Robustification of the PC-algorithm for Directed Acyclic Graphs

Markus Kalisch, Peter Bühlmann

September 13, 2007

Abstract

The PC-algorithm ([13]) was shown to be a powerful method for estimating the equivalence class of a potentially very high-dimensional acyclic directed graph (DAG) with corresponding Gaussian distribution [7]. Here we will propose a computationally efficient robustification of the PC-algorithm and prove its consistency. Furthermore, we will compare the robustified and standard version of the PC-algorithm on simulated data using the new corresponding R-package `pcalg`.

Keywords: Gaussian graphical modeling; Partial correlation analysis; R-package pcalg; Qn-estimator

1 Introduction

Graphical models are a popular probabilistic tool to analyze and visualize conditional independence relationships between random variables [see 4, 8]. Major building blocks of the models are nodes, which represent random variables and edges, which encode conditional dependence relations of the enclosing vertices. The structure of conditional independence among the random variables can be explored using the Markov properties. Of particular interest are directed acyclic graphs (DAGs), containing directed rather than undirected edges, which restrict in a sense the conditional dependence relations. These graphs can be interpreted by applying the directed Markov property [see 8].

Although estimation of a DAG from data is difficult and computationally non-trivial due to the enormous size of the space of DAGs, there have been quite successful approaches like MWST [Maximum Weight Spanning Trees; see 3, 5] or GES [Greedy Equivalent Search, see 2]. In the past, interesting hybrid methods have been developed. Very recently, [14] proposed a computationally very competitive algorithm. We also refer to their paper for a large-scale numerical comparison study among a wide range of algorithms.

An attractive alternative to greedy or structurally restricted approaches is the PC-algorithm (after its authors, Peter and Clark) from [13]. It starts from a complete, undirected graph and deletes recursively edges based on conditional independence decisions. This yields an undirected graph which can then be partially directed and further extended to represent the underlying DAG. The PC-algorithm runs in the worst case in exponential time (as a function of the number of nodes), but if the true underlying DAG is sparse, which is often a reasonable assumption, this reduces to a polynomial runtime.

It was shown in [7], that the PC-algorithm for Gaussian data is asymptotically consistent for finding the equivalence class or the skeleton of a DAG, even if the number of nodes is much larger than sample size. Here, we will propose a robustification of the Gaussian PC-algorithm. The latter is based on the maximum likelihood estimate for the covariance matrix of the variables. An effective robustification can be achieved by using a covariance estimator which is robust for every matrix element only: such elementwise robustness is feasible in high-dimensional settings and our results support that there is no need for simultaneous robustness of the whole covariance matrix. In fact, we prove asymptotic consistency and 50% breakdown point of the robustified PC-algorithm. Furthermore, we report on some finite sample numerical results. The robust PC-algorithm turns out to be very useful in presence of severely contaminated data or heavy outliers. Finally, our implementation of the robust PC-algorithm is computationally feasible for a large number of nodes, i.e., in the hundreds or thousands, if the underlying DAG is sparse.

2 Finding the Equivalence Class of a DAG

2.1 Definitions and Preliminaries

A graph $G = (V, E)$ consists of a set of nodes or vertices $V = \{1, \ldots, p\}$ and a set of edges $E \subseteq V \times V$, i.e. the edge set is a subset of ordered pairs of distinct nodes. In our setting, the set of nodes corresponds to the components of a random vector $\mathbf{X} \in \mathbf{R}^p$. An edge $(i, j) \in E$ is called directed if $(i, j) \in E$ but $(j, i) \notin E$; we then use the notation $i \rightarrow j$. If both $(i, j) \in E$ and $(j, i) \in E$, the edge is called undirected; we then use the notation $i - j$. An directed acyclic graph (DAG) is a graph G where all edges are directed and not containing any cycle.

If there is a directed edge $i \rightarrow j$, node i is said to be a parent of node j. The set of parents of node j is denoted by $pa(j)$. The adjacency set of a node j in graph G, denoted by $adj(G, j)$, are all nodes i which are directly connected to j by an edge (directed or undirected). The elements of $adj(G, j)$ are also called neighbors of or adjacent to j.

A probability distribution P on $\mathbf{R}^p$ is said to be faithful with respect to a graph G if conditional independencies of the distribution can be inferred from so-called d-separation in the graph G and vice-versa. More precisely: consider a random vector $\mathbf{X} \sim P$. Faithfulness of P with respect to G means: for any

$i, j \in V$ with $i \neq j$ and any set $\mathbf{s} \subseteq V$,

$$\begin{aligned} & \mathbf{X}^{(i)} \text{ and } \mathbf{X}^{(j)} \text{ are conditionally independent given } \{\mathbf{X}^{(r)};\ r \in \mathbf{s}\} \\ \Leftrightarrow\ & \text{node } i \text{ and node } j \text{ are d-separated by the set } \mathbf{s}. \end{aligned}$$

The notion of d-separation is described in more detail in [8, Prop. 3.25]. We remark here that faithfulness is ruling out some classes of probability distributions but non-faithful distributions of the multivariate normal family (which we will limit ourselves to) form a Lebesgue null-set in the space of distributions associated with a DAG G, see [13, Th. 3.2, Ch. 13.3].

The skeleton of a DAG G is the undirected graph obtained from G by substituting undirected edges for directed edges. A v-structure in a DAG G is an ordered triple of nodes (i, j, k) such that G contains the directed edges $i \to j$ and $k \to j$, and i and k are not adjacent in G.

It is well known that for a probability distribution P which is generated from a DAG G, there is a whole equivalence class of DAGs with corresponding distribution P [see 2, Section 2.2], and we can only identify an equivalence class of DAGs, even when having infinitely many observations. Using a result from [15], we can characterize equivalent classes more precisely: Two DAGs are equivalent if and only if they have the same skeleton and the same v-structures.

