# An Order-Independent Algorithm for Learning Chain Graphs

**Mohammad Ali Javidian**[⋆], **Marco Valtorta**[†], **and Pooyan Jamshidi**[†]
[⋆]School of Electrical and Computer Engineering, Purdue University
[†]Department of Computer Science and Engineering, University of South Carolina
mjavidia@purdue.edu, {mgv, pjamshid}@cse.sc.edu

## Abstract

LWF chain graphs combine directed acyclic graphs and undirected graphs. We propose a PC-like algorithm, called **PC4LWF**, that finds the structure of chain graphs under the faithfulness assumption to resolve the problem of scalability of the proposed algorithm by Studený (1997). We prove that **PC4LWF** is order dependent, in the sense that the output can depend on the order in which the variables are given. This order dependence can be very pronounced in high-dimensional settings. We propose two modifications of the **PC4LWF** algorithm that remove part or all of this order dependence. Simulation results with different sample sizes, network sizes, and p-values demonstrate the competitive performance of the **PC4LWF** algorithms in comparison with the LCD algorithm proposed by Ma et al. (2008) in low-dimensional settings and improved performance (with regard to error measures) in high-dimensional settings.

## Introduction

Currently systems containing both causal and non-causal relationships are mostly modeled with *directed acyclic graphs* (DAGs). Chain graphs (CGs) are a type of mixed graphs, admitting both directed and undirected edges, which contain no partially directed cycles. So, CGs may contain two types of edges, the directed type that corresponds to the causal relationship in DAGs and a second type of edge representing a symmetric relationship (Sonntag et al. 2015). *LWF Chain graphs* were introduced by **L**auritzen, **W**ermuth and **F**rydenberg (Frydenberg 1990), (Lauritzen and Wermuth 1989) as a generalization of graphical models based on undirected graphs and DAGs. From the *causality* point of view, in an LWF CG directed edges represent *direct causal effect*s, and undirected edges represent causal effects due to *interference* (Bhattacharya, Malinsky, and Shpitser 2019).

One important aspect of Probabilistic Graphical Models (PGMs), such as DAGs and CGs, is the possibility of learning the structure of models directly from sampled data. Six *constraint-based* algorithms, that use a statistical analysis to test the presence of a conditional independency, exist for learning LWF CGs: (1) the inductive causation like (IC-like) algorithm (Studený 1997), (2) the decomposition-based (LCD) algorithm (Ma, Xie, and Geng 2008), (3) the answer set programming (ASP) algorithm (Sonntag et al. 2015), (4) the inclusion optimal (CKES) algorithm (Peña, Sonntag, and Nielsen 2014), (5) the local structure learning of chain graphs with false discovery rate control (Wang, Liu, and Zhu 2019), and (6) the Markov blanket discovery (MbLWF) algorithm (Javidian, Valtorta, and Jamshidi 2020a).

Similar to the *inductive causation (IC)* algorithm (Verma and Pearl 1991), the IC-like algorithm (Studený 1997) cannot be applied to large numbers of variables because for testing whether there is a set separating $X$ and $Y$ in the skeleton recovery, the IC-like algorithm might search all $2^{n-2}$ subsets of all $n$ random variables not including $X$ and $Y$. In order to overcome the scalability of the IC-like algorithm, we propose a constraint-based method for learning the structural of chain graphs based on the idea of the *PC algorithm* (Spirtes, Glymour, and Scheines 2000), which is used for learning the structure of Bayesian networks (BNs). Our method modifies the IC-like algorithm to make it computationally feasible in the phase of skeleton recovery and to avoid the time consuming procedure of complex recovery. We prove that our proposed PC-like algorithm, called **PC4LWF**, is *order dependent*, in the sense that the output can depend on the order in which the variables are given. The order dependency can become very problematic for high-dimensional data, leading to highly variable results for different variable orderings (Colombo and Maathuis 2014). We propose several modifications of the **PC4LWF** 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.

