (HICSS), 2011 44th Hawaii International Conference on*, pp. 1–8. Kauai, Hawaii, USA: IEEE.

Cano, A., M. Gómez-Olmedo, and S. Moral (2008). A score based ranking of the edges for the PC algorithm. In *Proceedings of the Fourth European Workshop on Probabilistic Graphical Models*, Hirtshals, Denmark, pp. 41–48.

Chickering, D. M. (1995). A transformational characterization of equivalent bayesian network structures. In *Proceedings of the Eleventh conference on Uncertainty in artificial intelligence*, Montreal, Canada, pp. 87–98. Morgan Kaufmann Publishers Inc.

Chickering, D. M. 2002. Optimal structure identification with greedy search. *Journal of Machine Learning Research* 3 (Nov):507–54.

Colombo, D., and M. H. Maathuis. 2014. Order-independent constraint-based causal structure learning. *The Journal of Machine Learning Research* 15 (1):3741–82.

Conati, C., A. Gertner, and K. Vanlehn. 2002. Using Bayesian networks to manage uncertainty in student modeling. *User Modeling and User-Adapted Interaction* 12 (4):371–417. doi:10.1023/A:1021258506583.

Cooper, G. F., and E. Herskovits. 1992. A Bayesian method for the induction of probabilistic networks from data. *Machine Learning* 9 (4):309–47. doi:10.1007/BF00994110.

Cowell, R. G., R. J. Verrall, and Y. Yoon. 2007. Modeling operational risk with Bayesian networks. *Journal of Risk and Insurance* 74 (4):795–827. doi:10.1111/j.1539-6975.2007.00235.x.

Demidenko, E. 2013. *Mixed models: Theory and applications with R*. New Jersey: John Wiley & Sons.

Friedman, N., M. Linial, I. Nachman, and D. Pe'er. 2000. Using Bayesian networks to analyze expression data. *Journal of Computational Biology* 7 (3–4):601–20. doi:10.1089/106652700750050961.

Glymour, C. N. 2001. *The mind's arrows: Bayes nets and graphical causal models in psychology*. Cambridge, Massachusetts, USA: MIT press.

Harris, N., and M. Drton. 2013. PC algorithm for nonparanormal graphical models. *Journal of Machine Learning Research* 14 (1):3365–83.

Heckerman, D. 1997. Bayesian networks for data mining. *Data Mining and Knowledge Discov- Ery* 1 (1):79–119. doi:10.1023/A:1009730122752.

Heckerman, D., D. Geiger, and D. M. Chickering. 1995. Learning Bayesian networks: The combination of knowledge and statistical data. *Machine Learning* 20 (3):197–243. doi:10.1007/BF00994016.

Isci, S., H. Dogan, C. Ozturk, and H. H. Otu. 2013. Bayesian network prior: Network analysis of biological data using external knowledge. *Bioinformatics* 30 (6):860–67. doi:10.1093/bioinformatics/btt660.

Isozaki, T. 2014. A robust causal discovery algorithm against faithfulness violation. *Information and Media Technologies* 9 (1):121–31.

Jensen, A. L., and F. V. Jensen (1996). MIDAS-an influence diagram for management of mildew in winter wheat. In *Proceedings of the Twelfth international conference on Uncertainty in artificial intelligence*, pp. 349–56. Portland, Oregon, USA: Morgan Kaufmann Publishers Inc.

Kalisch, M., M. Mächler, D. Colombo, M. H. Maathuis, P. Bühlmann. 2012. Causal inference using graphical models with the R package pcalg. *Journal of Statistical Software* 47 (11). doi: 10.18637/jss.v047.i11.

Lagani, V., G. Athineou, A. Farcomeni, M. Tsagris, and I. Tsamardinos. 2017. Feature selection with the R package MXM: Discovering statistically-equivalent feature subsets. *Journal of Statistical Software* 80 (7). doi:10.18637/jss.v080.i07.

Pearl, J. 1988. *Probabilistic reasoning in intelligent systems: Networks of plausible reasoning*. Los Altos: Morgan Kaufmann Publishers.

R Core Team. 2016. *R: A language and environment for statistical computing*. Vienna, Austria: R Foundation for Statistical Computing.

Scutari, M. 2010. Learning bayesian networks with the bnlearn R package. *Journal of Sta-Tistical Software* 35 (3).

