Order-Independent Structure Learning of Multivariate Regression Chain Graphs[⋆]

Mohammad Ali Javidian, Marco Valtorta, and Pooyan Jamshidi

University of South Carolina
javidian@email.sc.edu, mgv@cse.sc.edu, pjamshid@cse.sc.edu

Abstract. This paper deals with multivariate regression chain graphs (MVR CGs), which were introduced by Cox and Wermuth in the nineties to represent linear causal models with correlated errors. We consider the PC-like algorithm for structure learning of MVR CGs, a constraint-based method proposed by Sonntag and Peña in 2012. We show that the PC-like algorithm is order-dependent, because the output can depend on the order in which the variables are given. This order-dependence is a minor issue in low-dimensional settings. However, it can be very pronounced in high-dimensional settings, where it can lead to highly variable results. We propose two modifications of the PC-like algorithm that remove part or all of this order-dependence. Simulations under a variety of settings demonstrate the competitive performance of our algorithms in comparison with the original PC-like algorithm in low-dimensional settings and improved performance in high-dimensional settings.

Keywords: Multivariate regression chain graph · Structural learning · Order independence · High-dimensional data · Scalable machine learning techniques.

1   Introduction

Chain graphs were introduced by Lauritzen, Wermuth and Frydenberg [5],[9] as a generalization of graphs based on undirected graphs and directed acyclic graphs (DAGs). Later Andersson, Madigan and Perlman introduced an alternative Markov property for chain graphs [1]. In 1993 [3], Cox and Wermuth introduced multivariate regression chain graphs (MVR CGs). The different interpretations of CGs have different merits, but none of the interpretations subsumes another interpretation [4].

Acyclic directed mixed graphs (ADMGs), also known as semi-Markov(ian) [12] models contain directed ($\rightarrow$) and bidirected ($\leftrightarrow$) edges subject to the restriction that there are no directed cycles [15]. An ADMG that has no partially directed cycle is called a *multivariate regression chain graph*. Cox and Wermuth represented these graphs using directed edges and dashed edges, but we follow

---

Richardson [15] because bidirected edges allow the m-separation criterion (defined in section 2) to be viewed more directly as an extension of d-separation than is possible with dashed edges [15].

Unlike in the other CG interpretations, the bidirected edge in MVR CGs has a strong intuitive meaning. It can be seen to represent one or more hidden common causes between the variables connected by it. In other words, in an MVR CG any bidirected edge $X \leftrightarrow Y$ can be replaced by $X \leftarrow H \rightarrow Y$ to obtain a Bayesian network representing the same independence model over the original variables, i.e. excluding the new variables H. These variables are called hidden, or latent, and have been marginalized away in the CG model. See [17],[7], [18] for details on the properties of MVR chain graphs.

Two *constraint-based* learning algorithms, that use a statistical analysis to test the presence of a conditional independency, exist for learning MVR CGs: (1) the PC-like algorithm [16], and (2) the answer set programming (ASP) algorithm [13]. The PC-like algorithm extends the original learning algorithm for Bayesian networks by **P**eter Spirtes and **C**lark Glymour [19]. It learns the structure of the underlying MVR chain graph in four steps: (a) determining the skeleton: the resulting undirected graph in this phase contains an undirected edge $u - v$ iff there is no set $S \subseteq V \setminus \{u, v\}$ such that $u \perp\!\!\!\perp v|S$; (b) determining the v-structures (unshielded colliders); (c) orienting some of the undirected/directed edges into directed/bidirected edges according to a set of rules applied iteratively; (d) transforming the resulting graph in the previous step into an MVR CG. The essential recovery algorithm obtained after step (c) contains all directed and bidirected edges that are present in every MVR CG of the same Markov equivalence class.

In this paper, we show that the PC-like algorithm is order-dependent, in the sense that the output can depend on the order in which the variables are given. We propose several modifications of the PC-like algorithm that remove part or all of this order-dependence, but do not change the result when perfect conditional independence information is used. When applied to data, the modified algorithms are partly or fully order-independent. Proofs, implementations in R, and details of experimental results can be found in the supplementary material at https://github.com/majavid/SUM2019.

2   Definitions and Concepts

Below we briefly list some of the most important concepts used in this paper.

If there is an arrow from a pointing towards b, a is said to be a parent of b. The set of parents of b is denoted as $pa(b)$. If there is a bidirected edge between a and b, a and b are said to be neighbors. The set of neighbors of a vertex a is denoted as $ne(a)$. The expressions $pa(A)$ and $ne(A)$ denote the collection of parents and neighbors of vertices in A that are not themselves elements of A. The boundary $bd(A)$ of a subset A of vertices is the set of vertices in $V \setminus A$ that are parents or neighbors to vertices in A.

A path of length n from a to b is a sequence $a = a_0, \ldots, a_n = b$ of distinct vertices such that $(a_i \rightarrow a_{i+1}) \in E$, for all $i = 1, \ldots, n$. A chain of length n from

a to b is a sequence $a = a_0, \ldots, a_n = b$ of distinct vertices such that $(a_i \to a_{i+1}) \in E$, or $(a_{i+1} \to a_i) \in E$, or $(a_{i+1} \leftrightarrow a_i) \in E$, for all $i = 1, \ldots, n$. We say that u is an ancestor of v and v is a descendant of u if there is a path from u to v in G. The set of ancestors of v is denoted as $an(v)$, and we define $An(v) = an(v) \cup v$. We apply this definition to sets: $an(X) = \{\alpha | \alpha$ is an ancestor of β for some $\beta \in X\}$. A partially directed cycle in a graph G is a sequence of n distinct vertices $v_1, \ldots, v_n (n \geq 3)$,and $v_{n+1} \equiv v_1$, such that $\forall i (1 \leq i \leq n)$ either $v_i \leftrightarrow v_{i+1}$ or $v_i \to v_{i+1}$, and $\exists j (1 \leq j \leq n)$ such that $v_i \to v_{i+1}$.