Our proposed algorithm, called **STABLE-PC4LWF** (Stable PC-like for LWF CGs), similarly to the LCD algorithm, is able to exploit parallel computations for scaling up the task of learning LWF CGs. This will enable effective LWF CG discovery on large/high-dimensional datasets. In fact, *lower complexity, higher power of computational independence tests, better learned structure quality*, and the *ability of exploiting parallel computing* make our proposed algorithm more desirable and suitable for big data analysis when LWF CGs are being used. Code for reproducing our results is available at `https://github.com/majavid/PC4LWF2020`. Our main contributions are the following:

**(1)** We propose a PC-like algorithm, called **PC4LWF**, for learning the structure of LWF CGs under the faithfulness

assumption that includes two main procedures: (i) a feasible PC-like method for learning CG skeleton, (ii) a polynomial time procedure for complex recovery.
**(2)** We show that our **PC4LWF** algorithm is order dependent. Then, we propose two modifications of this algorithm that remove part or all of this order dependence.
**(3)** We experimentally compare the performance of our proposed PC-like algorithms with the LCD algorithm in the evaluation section and show that the PC-like algorithms are comparable to the LCD algorithm in low-dimensional settings and superior in high-dimensional settings in terms of error measures and runtime.
**(4)** We release supplementary material including data and an R package that implements the proposed algorithms.

## Definitions and Concepts

We refer the reader to (Javidian, Valtorta, and Jamshidi 2020b) for most (rather standard) definitions of graphical notions. Below are some of the central concepts used in this paper.

An *LWF chain graph* is a graph in which there are no partially directed cycles. The chain components $\mathcal{T}$ of a chain graph are the connected components of the undirected graph obtained by removing all directed edges from the chain graph. A *minimal complex* (or simply a complex or a *U*-structure) in a chain graph is an induced subgraph of the form $a \rightarrow v_1 - \cdots - v_r \leftarrow b$. The *skeleton* (underlying graph) of an LWF CG $G$ is obtained from $G$ by changing all directed edges of $G$ into undirected edges. For a CG $G$ we define its *moral graph* $G^m$ as the undirected graph with the same vertex set but with $\alpha$ and $\beta$ adjacent in $G^m$ if and only if either $\alpha \rightarrow \beta$, or $\alpha - \beta$, or $\beta \rightarrow \alpha$ or if there are $\gamma_1, \gamma_2$ in the same chain component such that $\alpha \rightarrow \gamma_1$ and $\beta \rightarrow \gamma_2$.

## PC4LWF: a PC-Like Algorithm for Learning LWF Chain Graphs

In this section, we discuss how the IC-like algorithm (Studený 1997) can be modified to obtain a computationally feasible algorithm for LWF CGs recovery. A brief review of the IC-like algorithm is presented first, then we present a PC-like algorithm, called **PC4LWF**, which is a constraint-based algorithm that learns a chain graph from a probability distribution faithful to some CG.

The IC-like algorithm (Studený 1997) is a constraint-based algorithm proposed for LWF CGs and is based on three sequential phases. The first phase finds the adjacencies (skeleton recovery), the second phase orients the edges that must be oriented the same in every CG in the Markov equivalence class (complex recovery), and the third phase transforms this graph into the largest CG (LCG recovery). The skeleton recovery of the IC-like algorithm works as follows: construct an undirected graph $H$ such that vertices $u$ and $v$ are connected with an undirected edge if and only if no set $S_{uv}$ can be found such that $u \perp\!\!\!\perp v | S_{uv}$. This procedure is very inefficient because this requires a number of independence tests that increases exponentially with the number of vertices. In other words, to determine whether there is a set separating $u$ and $v$, we might search all $2^{n-2}$ subsets of all

**Algorithm 1:** Edge-removal step of the PC algorithm for Bayesian Networks

```
for i ← 0 to |V_H| − 2 do
    while possible do
        Select any ordered pair of nodes u and v in H such
          that u ∈ ad_H(v), |ad_H(u) \ v| ≥ i
          (ad_H(x) := {y ∈ V | x → y, y → x, or x − y});
        if there exists S ⊆ (ad_H(u) \ v) s.t. |S| = i and
          u ⊥⊥_p v|S (i.e., u is independent of v given S in
          the probability distribution p) then
            Set S_uv = S_vu = S;
            Remove the edge u − v from H;
        end
    end
end
```

$n$ random variables excluding $u$ and $v$. So, the complexity for investigating each possible edge in the skeleton is $O(2^n)$ and hence the complexity for constructing the skeleton is $O(n^2 2^n)$, where $n$ is the number of vertices in the LWF CG. Since it is enough to find one $S$ making $u$ and $v$ independent to remove the undirected edge $u$ —— $v$, one obvious shortcut is to do the tests in some order, and skip unnecessary tests. In the PC algorithm for BNs the revised edge-removal step is done as shown in Algorithm 1.