A common tool for visualizing equivalence classes of DAGs are *completed partially directed acyclic graphs (CPDAG).* A partially directed acyclic graph (PDAG) is a graph where some edges are directed and some are undirected and one cannot trace a cycle by following the direction of directed edges and any direction for undirected edges. Equivalence among PDAGs or of PDAGs and DAGs can be decided as for DAGs by inspecting the skeletons and v-structures. A PDAG is *completed*, if (1) every directed edge exists also in every DAG belonging to the equivalence class of the DAG and (2) for every undirected edge $i - j$ there exists a DAG with $i \to j$ and a DAG with $i \leftarrow j$ in the equivalence class. CPDAGs encode all independence informations contained in the corresponding equivalence class. It was shown in [1] that two CPDAGs are identical if and only if they represent the same equivalence class, that is, they represent a equivalence class uniquely.

Although the main goal is to identify the CPDAG, the skeleton itself already contains interesting information. In particular, if P is faithful with respect to a DAG G,

$$\begin{aligned} & \text{there is an edge between nodes } i \text{ and } j \text{ in the skeleton of DAG } G \\ \Leftrightarrow\ & \text{for all } \mathbf{s} \subseteq V \setminus \{i, j\},\ \mathbf{X}^{(i)} \text{ and } \mathbf{X}^{(j)} \text{ are conditionally dependent} \\ & \text{given } \{\mathbf{X}^{(r)};\ r \in \mathbf{s}\}, \end{aligned} \tag{1}$$

[13, Th. 3.4, Ch. 13.5].

As we will see later in more detail, estimating the CPDAG consists of two main parts: (1) Estimation of the skeleton and (2) partial orientation of edges.

This will provide the structure of the remaining part of this paper. First, we will discuss the part of the PC-algorithm that leads to the skeleton. Afterwards, we will complete the algorithm by discussing the extensions for finding the CPDAG. We will use the same format when discussing theoretical properties of the PC-algorithm.

2.2 The PC-algorithm for Finding the Skeleton

A naive strategy for finding the skeleton would be to check conditional independencies given all subsets $\mathbf{s} \subseteq V \setminus \{i, j\}$ (see formula (1)), i.e. all partial correlations in the case of multivariate normal distributions as first suggested by [16]. This would become computationally infeasible and statistically ill-posed for p larger than sample size. A much better approach is used by the PC-algorithm which is able to exploit sparseness of the graph.

2.2.1 Population Version for the skeleton

In the population version of the PC-algorithm, we assume that perfect knowledge about all necessary conditional independence relations is available. We refer here to the PC-algorithm what others call the first part of the PC-algorithm; the other part is described in Algorithm 2 in Section 2.3.

The (first part of the) PC-algorithm is given in Algorithm 1. A proof that this algorithm produces the correct skeleton can be easily deduced from Theorem 5.1 in [13, Ch. 13.15].

2.2.2 Sample Version for the Skeleton

For finite samples, we need to estimate conditional independencies. We limit ourselves to the Gaussian case, where all nodes correspond to random variables with a multivariate normal distribution. Furthermore, we assume faithful models, i.e. the conditional independence relations can be read off the graph and vice versa; see Section 2.1.

In the Gaussian case, conditional independencies can be inferred from partial correlations. They can be defined recursively: for $i, j \in V$, $\mathbf{k} \subset V \backslash \{i, j\}$ and some $h \in \mathbf{k}$,

$$\rho_{i,j|\mathbf{k}} = \frac{\rho_{i,j|\mathbf{k}\backslash h} - \rho_{i,h|\mathbf{k}\backslash h}\rho_{j,h|\mathbf{k}\backslash h}}{\sqrt{(1-\rho^2_{i,h|\mathbf{k}\backslash h})(1-\rho^2_{j,h|\mathbf{k}\backslash h})}}. \tag{2}$$

Estimation of the partial correlations is done via the plug-in principle. Hence, due to the recursive nature, $\hat{\rho}_{i,j|\mathbf{k}}$ can be inferred from the correlation matrix $[\hat{\rho}_{i,j}]_{i,j=1,\ldots,p}$. Usually, $\hat{\rho}_{i,j}$ is the sample correlation, i.e. the Gaussian maximum likelihood estimator. We will discuss a robust version of it in section 3. For testing whether a partial correlation is zero or not, we apply Fisher's z-transform

$$Z(i, j|\mathbf{k}) = \frac{1}{2} \log \left(\frac{1 + \hat{\rho}_{i,j|\mathbf{k}}}{1 - \hat{\rho}_{i,j|\mathbf{k}}} \right). \tag{3}$$

Algorithm 1 The PC$_{pop}$-algorithm

```
INPUT: Vertex Set $V$, Conditional Independence Information
OUTPUT: Estimated skeleton $C$, separation sets $S$ (only needed when directing the skeleton afterwards)
Form the complete undirected graph $\tilde{C}$ on the vertex set V.
$\ell = -1; \quad C = \tilde{C}$
repeat
  $\ell = \ell + 1$
  repeat
    Select a (new) ordered pair of nodes $i$,$j$ that are adjacent in $C$ such that $|adj(C,i) \setminus \{j\}| \geq \ell$
    repeat
      Choose (new) $\mathbf{k} \subseteq adj(C,i) \setminus \{j\}$ with $|\mathbf{k}| = \ell$.
      if $i$ and $j$ are conditionally independent given $\mathbf{k}$ then
        Delete edge $i, j$
        Denote this new graph by $C$.
        Save $\mathbf{k}$ in $S(i,j)$ and $S(j,i)$
      end if
    until edge $i, j$ is deleted or all $\mathbf{k} \subseteq adj(C,i) \setminus \{j\}$ with $|\mathbf{k}| = \ell$ have been chosen
  until all ordered pairs of adjacent variables $i$ and $j$ such that $|adj(C,i) \setminus \{j\}| \geq \ell$ and $\mathbf{k} \subseteq adj(C,i) \setminus \{j\}$ with $|\mathbf{k}| = \ell$ have been tested for conditional independence
until for each ordered pair of adjacent nodes $i$,$j$: $|adj(C,i) \setminus \{j\}| < \ell$.
```

Classical decision theory yields then the following rule when using the significance level α. Reject the null-hypothesis $H_0(i,j|\mathbf{k}) : \rho_{i,j|\mathbf{k}} = 0$ against the two-sided alternative $H_A(i,j|\mathbf{k}) : \rho_{i,j|\mathbf{k}} \neq 0$ if $\sqrt{n - |\mathbf{k}| - 3}|Z(i,j|\mathbf{k})| > \Phi^{-1}(1-\alpha/2)$, where $\Phi(\cdot)$ denotes the cdf of $\mathcal{N}(0,1)$.