A graph with only undirected edges is called an undirected graph (UG). A graph with only directed edges and without directed cycles is called a directed acyclic graph (DAG). Acyclic directed mixed graphs, also known as semi-Markov(ian) [12] models contain directed ($\to$) and bidirected ($\leftrightarrow$) edges subject to the restriction that there are no directed cycles [15]. A graph that has no partially directed cycles is called a *chain graph*.

A nonendpoint vertex ζ on a chain is a *collider* on the chain if the edges preceding and succeeding ζ on the chain have an arrowhead at ζ, that is, $\to \zeta \leftarrow$, $or \leftrightarrow \zeta \leftrightarrow, or \leftrightarrow \zeta \leftarrow, or \to \zeta \leftrightarrow$. A nonendpoint vertex ζ on a chain which is not a collider is a noncollider on the chain. A chain between vertices α and β in chain graph G is said to be m-connecting given a set Z (possibly empty), with $\alpha, \beta \notin Z$, if every noncollider on the path is not in Z, and every collider on the path is in $An_G(Z)$.

A chain that is not m-connecting given Z is said to be blocked given (or by) Z. If there is no chain m-connecting α and β given Z, then α and β are said to be m-separated given Z. Sets X and Y are m-separated given Z, if for every pair α, β, with $\alpha \in X$ and $\beta \in Y$, α and β are m-separated given Z (X, Y, and Z are disjoint sets; X, Y are nonempty). We denote the independence model resulting from applying the m-separation criterion to G, by $\Im_m(G)$. This is an extension of Pearl's d-separation criterion [11] to MVR chain graphs in that in a DAG D, a chain is d-connecting if and only if it is m-connecting.

We say that two MVR CGs G and H are Markov equivalent or that they are in the same Markov equivalence class iff $\Im_m(G) = \Im_m(H)$. If G and H have the same adjacencies and unshielded colliders, then $\Im_m(G) = \Im_m(H)$ [21].

Just like for many other probabilistic graphical models there might exist multiple MVR CGs that represent the same independence model. Sometimes it can however be desirable to have a unique graphical representation of the different representable independence models in the MVR CGs interpretation. A graph G^* is said to be the essential MVR CG of an MVR CG G if it has the same skeleton as G and contains all and only the arrowheads common to every MVR CG in the Markov equivalence class of G. One thing that can be noted here is that an essential MVR CG does not need to be a MVR CG. Instead these graphs can contain three types of edges, undirected, directed and bidirected [17].

3   Order-Dependent PC-like Algorithm

In this section, we show that the PC-like algorithm proposed by Sonntag and Peña in [16] is order-dependent, in the sense that the output can depend on the order in which the variables are given. The PC-like algorithm for learning MVR CGs under the faithfulness assumption is formally described in Algorithm 1.
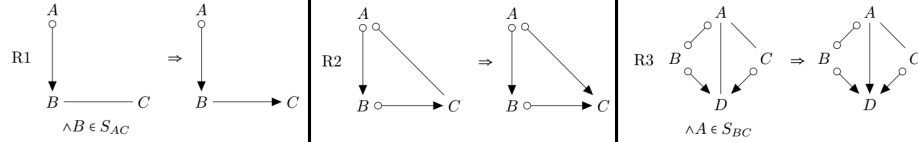
Fig. 1. The Rules [16]

In applications, we do not have perfect conditional independence information. Instead, we assume that we have an i.i.d. sample of size n of variables $V = (X_1, \ldots, Xp)$. In the PC-like algorithm [16] all conditional independence queries are estimated by statistical conditional independence tests at some pre-specified significance level (p value) α. For example, if the distribution of V is multivariate Gaussian, one can test for zero partial correlation, see, e.g., [8]. For this purpose, we use the gaussCItest() function from the R package pcalg throughout this paper. Let order(V) denote an ordering on the variables in V. We now consider the role of order(V) in every step of the algorithm.

In the skeleton recovery phase of the PC-like algorithm [16], the order of variables affects the estimation of the skeleton and the separating sets. In particular, as noted for the special case of Bayesian networks in [2], for each level of i, the order of variables determines the order in which pairs of adjacent vertices and subsets S of their adjacency sets are considered (see lines 4 and 5 in Algorithm 1). The skeleton H is updated after each edge removal. Hence, the adjacency sets typically change within one level of i, and this affects which other conditional independencies are checked, since the algorithm only conditions on subsets of the adjacency sets. When we have perfect conditional independence information, all orderings on the variables lead to the same output. In the sample version, however, we typically make mistakes in keeping or removing edges, because conditional independence relationships have to be estimated from data. In such cases, the resulting changes in the adjacency sets can lead to different skeletons, as illustrated in Example 1.

Moreover, different variable orderings can lead to different separating sets in the skeleton recovery phase. When we have perfect conditional independence information, this is not important, because any valid separating set leads to the correct v-structure decision in the orientation phase. In the sample version, however, different separating sets in the skeleton recovery phase of the algorithm may yield different decisions about v-structures in the orientation phase. This is illustrated in Example 2.