Since the PC algorithm only looks at adjacencies of $u$ and $v$ in the current stage of the algorithm, rather than all possible subsets, the PC algorithm performs fewer independence tests compared to the IC algorithm. The complexity of the PC algorithm for DAGs is difficult to evaluate exactly, but with the *sparseness assumption* the worst case is bounded by $O(n^q)$ assuming an exact d-separation test (Spirtes, Glymour, and Scheines 2000) and bounded by $O(n^q)$ with high probability when a conditional independence test is used, where $n$ is the number of vertices and $q$ is the maximum number of the adjacent vertices of the true underlying DAG (Kalisch and Bühlmann 2007). Our main intuition is that replacing the skeleton recovery phase in the IC-like algorithm with a PC-like approach will speed up this phase and make it computationally scalable when the true underlying LWF CG is sparse, which is often a reasonable assumption (Kalisch and Bühlmann 2007; Colombo and Maathuis 2014) (see the skeleton recovery phase of Algorithm 2).

The looping procedure of the IC-like algorithm for complex recovery is computationally expensive. We use a polynomial time approach similar to the proposed algorithm by (Ma, Xie, and Geng 2008) to reduce the computational cost of the complex recovery (see the complex recovery phase of Algorithm 2). Finally, the IC-like algorithm uses three basic rules, namely the *transitivity rule*, the *necessity rule*, and the *double-cycle rule*, for changing the obtained pattern in the previous phase into the corresponding largest CG (see (Studený 1997) for details). When we have perfect conditional independence, both IC-like and LCD algorithms recover the structure of the model correctly if the probability distribution of the data is *faithful* to some LWF CGs i.e., all conditional independencies among variables can be represented by an LWF CG. The entire process is formally described in

**Algorithm 2: PC4LWF**: a PC-like algorithm for Learning LWF CGs

**Input:** a set $V$ of nodes and a probability distribution $p$ faithful to an unknown LWF CG $G$.
**Output:** The pattern of $G$.

1 Let $H$ denote the complete undirected graph over $V$;
/* Skeleton Recovery */
2 Run Algorithm 1;
/* Complex Recovery (Ma, Xie, and Geng 2008) */
3 Initialize $H^* = H$;
4 **for** *each vertex pair* $\{u, v\}$ *s.t.* $u$ *and* $v$ *are not adjacent in* $H$ **do**
5  **for** *each* $u - w$ *in* $H^*$ **do**
6  **if** $u \not\perp\!\!\!\perp_p v | (S_{uv} \cup \{w\})$ **then**
7  Orient $u - w$ as $u \to w$ in $H^*$;
8  **end**
9  **end**
10 **end**
11 Take the pattern of $H^*$;
/* To get the pattern of $H^*$ in line 11, at each step, we consider a pair of candidate complex arrows $u_1 \to w_1$ and $u_2 \to w_2$ with $u_1 \neq u_2$, then we check whether there is an undirected path from $w_1$ to $w_2$ such that none of its intermediate vertices is adjacent to either $u_1$ or $u_2$. If there exists such a path, then $u_1 \to w_1$ and $u_2 \to w_2$ are labeled (as complex arrows). We repeat this procedure until all possible candidate pairs are examined. The pattern is then obtained by removing directions of all unlabeled as complex arrows in $H^*$ (Ma, Xie, and Geng 2008). */

Algorithm 2. We prove the correctness of Algorithm 2 in (Javidian, Valtorta, and Jamshidi 2020b).

**Computational Complexity Analysis of Algorithm 2.** The complexity of the algorithm for a graph $G$ is bounded by the largest degree in $G$. Let $k$ be the maximal degree of any vertex and let $n$ be the number of vertices. Then in the (rare) worst case the number of conditional independence (CI) tests required by the algorithm is bounded by $2\binom{n}{2}\sum_{i=0}^{k}\binom{n-2}{i} \leq \frac{n^2(n-2)^k}{(k-1)!}$ (Javidian, Valtorta, and Jamshidi 2020b).