The sample version of the PC-algorithm is almost identical to the population version in Section 2.2.1.

The PC-algorithm

> Run the PC$_{pop}(m)$-algorithm as described in Section 2.2.1 but replace in line 11 of Algorithm 1 the if-statement by
> **if** $\sqrt{n - |\mathbf{k}| - 3}|Z(i,j|\mathbf{k})| \leq \Phi^{-1}(1-\alpha/2)$ **then**.

The only tuning parameter of the PC-algorithm is α, i.e. the significance level for testing partial correlations.

2.3 Extending the Skeleton to the Equivalence Class

While finding the skeleton as in Algorithm 1, we recorded the separation sets that made edges drop out in the variable denoted by S. This was not necessary for finding the skeleton itself but will be essential for extending the skeleton to the equivalence class. In Algorithm 2 we describe the work of [10, p.50f] to extend the skeleton to a CPDAG belonging to the equivalence class of the underlying DAG.

Algorithm 2 Extending the skeleton to a CPDAG

```
INPUT: Skeleton $G_{skel}$, separation sets $S$
OUTPUT: CPDAG $G$
for all pairs of nonadjacent variables $i, j$ with common neighbour $k$ do
  if $k \notin S(i,j)$ then
    Replace $i - k - j$ in $G_{skel}$ by $i \rightarrow k \leftarrow j$
  end if
end for
In the resulting PDAG, try to orient as many undirected edges as possible by
repeated application of the following three rules:
R1 Orient $j - k$ into $j \rightarrow k$ whenever there is an arrow $i \rightarrow j$ such that $i$ and
$k$ are nonadjacent.
R2 Orient $i - j$ into $i \rightarrow j$ whenever there is a chain $i \rightarrow k \rightarrow j$.
R3 Orient $i - j$ into $i \rightarrow j$ whenever there are two chains $i - k \rightarrow j$ and
$i - l \rightarrow j$ such that $k$ and $l$ are nonadjacent.
R4 Orient $i - j$ into $i \rightarrow j$ whenever there are two chains $i - k \rightarrow l$ and
$k \rightarrow l \rightarrow j$ such that $k$ and $l$ are nonadjacent.
```

The output of Algorithm 2 is a CPDAG, which was first proved by [9]. The computational complexity of the PC-algorithm is difficult to evaluate exactly, but the worst case is bounded by $O(n \max\{p^q, p^2\})$, where q is the maximal size of the adjacency sets. We note that the bound may be very loose for many distributions. For non-worst cases we can do the computations for quite large values of q and fairly dense graphs, e.g. some nodes j have neighborhoods of size up to $|adj(G, j)| = 30$.

3 Robustification of the PC-algorithm

The only quantity that has to be estimated from data in order to run the PC-algorithm is the correlation among all pairs of variables $\mathbf{X}^{(i)}$ and $\mathbf{X}^{(j)}$. The correlation estimates are then used to iteratively compute estimates of partial correlations as in (2), which in turn are used to obtain z-values in (3). It is well known, that the standard sample correlation estimator (the Gaussian MLE) is not robust. Therefore, a small quantity of outliers suffices to completely distort the resulting graph. In order to robustify the PC-algorithm, we first note, that the correlation $\rho(\mathbf{X}^{(i)}, \mathbf{X}^{(j)}) = \rho_{i,j}$ can be written in terms of standard deviations $\sigma_X = \sqrt{Var(X)}$:

$$\rho_{i,j} = \rho_{\mathbf{X}^{(i)},\mathbf{X}^{(j)}} = \frac{\sigma^2_{a\mathbf{X}^{(i)}+b\mathbf{X}^{(j)}} - \sigma^2_{a\mathbf{X}^{(i)}-b\mathbf{X}^{(j)}}}{\sigma^2_{a\mathbf{X}^{(i)}+b\mathbf{X}^{(j)}} + \sigma^2_{a\mathbf{X}^{(i)}-b\mathbf{X}^{(j)}}} \tag{4}$$

where $a = \frac{1}{\sigma_{\mathbf{X}^{(i)}}}$ and $b = \frac{1}{\sigma_{\mathbf{X}^{(j)}}}$, see [6]. To robustify the scale estimate, we replace the empirical standard deviation by a robust scale estimate. A good choice is the Q_n estimator [11] which is defined as a scaled order statistics of differences in a sample of n realisations $X_1, ..., X_n$:

$$Q_{n;X} = d\{|X_i - X_j|; i < j\}_{(k)}, \tag{5}$$

where $d = 2.2219$ is a constant factor and $k = \binom{h}{2} \approx \frac{\binom{n}{2}}{4}$ with $h = [n/2] + 1$ roughly equal to half the number of observations. The Q_n estimator has some attractive properties. Apart from its simple and explicit formula, the definition is suitable for asymmetric distributions. The breakdown point is 50% and the efficiency at Gaussian distributions is 82%. By using an algorithm by [11], Q_n can be computed with $O(n)$ memory storage and $O(n \log n)$ essential operations. Thus, we obtain a robust estimate for the individual correlations by using

$$\hat{\rho}_{Q_n;i,j} = \frac{Q^2_{n;a\mathbf{X}^{(i)}+b\mathbf{X}^{(j)}} - Q^2_{n;a\mathbf{X}^{(i)}-b\mathbf{X}^{(j)}}}{Q^2_{n;a\mathbf{X}^{(i)}+b\mathbf{X}^{(j)}} + Q^2_{n;a\mathbf{X}^{(i)}-b\mathbf{X}^{(j)}}} \tag{6}$$