Algorithm 1: The order-dependent PC-like algorithm for learning MVR chain graphs [16]

Input: A set V of nodes and a probability distribution p faithful to an unknown MVR CG G and an ordering order(V) on the variables.

Output: An MVR CG G' s.t. G and G' are Markov equivalent and G' has exactly the minimum set of bidirected edges for its equivalence class.

1 Let H denote the complete undirected graph over V;

/* Skeleton Recovery   */

2 **for** $i \leftarrow 0$ **to** $|V_H| - 2$ **do**

3   **while** *possible* **do**

4   Select any ordered pair of nodes u and v in H such that $u \in ad_H(v)$ and $|ad_H(u) \setminus v| \geq i$ using order(V);

/* $ad_H(x) := \{y \in V | x \longrightarrow y, y \longrightarrow x, \text{ or } x \text{———} y\}$ */

5   **if** *there exists* $S \subseteq (ad_H(u) \setminus v)$ *s.t.* $|S| = i$ *and* $u \perp\!\!\!\perp_p v | S$ *(i.e., u is independent of v given S in the probability distribution p)* **then**

6   Set $S_{uv} = S_{vu} = S$;

7   Remove the edge $u \text{———} v$ from H;

8   **end**

9   **end**

10 **end**

/* v-structure Recovery   */

11 **for** *each m-separator* S_{uv} **do**

12   **if** $u \circ\!\!-\!\!- w -\!\!\!-\circ v$ *appears in the skeleton and w is not in S_{uv}* **then**

/* $u \circ\!\!-\!\!- w$ means $u \longleftarrow w$ or $u \text{———} w$. Also, $w -\!\!\!-\circ v$ means $w \longrightarrow v$ or $w \text{———} v$.   */

13   Determine a v-structure $u \circ\!\!\rightarrow w \leftarrow\!\!\circ v$;

14   **end**

15 **end**

16 Apply rules 1-3 in Figure 1 while possible;

/* After this line, the learned graph is the *essential graph* of MVR CG G.   */

17 Let G'_u be the subgraph of G' containing only the nodes and the undirected edges in G';

18 Let T be the junction tree of G'_u;

/* If G'_u is disconnected, the cliques belonging to different connected components can be linked with empty separators, as described in [6, Theorem 4.8].   */

19 Order the cliques $C_1, \cdots, C_n$ of G'_u s.t. C_1 is the root of T and if C_i is closer to the root than C_j in T then $C_i < C_j$;

20 Order the nodes such that if $A \in C_i$, $B \in C_j$, and $C_i < C_j$ then $A < B$;

**21** Orient the undirected edges in G' according to the ordering obtained in line 21.

Finally, we consider the role of order(V) on the orientation rules in the essential graph recovery phase of the sample version of the PC-like algorithm. Example 3 illustrates that different variable orderings can lead to different orientations, even if the skeleton and separating sets are order-independent.

Example 1 (Order-dependent skeleton of the PC-like algorithm.). Suppose that the distribution of $V = \{a, b, c, d, e\}$ is faithful to the DAG in Figure 2(a). This DAG encodes the following conditional independencies (using the notation defined in line 5 of Algorithm 1) with minimal separating sets: $a \perp\!\!\!\perp d | \{b, c\}$ and $a \perp\!\!\!\perp e | \{b, c\}$.

Suppose that we have an i.i.d. sample of (a, b, c, d, e), and that the following conditional independencies with minimal separating sets are judged to hold at some significance level α: $a \perp\!\!\!\perp d | \{b, c\}$, $a \perp\!\!\!\perp e | \{b, c, d\}$, and $c \perp\!\!\!\perp e | \{a, b, d\}$. Thus, the first two are correct, while the third is false.

We now apply the skeleton recovery phase of the PC-like algorithm with two different orderings: $\mathrm{order}_1(V) = (d, e, a, c, b)$ and $\mathrm{order}_2(V) = (d, c, e, a, b)$. The resulting skeletons are shown in Figures 2(b) and 2(c), respectively.

(a)   (b)   (c)
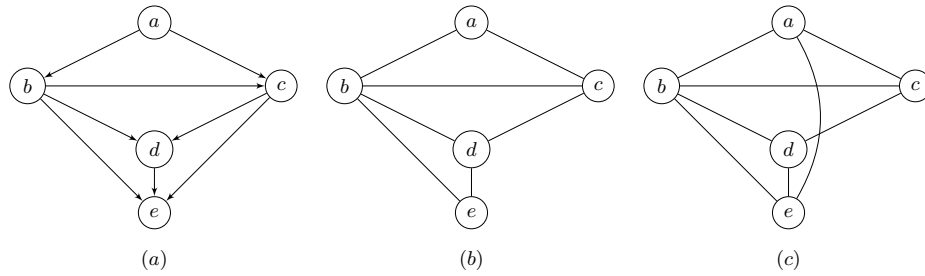
Fig. 2. (a) The DAG G, (b) the skeleton returned by Algorithm 1 with $\mathrm{order}_1(V)$, (c) the skeleton returned by Algorithm 1 with $\mathrm{order}_2(V)$.

We see that the skeletons are different, and that both are incorrect as the edge $c - e$ is missing. The skeleton for $\mathrm{order}_2(V)$ contains an additional error, as there is an additional edge $a - e$. We now go through Algorithm 1 to see what happened. We start with a complete undirected graph on V. When $i = 0$, variables are tested for marginal independence, and the algorithm correctly does not remove any edge. Also, when $i = 1$, the algorithm correctly does not remove any edge. When $i = 2$, there is a pair of vertices that is thought to be conditionally independent given a subset of size two, and the algorithm correctly removes the edge between a and d. When $i = 3$, there are two pairs of vertices that are thought to be conditionally independent given a subset of size three. Table 1 shows the trace table of Algorithm 1 for $i = 3$ and $\mathrm{order}_1(V) = (d, e, a, c, b)$.