## The Stable PC-like Algorithms

In this section, we show that the **PC4LWF** algorithm proposed in the previous section is order dependent, in the sense that the output can depend on the order in which the variables are given. 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 $V = (X_1, \ldots, X_p)$. In **PC4LWF**, all conditional independence queries are estimated by statistical CI tests at some pre-specified significance level (p value) $\alpha$. For example, if the distribution of $V$ is multivariate Gaussian, one can test for zero partial correlation, see, e.g., (Kalisch and Bühlmann 2007). Hence, 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 2.

In the skeleton recovery phase of the **PC4LWF** algorithm (line 2), the order of variables affects the estimation of the skeleton and the separating sets. In particular, as noted for the special case of BNs in (Colombo and Maathuis 2014), 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. However, in the *sample version* (i.e., where CI tests have to be estimated from data), we typically make mistakes in keeping or removing edges. The main sources of erroneous CI tests are large condition sets given a limited sample size and the curse-of-dimensionality (Cheng, Bell, and Liu 1997). 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 $U$-structure decision in the complex recovery phase. In the sample version, however, different separating sets in the skeleton recovery phase may yield different decisions about $U$-structures in the complex recovery phase (line 3-11 of Algorithm 2). This is illustrated in Example 2.

**Example 1** ***(Order dependent skeleton of the PC4LWF algorithm)*** *Suppose that the distribution of $V = \{a, b, c, d, e\}$ is faithful to the DAG in Figure 1(a). This DAG encodes the following conditional independencies 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 $\alpha$: $a \perp\!\!\!\perp d | \{b, c\}$, $a \perp\!\!\!\perp e | \{b, c, d\}$, 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 **PC4LWF** algorithm with two different orderings: $order_1(V) = (d, e, a, c, b)$ and $order_2(V) = (d, c, e, a, b)$. The resulting skeletons are shown in Figures 1(b) and 1(c), respectively.*

Table 1: The trace table of Algorithm 2 for $i = 3$ and $\text{order}_1(V) = (d, e, a, c, b)$.

| 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 |

*We see that the skeletons are different, and that both are incorrect as the edge $c$ — $e$ is missing. The skeleton for $order_2(V)$ contains an additional error, as there is an additional edge $a$ — $e$. We now go through Algorithm 2 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 ver-*

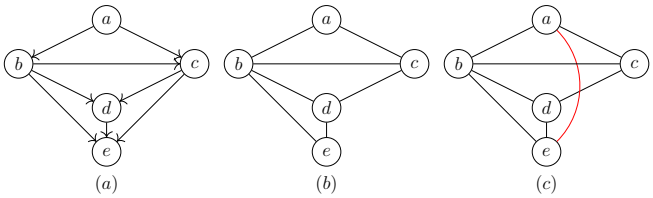


Figure 1: (a) The DAG $G$, (b) the skeleton returned by Algorithm 2 with $\text{order}_1(V)$, (c) the skeleton returned by Algorithm 2 with $\text{order}_2(V)$.

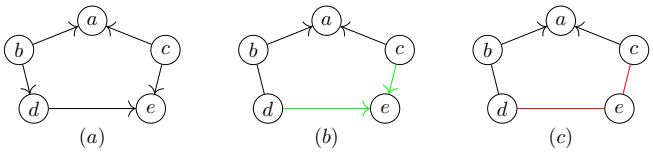


Figure 2: (a) The DAG $G$, (b) the CG returned after the complex recovery phase of Algorithm 2 with $\text{order}_1(V)$, (c) the CG returned after the complex recovery phase of Algorithm 2 with $\text{order}_3(V)$.

Table 2: The trace table of Algorithm 2 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 $u$ —— $v$ 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 |

*tices 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 2 for $i = 3$ and $\text{order}_1(V) = (d, e, a, c, b)$. Table 2 shows the trace table of Algorithm 2 for $i = 3$ and $\text{order}_2(V) = (d, c, e, a, b)$. This illustrates that order dependent separating sets in the skeleton recovery phase of the sample version of Algorithm 2 can lead to order dependent skeletons.*

**Example 2** ***(Order dependent separating sets and U-structures of the* `PC4LWF` *algorithm.)*** *Suppose that the distribution of $V = \{a, b, c, d, e\}$ is faithful to the DAG $G$ in Figure 2(a). DAG $G$ 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 complex recovery phase of Algorithm 2 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 2(b) and 2(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 $S_{cd} = S_{dc} = \{e\}$. This illustrates that order dependent separating sets in the skeleton recovery phase of the sample version of Algorithm 2 can lead to order dependent U-structures.*