Scutari, M., and R. Nagarajan. 2013. Identifying significant edges in graphical models of molec- ular networks. *Artificial Intelligence in Medicine* 57 (3):207–17. doi:10.1016/j.artmed.2012.12.006.

Shevlyakov, G., and P. Smirnov. 2011. Robust estimation of the correlation coefficient: An attempt of survey. *Austrian Journal of Statistics* 40 (1&2):147–56.

Spirtes, P., and C. Glymour. 1991. An algorithm for fast recovery of sparse causal graphs. *Social Science Computer Review* 9 (1):62–72. doi:10.1177/089443939100900106.

Spirtes, P., C. N. Glymour, and R. Scheines. 2000. *Causation, Prediction, and Search*. Cambridge, Massachusetts, USA: MIT press.

Suchánek, P., F. Marecki, and R. Bucki. 2014. Self-learning Bayesian networks in diagnosis. *Procedia Computer Science* 35:1426–35. doi:10.1016/j.procs.2014.08.200.

Szekely, G. J., M. L. Rizzo. 2014. Partial distance correlation with methods for dissimi- larities. *The Annals of Statistics*. 42(6):2382–412. doi:10.1214/14-AOS1255.

Székely, G. J., M. L. Rizzo, and N. K. Bakirov. 2007. Measuring and testing dependence by correlation of distances. *The Annals of Statistics* 2769–94. doi:10.1214/009053607000000505.

Triantafillou, S., I. Tsamardinos, and A. Roumpelaki. 2014. Learning neighborhoods of high confidence in constraint-based causal discovery. In The Seventh European Workshop on Probabilistic Graphical Models 17-19September 2014 Utrecht, The Netherlands.

Tsagris, M., G. Borboudakis, V. Lagani, and I. Tsamardinos. 2017. Constraint-based causal discovery with mixed data. In *The 2017 ACM SIGKDD workshop on causal discovery*, Halifax, Nova Scotia, Canada.

Tsamardinos, I., and L. E. Brown. 2008. Bounding the false discovery rate in local bayesian network learning. In *AAAI*, 1100–05.

Tsamardinos, I., L. E. Brown, and C. F. Aliferis. 2006. The max-min hill-climbing Bayesian network structure learning algorithm. *Machine Learning* 65 (1):31–78. doi:10.1007/s10994-006-6889-7.

Verma, T., and J. Pearl (1990). Equivalence and synthesis of causal models. In *Proceedings of the Sixth Conference on Uncertainty in Artificial Intelligence*, Cambridge, MA, USA, pp. 220–27.

Yohai, V. J. 1987. High breakdown-point and high efficiency robust estimates for regression. *The Annals of Statistics* 15 (2):642–56. doi:10.1214/aos/1176350366.

Zagorecki, A., P. Orzechowski, and K. Holownia. 2013. A system for automated general medical diagnosis using Bayesian networks. *MedInfo* 192:461–65.

Appendix

R code used to generate Figure 1.

```
library(MXM)
library(pcalg)
set.seed(489)
n <- 100 ## sample size
p <- 10 ## number of variables (or nodes)
A <- MXM::rdag2(n, p = p, nei = 4) ## generate data and store the
## true adjacency matrix
id <- c(7, 8, 3, 2, 6, 1, 9, 10, 4, 5)
dat <- A$x[, id] ## re-order the data
g1 <- MXM::pc.skel(dat, method = "pearson", alpha = 0.01) ## skeleton
g1 <- MXM::pc.or(g1)$G ## orientation rules
a1 <- pcalg::pc(suffStat = list(C = cor(dat), n = n), indepTest =
gaussCItest, p = p, alpha = 0.01) ## skeleton and orientation phase
g2 <- a1@graph
g2 <- pcalg::wgtMatrix(g2, transpose = FALSE)
MXM::plotnetwork(2 * A$G)
MXM::plotnetwork(g1)
k1 <- which(g2 = = 1 & t(g2) = = 0)
k2 <- which(g2 = = 0 & t(g2) = = 1)
g2[k1] <- 2
g2[k2] <- 3
colnames(g2) <- rownames(g2) <- colnames(g1)
MXM::plotnetwork(g2)
plot(a1) ## in pcalg's format
```