Table 1. The trace table of Algorithm 1 for $i = 3$ and $\mathrm{order}_1(V) = (d, e, a, c, b)$.

| Ordered Pair (u,v)	$ad_H(u)$	S_{uv}	Is $S_{uv} \subseteq ad_H(u) \setminus \{v\}$?	Is $u - v$ removed?
(e, a)	$\{a, b, c, d\}$	$\{b, c, d\}$	Yes	Yes
(e, c)	$\{b, c, d\}$	$\{a, b, d\}$	No	No
(c, e)	$\{a, b, d, e\}$	$\{a, b, d\}$	Yes	Yes

Table 2 shows the trace table of Algorithm 1 for $i = 3$ and $\mathrm{order}_2(V) = (d, c, e, a, b)$.

Table 2. The trace table of Algorithm 1 for $i = 3$ and $\text{order}_2(V) = (d, c, e, a, b)$.

Ordered Pair (u, v)	$ad_H(u)$	S_{uv}	Is $S_{uv} \subseteq ad_H(u) \setminus \{v\}$?	Is $u \text{---} v$ removed?
(c, e)	$\{a, b, d, e\}$	$\{a, b, d\}$	Yes	Yes
(e, a)	$\{a, b, d\}$	$\{b, c, d\}$	No	No
(a, e)	$\{b, c, e\}$	$\{b, c, d\}$	No	No

Example 2 (Order-dependent separating sets and v-structures of the PC-like algorithm.). Suppose that the distribution of $V = \{a, b, c, d, e\}$ is faithful to the DAG in Figure 3(a). This DAG encodes the following conditional independencies with minimal separating sets: $a \perp\!\!\!\perp d|b, a \perp\!\!\!\perp e|\{b, c\}, a \perp\!\!\!\perp e|\{c, d\}, b \perp\!\!\!\perp c, b \perp\!\!\!\perp e|d$, and $c \perp\!\!\!\perp d$.

Suppose that we have an i.i.d. sample of (a, b, c, d, e). Assume that all true conditional independencies are judged to hold except $c \perp\!\!\!\perp d$. Suppose that $c \perp\!\!\!\perp d|b$ and $c \perp\!\!\!\perp d|e$ are thought to hold. Thus, the first is correct, while the second is false. We now apply the v-structure recovery phase of the PC-like algorithm with two different orderings: $\text{order}_1(V) = (d, c, b, a, e)$ and $\text{order}_3(V) = (c, d, e, a, b)$. The resulting CGs are shown in Figures 3(b) and 3(c), respectively. Note that while the separating set for vertices c and d with $\text{order}_1(V)$ is $S_{dc} = S_{cd} = \{b\}$, the separating set for them with $\text{order}_2(V)$ is $S_{cd} = S_{dc} = \{e\}$.
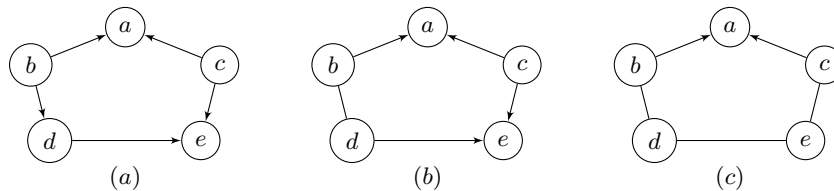
Fig. 3. (a) The DAG G, (b) the CG returned after the v-structure recovery phase of Algorithm 1 with $\text{order}_1(V)$, (c) the CG returned after the v-structure recovery phase of Algorithm 1 with $\text{order}_3(V)$.

This illustrates that order-dependent separating sets in the skeleton recovery phase of the sample version of the PC-algorithm can lead to order-dependent v-structures.

Example 3 (Order-dependent orientation rules of the PC-like algorithm.). Consider the graph in Figure 4, and assume that this is the output of the sample version of the PC-like algorithm after v-structure recovery. Also, consider that $c \in S_{a,d}$ and $d \in S_{b,f}$. Thus, we have two v-structures, namely $a \longrightarrow c \longleftarrow e$ and $b \longrightarrow d \longleftarrow f$, and four unshielded triples, namely $(e, c, d), (c, d, f), (a, c, d)$, and (b, d, c). Thus, we then apply the orientation rules in the essential recovery phase of the algorithm, starting with rule R1. If one of the two unshielded triples (e, c, d) or (a, c, d) is considered first, we obtain $c \longrightarrow d$. On the other hand, if one of the unshielded triples (b, d, c) or (c, d, f) is considered first, then we obtain $c \longleftarrow d$. Note that we have no issues with overwriting of edges here, since as soon as the edge $c \text{---} d$ is oriented, all edges are oriented and no further orientation

rules are applied. These examples illustrate that the essential graph recovery phase of the PC-like algorithm can be order-dependent regardless of the output of the previous steps.
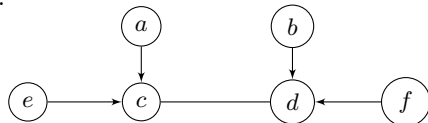
Fig. 4. Possible mixed graph after v-structure recovery phase of the sample version of the PC-like algorithm.