The robust version of partial correlation is, analogously to (2),

$$\hat{\rho}_{Q_n;i,j|\mathbf{k}} = \frac{\hat{\rho}_{Q_n;i,j|\mathbf{k}\setminus h} - \hat{\rho}_{Q_n;i,h|\mathbf{k}\setminus h}\hat{\rho}_{Q_n;j,h|\mathbf{k}\setminus h}}{\sqrt{(1 - \hat{\rho}^2_{Q_n;i,h|\mathbf{k}\setminus h})(1 - \hat{\rho}^2_{Q_n;j,h|\mathbf{k}\setminus h})}}. \tag{7}$$

The corresponding z-value is then defined as

$$Z_{Q_n}(i, j|\mathbf{k}) = \frac{1}{2} \log\left(\frac{1 + \hat{\rho}_{Q_n;i,j|\mathbf{k}}}{1 - \hat{\rho}_{Q_n;i,j|\mathbf{k}}}\right). \tag{8}$$

Finally, we define the robust PC-algorithm as follows.

The robust PC-algorithm

> Run the PC-algorithm as described in section 2.2.2 but replace $Z(i,j|\mathbf{k})$ by $Z_{Q_n}(i,j|\mathbf{k})$.

4 Consistency of the PC algorithm

The standard PC algorithm on Gaussian data was shown to be asymptotically consistent for finding the equivalence class or the skeleton of a DAG [7]. This result also holds for high-dimensional settings where $p \gg n$.

In this section, we will show that in the Gaussian case (i.e. uncontaminated model) and assuming a finite dimension $p < \infty$, i.e. a stricter assumption than in [7], the PC algorithm is consistent whenever every element of the covariance matrix is estimated consistently. Thus, the previous result is generalized under stricter conditions. This will yield the consistency of the robust PC-algorithm as a special case.

We will use the following assumptions:

(A1) The distribution P is multivariate Gaussian $\mathcal{N}(\mu, \mathbf{\Sigma})$ with finite dimensionality $p < \infty$.

(A2) The distribution P is faithful to the DAG G.

(A3) The distribution P is not degenerated, i.e., $\Sigma_{ii} > 0$ and $|\rho_{ij}| = |\Sigma_{ij}/(\Sigma_{ii}\Sigma_{jj})^{\frac{1}{2}}| < 1$ for all $i \neq j$.

Denote by $\hat{G}_{skel}(\alpha)$ the estimate of the skeleton using the PC-algorithm (standard or robust version) when using significance level α. Moreover, we denote by G_{skel} the skeleton of a DAG G.

Theorem 1 *Assume (A1)-(A3) and let $\hat{\Sigma}_{ij}$ be any estimate of Σ_{ij} which is (elementwise) consistent for all $i, j \in \{1, ..., p\}$ having convergence rate n^{-r} ($r \leq \frac{1}{2}$), i.e., $n^r(\hat{\Sigma}_{ij} - \Sigma_{ij}) = O_p(1)$ for all i, j. Then, for any sequence $(\alpha_n)_{n \in N}$ with $\frac{1-\Phi(n^{-r+\frac{1}{2}})}{\alpha_n} \to \infty$, $\frac{\alpha_n}{1-\Phi(\sqrt{n})} \to \infty$ $(n \to \infty)$,*

$$\mathbb{P}[\hat{G}_{skel}(\alpha_n) = G_{skel}] \to 1 \quad (n \to \infty).$$

A proof is given in section 7.

Remark: Typically, $r = \frac{1}{2}$ and hence we require $\alpha_n \to 0$, $\frac{\alpha_n}{1-\Phi(\sqrt{n})} \to \infty$ $(n \to \infty)$.

The consistency result of Theorem 1 can be easily extended to the equivalence class of the DAG. We denote by G_{CPDAG} the CPDAG of a DAG G.

Theorem 2 *Assume (A1)-(A3) and let* $\hat{\Sigma}_{ij}$ *be any estimate of* Σ_{ij} *which is (elementwise) consistent for all* $i, j \in \{1, ..., p\}$ *having convergence rate* n^{-r} $(r \leq \frac{1}{2})$*, i.e.,* $n^r(\hat{\Sigma}_{ij} - \Sigma_{ij}) = O_p(1)$ *for all* i, j*. Then, for any sequence* $(\alpha_n)_{n \in N}$ *with* $\frac{1-\Phi(n^{-r+\frac{1}{2}})}{\alpha_n} \to \infty$, $\frac{\alpha_n}{1-\Phi(\sqrt{n})} \to \infty$ $(n \to \infty)$,

$$\mathbb{P}[\hat{G}_{CPDAG}(\alpha_n) = G_{CPDAG}] \to 1 \quad (n \to \infty).$$

A proof is given in section 7. Using Theorem 1, it is now easy to analyze the consistency properties of the robust PC-algorithm:

Lemma 1 *Assume (A1). Then, the estimate* $\hat{\rho}_{Q_n;i,j}$ *defined in (6) and the corresponding covariance estimate* $\hat{\Sigma}_{Q_n;ij} = \hat{\rho}_{Q_n;i,j} Q_{n;\mathbf{X}^{(\mathrm{i})}} Q_{n;\mathbf{X}^{(\mathrm{j})}}$ *are* $\sqrt{n}$*-consistent.*

A proof is given in section 7. Combining Theorem 1 and Lemma 1, we obtain:

Corollary 1 *Assume (A1)-(A3). The robust PC-algorithm based on the* Q_n*-estimator is consistent in the sense of Theorem 1 and Theorem 2 (with* $r = \frac{1}{2}$*).*

A proof is given in section 7. Furthermore, the breakdown point of the robust PC-algorithm can be easily found:

Proposition 1 *The breakdown point of the robust PC-algorithm is 50%.*

A proof is given in section 7.