4   Order Independent Algorithms for Learning MVR CGs

We now propose several modifications of the original PC-like algorithm (and hence also of the related algorithms) that remove the order-dependence in the various stages of the algorithm, analogously to what Colombo and Maathuis [2] did for the original PC algorithm in the case of DAGs. For this purpose, we discuss the skeleton, v-structures, and the orientation rules, respectively.

4.1   Order-Independent Skeleton Recovery

We first consider estimation of the skeleton in the adjacency search of the PC-like algorithm. The pseudocode for our modification is given in Algorithm 2. The resulting PC-like algorithm in Algorithm 2 is called *stable PC-like*.

   The main difference between Algorithms 1 and 2 is given by the for-loop on lines 3-5 in the latter one, which computes and stores the adjacency sets $a_H(v_i)$ of all variables after each new size i of the conditioning sets. These stored adjacency sets $a_H(v_i)$ are used whenever we search for conditioning sets of this given size i. Consequently, an edge deletion on line 10 no longer affects which conditional independencies are checked for other pairs of variables at this level of i.

   In other words, at each level of i, Algorithm 2 records which edges should be removed, but for the purpose of the adjacency sets it removes these edges only when it goes to the next value of i. Besides resolving the order-dependence in the estimation of the skeleton, our algorithm has the advantage that it is easily parallelizable at each level of i. The stable PC-like algorithm is correct, i.e. it returns an MVR CG to which the given probability distribution is faithful (Theorem 1), and it yields order-independent skeletons in the sample version (Theorem 2). We illustrate the algorithm in Example 4.

Theorem 1. *Let the distribution of V be faithful to an MVR CG G, and assume that we are given perfect conditional independence information about all pairs of variables (u,v) in V given subsets $S \subseteq V \setminus \{u,v\}$. Then the output of the stable PC-like algorithm is an MVR CG that has exactly the minimum set of bidirected edges for its equivalence class.*

Theorem 2. *The skeleton resulting from the sample version of the stable PC-like algorithm is order-independent.*

Example 4 (Order-independent skeletons). We go back to Example 1, and consider the sample version of Algorithm 2. The algorithm now outputs the skeleton shown in Figure 2(b) for both orderings $\text{order}_1(V)$ and $\text{order}_2(V)$. We again go through the algorithm step by step. We start with a complete undirected graph on V. No conditional independence found when $i = 0$. Also, when $i = 1$, the algorithm correctly does not remove any edge. When $i = 2$, the algorithm first computes the new adjacency sets: $a_H(v) = V \setminus \{v\}, \forall v \in V$. There is a pair of variables that is thought to be conditionally independent given a subset of size two, namely (a, d). Since the sets $a_H(v)$ are not updated after edge removals, it does not matter in which order we consider the ordered pair. Any ordering leads to the removal of edge between a and d. When $i = 3$, the algorithm first computes the new adjacency sets: $a_H(a) = a_H(d) = \{b, c, e\}$ and $a_H(v) = V \setminus \{v\}$, for $v = b, c, e$. There are two pairs of variables that are thought to be conditionally independent given a subset of size three, namely (a, e) and (c, e). Since the sets $a_H(v)$ are not updated after edge removals, it does not matter in which order we consider the ordered pair. Any ordering leads to the removal of both edges $a \relbar\joinrel\relbar e$ and $c \relbar\joinrel\relbar e$.

Algorithm 2: The order-independent (stable) PC-like algorithm for learning MVR chain graphs.

Input: A set V of nodes and a probability distribution p faithful to an unknown MVR CG G and an ordering $\text{order}(V)$ on the variables.
Output: An MVR CG G' s.t. G and G' are Markov equivalent and G' has exactly the minimum set of bidirected edges for its equivalence class.

1 Let H denote the complete undirected graph over $V = \{v_1, \ldots, v_n\}$;
    /* Skeleton Recovery   */
2 **for** $i \leftarrow 0$ **to** $|V_H| - 2$ **do**
3   **for** $j \leftarrow 1$ **to** $|V_H|$ **do**
4   Set $a_H(v_i) = ad_H(v_i)$;
5   **end**
6   **while** *possible* **do**
7   Select any ordered pair of nodes u and v in H such that $u \in a_H(v)$ and $|a_H(u) \setminus v| \geq i$ using $\text{order}(V)$;
8   **if** *there exists $S \subseteq (a_H(u) \setminus v)$ s.t. $|S| = i$ and $u \perp\!\!\!\perp_p v|S$ (i.e., u is independent of v given S in the probability distribution p)* **then**
9   Set $S_{uv} = S_{vu} = S$;
10   Remove the edge $u \relbar\joinrel\relbar v$ from H;
11   **end**
12   **end**
13 **end**
    /* v-structure Recovery and orientation rules   */
14 Follow the same procedures in Algorithm 1 (lines: 11-21).

4.2 Order-Independent v-structures Recovery

We propose two methods to resolve the order-dependence in the determination of the v-structures, using the conservative PC algorithm (CPC) of Ramsey et al. [14] and the majority rule PC-like algorithm (MPC) of Colombo & Maathuis [2].

The **Conservative PC-like algorithm (CPC-like algorithm)** works as follows. Let H be the undirected graph resulting from the skeleton recovery phase of the PC-like algorithm (Algorithm 1). For all unshielded triples (X_i, X_j, X_k) in H, determine all subsets S of $ad_H(X_i)$ and of $ad_H(X_k)$ that make X_i and X_k conditionally independent, i.e., that satisfy $X_i \perp\!\!\!\perp_p X_k | S$. We refer to such sets as separating sets. The triple (X_i, X_j, X_k) is labelled as *unambiguous* if at least one such separating set is found and either X_j is in all separating sets or in none of them; otherwise it is labelled as *ambiguous*. If the triple is unambiguous, it is oriented as v-structure if and only if X_j is in none of the separating sets. Moreover, in the v-structure recovery phase of the PC-like algorithm (Algorithm 1, lines 11-15), the orientation rules are adapted so that only unambiguous triples are oriented. The output of the CPC-like algorithm is a mixed graph in which ambiguous triples are marked. We refer to the combination of the stable PC-like and CPC-like algorithms as the *stable CPC-like algorithm*.

In the case of DAGs, Colombo and Maathuis [2] found that the CPC-algorithm can be very conservative, in the sense that very few unshielded triples are unambiguous in the sample version, where conditional independence relationships have to be estimated from data. They proposed a minor modification of the CPC approach, called *Majority rule PC algorithm (MPC)* to mitigate the (unnecessary) severity of CPC-like approach. We similarly propose the **Majority rule PC-like algorithm (MPC-like)** for MVR CGs. As in the CPC-like algorithm, we first determine all subsets S of $ad_H(X_i)$ and of $ad_H(X_k)$ that make X_i and X_k conditionally independent, i.e., that satisfy $X_i \perp\!\!\!\perp_p X_k | S$. The triple (X_i, X_j, X_k) is labelled as *(α, β)-unambiguous* if at least one such separating set is found or X_j is in no more than $\alpha\%$ or no less than $\beta\%$ of the separating sets, for $0 \leq \alpha \leq \beta \leq 100$. Otherwise it is labelled as *ambiguous*. (As an example, consider $\alpha = 30$ and $\beta = 60$.) If a triple is unambiguous, it is oriented as a v-structure if and only if X_j is in less than $\alpha\%$ of the separating sets. As in the CPC-like algorithm, the orientation rules in the v-structure recovery phase of the PC-like algorithm (Algorithm 1, lines 11-15) are adapted so that only unambiguous triples are oriented, and the output is a mixed graph in which ambiguous triples are marked. Note that the CPC-like algorithm is the special case of the MPC-like algorithm with $\alpha = 0$ and $\beta = 100$. We refer to the combination of the stable PC-like and MPC-like algorithms as the *stable MPC-like algorithm*.

Theorem 3. *Let the distribution of V be faithful to an MVR CG G, and assume that we are given perfect conditional independence information about all pairs of variables (u, v) in V given subsets $S \subseteq V \setminus \{u, v\}$. Then the output of the (stable) CPC/MPC-like algorithm is an MVR CG that is Markov equivalent with G that has exactly the minimum set of bidirected edges for its equivalence class.*

Theorem 4. *The decisions about v-structures in the sample version of the stable CPC/MPC-like algorithm is order-independent.*

Example 5 (Order-independent decisions about v-structures). We consider the sample versions of the stable CPC/MPC-like algorithm, using the same input as in Example 2. In particular, we assume that all conditional independencies induced by the MVR CG in Figure 3(a) are judged to hold except $c \perp\!\!\!\perp d$. Suppose that $c \perp\!\!\!\perp d | b$ and $c \perp\!\!\!\perp d | e$ are thought to hold. Let $\alpha = \beta = 50$.

Denote the skeleton after the skeleton recovery phase by H. We consider the unshielded triple (c, e, d). First, we compute $a_H(c) = \{a, d, e\}$ and $a_H(d) = \{a, b, c, e\}$, when $i = 1$. We now consider all subsets S of these adjacency sets, and check whether $c \perp\!\!\!\perp d | S$. The following separating sets are found: $\{b\}, \{e\}$, and $\{b, e\}$. Since e is in some but not all of these separating sets, the stable CPC-like algorithm determines that the triple is ambiguous, and no orientations are performed. Since e is in more than half of the separating sets, stable MPC-like determines that the triple is unambiguous and not a v-structure. The output of both algorithms is given in Figure 3(c).

At this point it should be clear why the modified PC-like algorithm is labeled "conservative": it is more cautious than the (stable) PC-like algorithm in drawing unambiguous conclusions about orientations. As we showed in Example 5, the output of the (stable) CPC-like algorithm may not be collider equivalent with the true MVR CG G, if the resulting CG contains an ambiguous triple.

4.3 Order-Independent Orientation Rules

Even when the skeleton and the determination of the v-structures are order-independent, Example 3 showed that there might be some order-dependent steps left in the sample version. Regarding the orientation rules, we note that the PC-like algorithm does not suffer from conflicting v-structures (as shown in [2] for the PC-algorithm in the case of DAGs), because bi-directed edges are allowed. However, the three orientation rules still suffer from order-dependence issues (see Example 3 and Figure 4). To solve this problem, we can use lists of candidate edges for each orientation rule as follows: we first generate a list of all edges that can be oriented by rule R1. We orient all these edges, creating bi-directed edges if there are conflicts. We do the same for rules R2 and R3, and iterate this procedure until no more edges can be oriented.

When using this procedure, we add the letter L (standing for lists), e.g., (stable) LCPC-like and (stable) LMPC-like. The (stable) LCPC-like and (stable) LMPC-like algorithms are fully order-independent in the sample versions. The procedure is illustrated in Example 6.