5 Numerical results

5.1 Simulating data

In order to simulate data, we first construct an adjacency matrix A as follows:

1. Fix an ordering of the variables.

2. Fill the adjacency matrix A with zeros.

3. Replace every matrix entry in the lower triangle (below the diagonal) by independent realizations of Bernoulli(s) random variables with success probability s where $0 < s < 1$. We will call s the sparseness of the model.

4. Replace each entry with a 1 in the adjacency matrix by independent realizations of a Uniform($[0.1, 1]$) random variable.

This then yields a matrix A whose entries are zero or in the range $[0.1, 1]$. The corresponding DAG draws a directed edge from node i to node j if $i < j$ and $A_{ji} \neq 0$. The DAGs (and skeletons thereof) that are created in this way have the following property: $\mathbb{E}[N_i] = s(p-1)$, where N_i is the number of neighbors of a node i.

Thus, a low sparseness parameter s implies few neighbors and vice-versa. The matrix A will be used to generate the data as follows. The value of the random variable $X^{(1)}$, corresponding to the first node, is given by

$$\begin{aligned}&\epsilon^{(1)} \sim N(0,1)\\ &X^{(1)} = \epsilon^{(1)}\end{aligned}$$

and the values of the next random variables (corresponding to the next nodes) can be computed recursively as

$$X^{(i)} = \sum_{k=1}^{i-1} A_{ik} X^{(k)} + \epsilon^{(i)} \ (i = 2, \ldots, p),$$

where all $\epsilon^{(1)}, \ldots, \epsilon^{(p)}$ are i.i.d., and $\epsilon^{(i)}$ is independent from $\{X^{(j)}, j < i\}$. Regarding the distribution of ϵ, we consider either $N(0,1)$, $0.9N(0,1) + 0.1t_3(0,1)$ (i.e. 10% contamination by a t-distribution with three degrees of freedom) or $0.9N(0,1) + 0.1Cauchy(0,1)$ (i.e. 10% contamination by a standard Cauchy distribution).

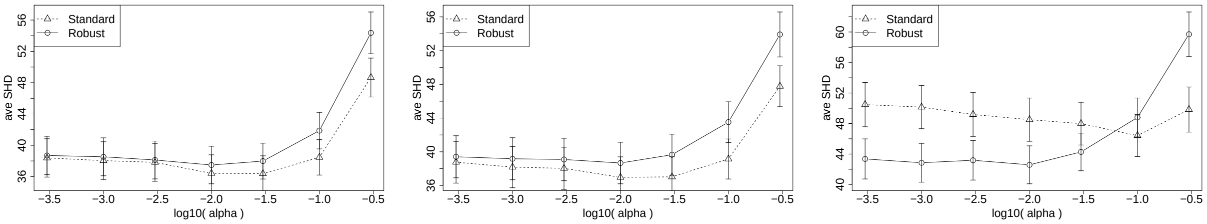
5.2 Performance of the robust PC-algorithm

We compare the standard Gaussian and robust version of the PC-algorithm on Gaussian and contaminated Gaussian (10% t_3 or *Cauchy*) data on a wide range of parameter settings. To this end, we simulated each combination of the following variable settings:

- $n \in \{100, 500, 1000, 5000, 10000\}$
- $p \in \{10, 25, 50\}$
- $\alpha \in \{0.0001, 0.0003, 0.001, 0.003, 0.01, 0.03, 0.1, 0.3\}$
- $\mathbb{E}[N] \in \{3, 6\}$

For each of the possible 210 combinations, 30 replicates were produced. In order to investigate the influence of the single tuning parameter α, we show the average Structural Hamming Distance [SHD; 14] together with 95% confidence intervals of the simulation results grouped by various values of α. Roughly speaking, the SHD counts the number of edge insertions, deletions and flips in order to transfer the estimated CPDAG into the correct CPDAG. Thus, a large SHD indicates a poor fit, while a small SHD indicates a good fit.

In Figure 1 (a) and Figure 1 (b) the rather similar results for the Gaussian distribution and the Gaussian distribution with 10% contamination from a t_3 distribution are shown. One can see that the standard Gaussian PC-algorithm performed better, since the average SHD is lower. The difference is highly significant (one-sided, paired Wilcoxon Test). For the standard Gaussian PC-algorithm, the values $\alpha = 0.01$ and $\alpha = 0.03$ yield a significantly lower average



(a) Gaussian

(b) Contamination: 10% t_3

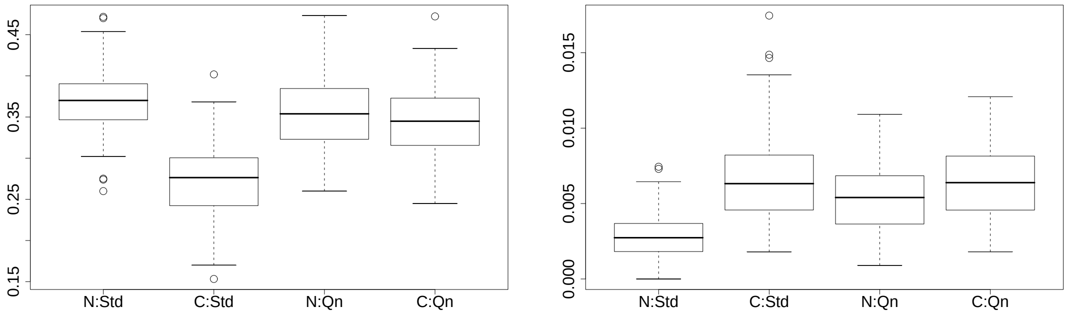
(c) Contamination: 10% Cauchy

Figure 1: Comparison between standard Gaussian (triangles) and robust (circles) PC-algorithm: The average Structural Hamming Distance (SHD) over a wide range of parameter settings is shown versus the tuning parameter α (together with 95% confidence intervals). Without or with moderate outliers, the standard Gaussian PC-algorithm performs slightly better. If the outliers are very severe, the standard Gaussian PC-algorithm breaks down and the robust version is superior.