Theorem 5. *Let the distribution of V be faithful to an MVR CG G, and assume that we are given perfect conditional independence information about all pairs of variables (u, v) in V given subsets $S \subseteq V \setminus \{u, v\}$. Then the output of the (stable) LCPC/LMPC-like algorithm is an MVR CG that is Markov equivalent with G that has exactly the minimum set of bidirected edges for its equivalence class.*

Theorem 6. *The sample versions of stable CPC-like and stable MPC-like algorithms are fully order-independent.*

Example 6. Consider the structure shown in Figure 4. As a first step, we construct a list containing all candidate structures eligible for orientation rule R1 in the phase of the essential graph recovery. The list contains the unshielded triples $(e, c, d), (c, d, f), (a, c, d)$, and (b, d, c). Now, we go through each element in the list and we orient the edges accordingly, allowing bi-directed edges. This yields the edge orientation $c \longleftrightarrow d$, regardless of the ordering of the variables.

5   Evaluation

In this section, we compare the performance of our algorithms (Table 3) with the original PC-like learning algorithm by running them on randomly generated MVR chain graphs in low-dimensional and high-dimensional data, respectively. We report on the Gaussian case only because of space limitations.

We evaluate the performance of the proposed algorithms in terms of the six measurements that are commonly used [2,8,10,20] for constraint-based learning algorithms: (a) the true positive rate (TPR) (also known as sensitivity, recall, and hit rate), (b) the false positive rate (FPR) (also known as fall-out), (c) the true discovery rate (TDR) (also known as precision or positive predictive value), (d) accuracy (ACC) for the skeleton, (e) the structural Hamming distance (SHD) (this is the metric described in [20] to compare the structure of the learned and the original graphs), and (f) run-time for the LCG recovery algorithms. In principle, large values of TPR, TDR, and ACC, and small values of FPR and SHD indicate good performance. All of these six measurements are computed on the essential graphs of the CGs, rather than the CGs directly, to avoid spurious differences due to random orientation of undirected edges.

Table 3. Order-dependence issues and corresponding modifications of the PC-like algorithm that remove the problem. "Yes" indicates that the corresponding aspect of the graph is estimated order-independently in the sample version.

	skeleton	v-structures decisions	edges orientations
PC-like	No	No	No
stable PC-like	Yes	No	No
stable CPC/MPC-like	Yes	Yes	No
stable LCPC/LMPC-like	Yes	Yes	Yes

Figure 5 shows that: (a) as we expected [10,8], all algorithms work well on sparse graphs ($N = 2$), (b) for all algorithms, typically the TPR, TDR, and ACC increase with sample size, (c) for all algorithms, typically the SHD and FPR decrease with sample size, (d) a large significance level ($\alpha = 0.05$) typically yields large TPR, FPR, and SHD, (e) while the stable PC-like algorithm has a better TDR and FPR in comparison with the original PC-like algorithm, the original PC-like algorithm has a better TPR (as observed in the case of DAGs [2]). This can be explained by the fact that the stable PC-like algorithm tends to perform more tests than the original PC-like algorithm, and (h) while the original PC-like algorithm has a (slightly) better SHD in comparison with the stable PC-like algorithm in low-dimensional data, the stable PC-like algorithm has a better SHD in high-dimensional data. Also, (very) small variances indicate that the order-independent versions of the PC-like algorithm in high-dimensional data are stable. When considering average running times versus sample sizes, as shown in Figure 5, we observe that: (a) the average run time increases when sample size increases; (b) generally, the average run time for the original PC-like algorithm is (slightly) better than that for the stable PC-like algorithm in both low and high dimensional settings.
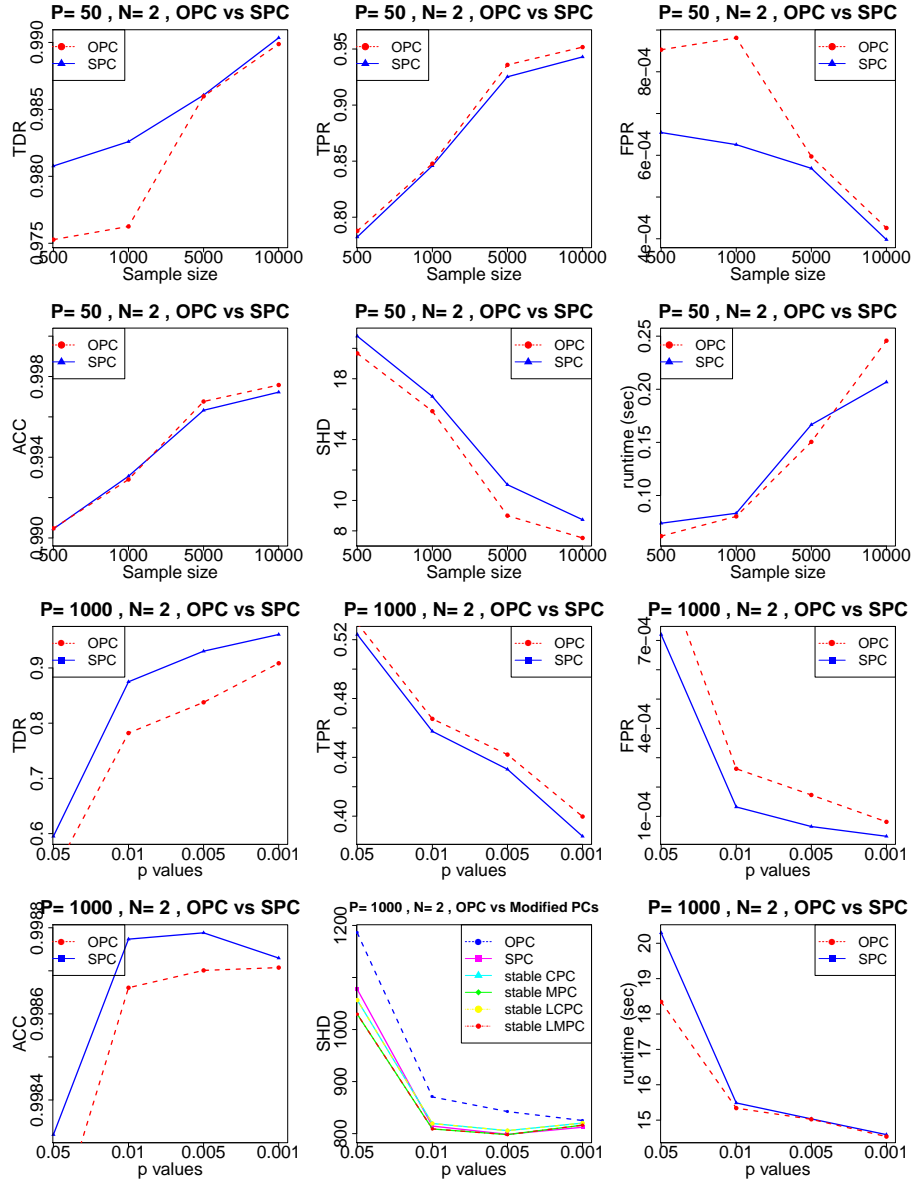
Fig. 5. The first two rows show the performance of the original (OPC) and stable PC-like (SPC) algorithms for randomly generated Gaussian chain graph models: average over 30 repetitions with 50 variables correspond to N = 2, and the significance level $\alpha = 0.001$. The last two rows show the performance of the original (OPC) and stable PC-like (SPC) algorithms for randomly generated Gaussian chain graph models: average over 30 repetitions with 1000 variables correspond to N = 2, sample size S=50, and the significance level $\alpha = 0.05, 0.01, 0.005, 0.001$.