SHD (two sided Wilcoxon Test with Bonferroni correction) than for the remaining values of α. For the robust PC-algorithm, the value of $\alpha = 0.01$ also seems to produce the lowest average SHD, but the result is not as clear (not significant). Therefore, when dealing with problems in the realm of the covered parameter settings, we would advocate $\alpha \approx 0.01$.

In Figure 1 (c), we show the corresponding result for Gaussian data with 10% contamination from a standard Cauchy distribution. The situation looks completely different than before. For values of $\alpha \leq 0.03$, the robust PC-algorithm performs significantly better than the standard Gaussian PC-algorithm while for $\alpha > 0.03$, the standard PC-algorithm performs better. For the standard Gaussian PC-algorithm, the average SHD for $\alpha = 0.1$ is significantly lower than for the remaining values of α. For the robust PC-algorithm, $\alpha = 0.3$ and $\alpha = 0.1$ produce significantly worse results. For $\alpha \leq 0.03$, no choice of α leads to significantly better results than another. Note that by using the robust PC-algorithm, the SHD can be decreased substantially yielding a much better fit.

We also analyzed the behavior of the true positive rate (TPR) and the false positive rate (FPR) and show the typical results by using an example. We used $n = 100$, $p = 50$, $\alpha = 0.01$, average neighborhood size $\mathbb{E}[N] = 5$ and we conducted 100 repetitions. In each of the 100 replications, we chose a graphical structure and generated errors according to a standard normal distribution and according to a mixture of 90% standard normal and 10% standard Cauchy. We then used the standard and the robust PC-algorithm to estimate the underlying skeleton of the graph. The result is given in Figure 2. As can be seen in Figure 2(a), the TPR for the standard estimate drops dramatically if outliers are introduced, whereas it does not drop significantly for the robust method. Analogously, the FPR in Figure 2(b) increases significantly for the standard estimate,



(a) TPR

(b) FPR

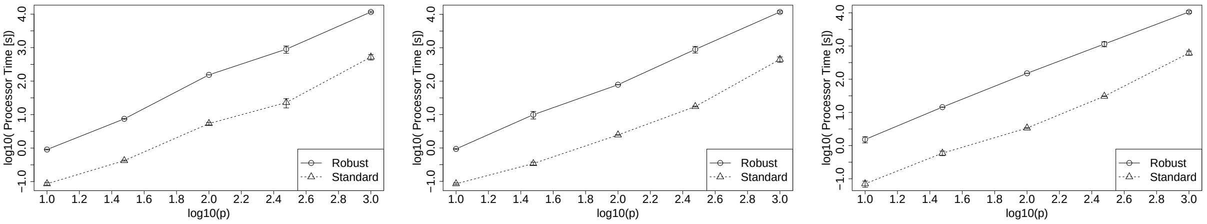
Figure 2: Example of typical behavior of TPR and FPR ($n = 100$, $p = 50$, $\alpha = 0.01$, $\mathbb{E}[N] = 5$ on 100 repetitions). In contrast to the standard method, the TPR and FPR of the robust method don't change very much, when outliers are introduced. Moreover, the robust method achieves significantly higher TPR while keeping FPR at a level comparable to the standard method.(N: Standard Normal Error, C: 10% Cauchy, Std: Standard method, Qn: Robust method)

while it does not change significantly for the robust method. Furthermore, in the contaminated setting, the TPR for the robust method is significantly higher than for the standard method, while the FPR of both methods don't vary much. Thus, the main advantage of the robust method lies in achieving a higher TPR while keeping the FPR rather constant.

5.3 Computational Complexity

The computational complexity of the robust PC-algorithm is up to a $\log(n)$-factor the same as for the analogue with the standard Gaussian PC-algorithm, namely $O(n \log(n) \max\{p^q, p^2\})$ in the worst case.

We provide a small example of the processor time for estimating a CPDAG by using the PC-algorithm. The runtime analysis was done on an AMD Athlon 64 X2 Dual Core Processor 5000+ with 2.6 GHz and 4 GB RAM running on Linux and using R 2.4.1. The number of variables varied between $p = 10$ and $p = 1000$ while the number of samples was fixed at $n = 1000$. The sparseness was $E[N] = 3$. For each parameter setting, 3 replicates were used. In each case, the significance level used in the PC-algorithm was $\alpha = 0.01$. Figure 3 gives a graphical impression of the results of this example. The runtimes seem not to be influenced much by the type of contamination. For both uncontaminated and contaminated (t_3 or Cauchy) data, the standard Gaussian PC-algorithm is on average roughly by a factor of 25 faster. The average processor times together with their standard deviations for estimating the CPDAG are given in Table 5.1. We only give the values for Cauchy-contamination, since the values in the



(a) Gaussian

(b) Contamination: 10% t_3

(c) Contamination: 10% Cauchy

Figure 3: Average processor time of the standard Gaussian and robust PC-algorithm for graphs of different sizes and different kinds of contamination by outliers. The standard Gaussian PC-algorithm is on average roughly by a factor of 25 faster.

other cases are very similar.

p	standard PC [s]	robust PC [s]
10	0.07 (0.02)	1.53 (0.3)
30	0.59 (0.09)	14.4 (0.1)
100	3.4 (0.1)	151 (1)
300	31 (0.4)	1148 (192)
1000	614 (88)	10600 (860)

Table 5.1: Average processor time of the standard Gaussian and robust PC-algorithm for graphs of different sizes, with standard errors in parentheses. The standard PC-algorithm is on average roughly by a factor of 25 faster.

Using the standard and the robust PC-algorithm, graphs of $p = 1000$ nodes could be estimated in about 10 minutes and 3 hours, while graphs with $p = 100$ nodes it took about 3 seconds and about 2 minutes, respectively. The simulation studies show that the standard PC-algorithm already is rather insensitive to outliers, provided, they are not too severe. The effect of very heavy outliers can be dramatically reduced by using the robust PC-algorithm; this increases the computational burden by roughly one order of magnitude.

6 R-Package `pcalg`

The R-package `pcalg` can be used to estimate from data the underlying skeleton or equivalence class of a DAG. To use this package, the statistics software `R` needs to be installed. Both `R` and the R-package `pcalg` are publicly available (with open source code) at
`http://www.r-project.org`.

In the following, we show an example of how to generate a random DAG, draw samples and infer from data the skeleton and the equivalence class of the original DAG. The line width of the edges in the resulting skeleton and CPDAG can be adjusted to correspond to the reliability of the estimated dependencies. The line width is proportional to the smallest value of $\sqrt{n - |\mathbf{k}| - 3}\ Z(i, j, \mathbf{k})$ causing an edge, see also formula (3). Therefore, thick lines are reliable.

```
library(pcalg)
## define parameters
p <- 10 # number of random variables
n <- 10000 # number of samples
s <- 0.2 # sparsness of the graph
```

For simulating data as described in Section 5.1:

```
## generate random data
set.seed(10)
g <- randomDAG(p,s) # generate a random DAG
d <- rmvDAG(n,g,errDist=''mixt3'') # generate random samples
```

Then we estimate the underlying skeleton by using the function `pcAlgo` and extend the skeleton to the CPDAG by using the function `udag2cpdag`.

```
gSkel <-
  pcAlgo(d,alpha=0.05,corMethod=''QnStable'') # estimate of the skeleton
gCPDAG <-
  udag2cpdag(gSkel)
```

The results can be easily plotted using the following commands:

```
plot(g)
plot(gSkel,zvalue.lwd=TRUE)
plot(gCPDAG,zvalue.lwd=TRUE)
```

The original DAG is shown in Figure 4(a). The estimated skeleton and the estimated CPDAG are shown in Figure 4(b) and Figure 4(c), respectively. Note the differing line width, which indicates the reliability (minimal z-values as in (3)) of the involved statistical tests (thick lines are reliable).

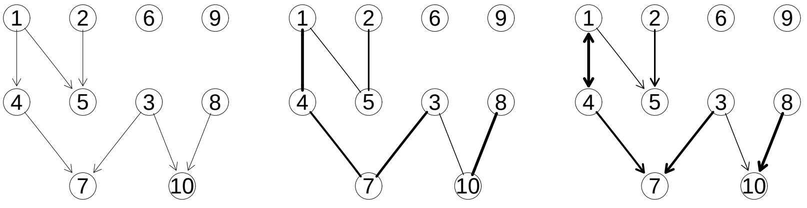
Acknowledgments

We'd like to thank Martin Mächler for his help with developing the R-package pcalg. Markus Kalisch was supported by the Swiss National Science Foundation (No. 200021-105276 and No. 200020-113270/1).

7 Proofs

7.1 Proof of Theorem 1 and Theorem 2

The z-values $Z_{ij|\mathbf{k}}$ are defined as in (3) or (8). Denote the empirical z-values by $\hat{z}_{ij|\mathbf{k}}$, which are defined in terms of the corresponding empirical correlations



(a) True DAG (b) Estimated Skeleton (c) Estimated CPDAG

Figure 4: These plots were generated using the R-package `pcalg` as described in section 6. (a) The true DAG. (b) The estimated skeleton using the R-function `pcAlgo` with $\alpha = 0.05$. (c) The estimated CPDAG using the R-function `udag2cpdag`. Double-headed arrows indicate undirected edges. Line width encodes the reliability (z-values) of the dependence estimates (thick lines are reliable).

$\hat{\rho}_{i,j|\mathbf{k}}$, and they are functions of $\hat{\Sigma}_{i,j}$ themselves.

Since $\hat{\Sigma}_{ij}$ is a consistent estimate, elementary calculations show that for any $\gamma > 0$,

$$\mathbb{P}[|\hat{z}_{ij} - z_{ij}| > \gamma] \to 0 \quad (n \to \infty) \quad \forall i, j \in \{1, ..., p\},$$

where $\hat{z}_{i,j}$ and $z_{i,j}$ are the z-transforms of $\hat{\rho}_{i,j}$ and $\rho_{i,j}$, respectively. Furthermore, since $p < \infty$ is finite, we obtain: for any $\gamma > 0$,

$$\max_{i,j,\mathbf{k}\subset V\setminus\{i,j\}} \mathbb{P}[|\hat{z}_{ij|\mathbf{k}} - z_{ij|\mathbf{k}}| > \gamma] \to 0 \quad (n \to \infty), \tag{9}$$

analogous to the computation in [17]. An error occurs in the sample PC-algorithm if there exists a pair of nodes i, j and a conditioning set $\mathbf{k} \subset V\setminus\{i, j\}$ such that error event $E^I_{i,j|\mathbf{k}} \cup E^{II}_{i,j|\mathbf{k}}$ (see below) occurs. In more detail, for some suitable c (see below), the error events are:

$$\text{type I error } E^I_{i,j|\mathbf{k}} : \ |\hat{z}_{i,j|\mathbf{k}}| > \frac{c}{2} \text{ and } z_{i,j|\mathbf{k}} = 0,$$
$$\text{type II error } E^{II}_{i,j|\mathbf{k}} : \ |\hat{z}_{i,j|\mathbf{k}}| \le \frac{c}{2} \text{ and } z_{i,j|\mathbf{k}} \neq 0.$$

Using (9) and a proper choice of c (see below)

$$\max_{i,j,\mathbf{k}} \mathbb{P}[E^I_{i,j|\mathbf{k}}] = \max_{i,j,\mathbf{k}} \mathbb{P}[|\hat{z}_{ij|\mathbf{k}} - z_{ij|\mathbf{k}}| > \frac{c}{2}] \to 0$$

and

$$\max_{i,j,\mathbf{k}} \mathbb{P}[E^{II}_{i,j|\mathbf{k}}] \le \max_{i,j,\mathbf{k}} \mathbb{P}[|\hat{z}_{ij|\mathbf{k}} - z_{ij|\mathbf{k}}| > \frac{c}{2}] \to 0.$$

Note, that c exists (e.g. $c = \min_{i,j,\mathbf{k}}\{|z_{ij|\mathbf{k}}|; z_{ij|\mathbf{k}} \neq 0\}$). In fact, we can use any $c = c_n$ with $c_n \to 0$, $c_n n^r \to \infty$ $(n \to \infty)$ ((9) also holds for $\gamma = \gamma_n$ with $\gamma_n n^r \to \infty$). This immediately translates to the specified range of the sequence $(\alpha_n)_{n\in N}$ in Theorem 1.