In summary, empirical simulations show that our algorithms achieve competitive results with the original PC-like learning algorithm; in particular, in the Gaussian case the order-independent algorithms achieve output of better quality than the original PC-like algorithm, especially in high-dimensional settings. Since we know of no score-based learning algorithms for MVR chain graphs (and, in fact, for any kind of chain graphs), we plan to investigate the feasibility of a scalable algorithm of this kind.

References

1. Andersson, S.A., Madigan, D., Perlman, M.D.: An alternative Markov property for chain graphs. In: Proceedings of UAI conference. pp. 40–48 (1996)
2. Colombo, D., Maathuis, M.H.: Order-independent constraint-based causal structure learning. The Journal of Machine Learning Research **15**(1), 3741–3782 (2014)
3. Cox, D.R., Wermuth, N.: Linear dependencies represented by chain graphs. Statistical Science **8**(3), 204–218 (1993)
4. Drton, M.: Discrete chain graph models. Bernoulli **15**(3), 736–753 (2009)
5. Frydenberg, M.: The chain graph markov property. Scandinavian Journal of Statistics **17**(4), 333–353 (1990)
6. Golumbic, M.: Algorithmic Graph Theory and Perfect Graphs. Academic P (1980)
7. Javidian, M.A., Valtorta, M.: On the properties of MVR chain graphs. In: Workshop proceedings of PGM Conference. pp. 13–24 (2018)
8. Kalisch, M., Bühlmann, P.: Estimating high-dimensional directed acyclic graphs with the pc-algorithm. J. Mach. Learn. Res. **8**, 613–636 (2007)
9. Lauritzen, S., Wermuth, N.: Graphical models for associations between variables, some of which are qualitative and some quantitative. The Annals of Statistics **17**(1), 31–57 (1989)
10. Ma, Z., Xie, X., Geng, Z.: Structural learning of chain graphs via decomposition. Journal of Machine Learning Research **9**, 2847–2880 (2008)
11. Pearl, J.: Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference. Morgan Kaufmann Publishers Inc. San Francisco, CA, USA (1988)
12. Pearl, J.: Causality. Models, reasoning, and inference. Cambridge UP (2009)
13. Peña, J.M.: Alternative markov and causal properties for acyclic directed mixed graphs. In: Proceedings of UAI Conference. pp. 577–586 (2016)
14. Ramsey, J., Spirtes, P., Zhang, J.: Adjacency-faithfulness and conservative causal inference. In: Proceedings of UAI Conference. pp. 401–408 (2006)
15. Richardson, T.S.: Markov properties for acyclic directed mixed graphs. Scandinavian Journal of Statistics **30**(1), 145–157 (2003)
16. Sonntag, D., Peña, J.M.: Learning multivariate regression chain graphs under faithfulness. Proceedings of PGM Workshop pp. 299–306 (2012)
17. Sonntag, D., Peña, J.M.: Chain graph interpretations and their relations revisited. International Journal of Approximate Reasoning **58**, 39 – 56 (2015)
18. Sonntag, D., Pea, J.M., Gómez-Olmedo, M.: Approximate counting of graphical models via mcmc revisited. International Journal of Intelligent Systems **30**(3), 384–420 (2015)
19. Spirtes, P., Glymour, C., Scheines, R.: Causation, Prediction and Search, second ed. MIT Press, Cambridge, MA. (2000)
20. Tsamardinos, I., , Brown, L.E., Aliferis, C.F.: The max-min hill-climbing bayesian network structure learning algorithm. Machine Learning **65**(1), 31–78 (Oct 2006)
21. Wermuth, N., Sadeghi, K.: Sequences of regressions and their independences. Test **21**, 215–252 (2012)