Therefore, we finally get

$$\begin{aligned}
& \mathbb{P}[\text{an error occurs in the PC-algorithm}] \\
\leq\ & \mathbb{P}[\bigcup_{i,j,\mathbf{k}\subset V\setminus\{i,j\}} E^{I}_{i,j|\mathbf{k}} \cup E^{II}_{i,j|\mathbf{k}}] \\
\leq\ & C_p \max_{i,j,\mathbf{k}\subset V\setminus\{i,j\}} \mathbb{P}[E^{I}_{i,j|\mathbf{k}} \cup E^{II}_{i,j|\mathbf{k}}] \to 0 \quad (n \to \infty) \qquad (10)
\end{aligned}$$

since $C_p < \infty$ using $p < \infty$.

7.2 Proof of Theorem 2

As mentioned in section 2.3, due to the result of [9], it is sufficient to estimate the correct skeleton and separation sets. The proof of Theorem 1 also covers the issue of choosing the correct separation sets S, that is, the probability of estimating the correct sets S goes to one as $n \to \infty$. Hence, the proof of Theorem 2 is completed.

7.3 Proof of Lemma 1

Consider first a univariate sample $X_1, ..., X_n \quad i.i.d. \sim N(0, \sigma^2)$ (without loss of generality with mean 0). Define $h(X_i, X_j) = |X_i - X_j|$ and $H_n(y) = \frac{1}{\binom{n}{2}} \sum_{i<j} I(h(X_i, X_j) \leq y)$. Then, the Q_n-estimator can be written as a functional T of H_n:

$$Q_n = T(H_n) = dH_n^{-1}(q)$$

where $d = 2.2219$ is a constant and $q = \frac{\binom{h}{2}}{\binom{n}{2}}$ with $h = [n/2] + 1$. Due to the Gaussian assumption, the assumptions of Theorem 3.1 in [12] hold, and we obtain

$$n^{1/2}[T(H_n) - \sigma] \to N(0, \tau^2_\infty)$$

for some limiting variance τ^2_∞. Therefore, Q_n is a $\sqrt{n}$-consistent estimate of the standard deviation σ of a univariate sample (choosing the appropriate constant d). It follows immediately that $\hat{\Sigma}_{Q_n;ij}$ and $\hat{\rho}_{Q_n;ij}$ are $\sqrt{n}$-consistent.

7.4 Proof of Proposition 1

Because the breakdown point of the Q_n-estimator for the scale of any of the $\mathbf{X}^{(i)}$'s is 50%, also $\hat{\rho}_{Q_n;i,j}$ and $\hat{\rho}_{Q_n;i,j|\mathbf{k}}$ in (6) and (7) respectively have breakdown point 50%. The claim then follows by the definition of the robust PC-algorithm.

References

[1] D.M. Chickering. Learning equivalence classes of bayesian-network structures. *Journal of Machine Learning Research*, 2:445–498, 2002.

[2] D.M. Chickering. Optimal structure identification with greedy search. *Journal of Machine Learning Research*, 3:507–554, 2002.

[3] C. Chow and C. Liu. Approximating discrete probability distributions with dependence trees. *IEEE Transactions on Information Theory*, 14(3):462–467, 1968.

[4] D. Edwards. *Introduction to Graphical Modelling.* Springer Verlag, 2nd edition edition, 2000.

[5] D. Heckerman, D. Geiger, and D.M. Chickering. Learning bayesian networks: The combination of knowledge and statistical data. *Machine Learning*, 20:197–243, 1995.

[6] P.J. Huber. *Robust Statistics.* John Wiley, 1981.

[7] M. Kalisch and P. Bühlmann. Estimating high-dimensional directed acyclic graphs with the PC-algorithm. *Journal of Machine Learning Research*, 8:613–636, 2007.

[8] S. Lauritzen. *Graphical Models.* Oxford University Press, 1996.

[9] C. Meek. Causal inference and causal explanation with background knowledge. In P.Besnard and S.Hanks, editors, *Uncertainty in Artificial Intelligence*, volume 11, pages 403–410, 1995.

[10] J. Pearl. *Causality.* Cambridge University Press, 2000.

[11] P.J. Rousseeuw and C. Croux. Alternatives to the median absolute deviation. *Journal of the American Statistical Association*, 88(424):1273–1283, 1993.

[12] R.J. Serfling. Generalized L-, M-, and R-Statistics. *The Annals of Statistics*, 12(1):76–86, 1984.

[13] P. Spirtes, C. Glymour, and R. Scheines. *Causation, Prediction, and Search.* The MIT Press, 2nd edition, 2000.

[14] I. Tsamardinos, L.E. Brown, and C.F. Aliferis. The max-min hill-climbing bayesian network structure learning algorithm. *Machine Learning*, 65(1):31–78, 2006.

[15] T. Verma and J. Pearl. Equivalence and synthesis of causal models. In M. Henrion, M. Shachter, R. Kanal, and J. Lemmer, editors, *Proceedings of the Sixth Conference on Uncertainty in Artificial Intelligence*, pages 220–227, 1990.

[16] T. Verma and J. Pearl. A theory of inferred causation. In J. Allen, R. Fikes, and E. Sandewall, editors, *Knowledge Representation and Reasoning: Proceedings of the Second International Conference*, pages 441–452. Morgan Kaufmann, New York, 1991.

[17] A. Wille and P. Bühlmann. Low-order conditional independence graphs for inferring genetic networks. *Statistical Applications in Genetics and Molecular Biology*, 5(1):1–32, 2006.