**Algorithm 3:** `STABLE-PC4LWF`: an order independent (stable) algorithm for learning LWF CGs.

**Input:** A set $V$ of nodes and a probability distribution $p$ faithful to an unknown LWF CG $G$ and an ordering order($V$) on the variables.
**Output:** The pattern of G

```
1  Let H denote the complete undirected graph over
     V = {v_1, ..., v_n};
   /* Skeleton Recovery                                   */
2  for i ← 0 to |V_H| − 2 do
3      for j ← 1 to |V_H| do
4          Set a_H(v_j) = ad_H(v_j);
5      end
6      while possible do
7          Select any ordered pair of nodes u and v in H such
             that u ∈ a_H(v) and |a_H(u) \ v| ≥ i using
             order(V);
8          if there exists S ⊆ (a_H(u) \ v) s.t. |S| = i and
             u ⊥⊥_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 —— v from H;
11         end
12     end
13 end
   /* Complex Recovery                                    */
14 Follow the same procedures in Algorithm 2 (lines: 11-19).
```

We now propose several modifications of the `PC4LWF` algorithm for learning LWF CGs (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 (Colombo and Maathuis 2014) did for the PC algorithm in the case of DAGs.

## Order Independent Skeleton Recovery

We first consider estimation of the skeleton in the adjacency search of the `PC4LWF` algorithm. The pseudocode for our modification is given in Algorithm 3, where the order-dependence is removed by the addition of the highlighted lines 3-5. The resulting algorithm is called `STABLE-PC4LWF` *(stable PC-like for LWF CGs)*. 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 $i$ i.e., computations required for $i$-level can be performed in parallel. As a result, the runtime of the parallelized `STABLE-PC4LWF` algorithm is much shorter than the `PC4LWF` algorithm for learning LWF chain graphs. Furthermore, this approach enjoys the advantage of knowing the number of CI tests of each level in advance. This allows the CI tests to be *evenly distributed* over different cores, so that the parallelized algorithm can achieve maximum possible speedup. The `STABLE-PC4LWF` is correct, i.e. it returns an LWF CG to which the given probability distribution is faithful (Theorem 1), and it yields order independent skeletons in the sample version (Theorem 2). Proofs of these theorems are in (Javidian, Valtorta, and Jamshidi 2020b).

Table 3: Order dependence issues and corresponding modifications of the **PC4LWF** algorithm that remove the problem. "Yes" indicates that the corresponding aspect of the graph is estimated order independently in the sample version.

| Algorithm | Skeleton Recovery | Complex Recovery |
|---|---|---|
| **PC4LWF** | No | No |
| **STABLE-PC4LWF** | Yes | No |
| Stable (Conservative/Majority rule) **PC4LWF** | Yes | Yes |

**Theorem 1** *Let the distribution of $V$ be faithful to an LWF 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-PC4LWF** *algorithm is the pattern of $G$.*

**Theorem 2** *The skeleton resulting from the sample version of the* **STABLE-PC4LWF** *algorithm is order independent.*

**Example 3 (Order independent skeletons)** *We go back to Example 1, and consider the sample version of Algorithm 3. The algorithm now outputs the skeleton shown in Figure 1(b) for both orderings $order_1(V)$ and $order_2(V)$. 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 $b$ and $c$. 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 $a$ —— $e$ and $c$ —— $e$.*

## Order Independent Complex Recovery

We propose two methods i.e., **Conservative PC-like algorithm** and **Majority rule PC-like algorithm** to resolve the order dependence in the determination of the minimal complexes in LWF CGs, by extending the proposed approaches *Conservative PC algorithm (MPC)* (Ramsey, Spirtes, and Zhang 2006) and *Majority rule PC algorithm (MPC)* (Colombo and Maathuis 2014) for unshielded colliders recovery in DAGs, respectively. Due to space limitation, we refer readers to (Javidian, Valtorta, and Jamshidi 2020b) for details.

## Evaluation

To investigate the performance of the proposed algorithms, we use the same approach as in (Ma, Xie, and Geng 2008) for evaluating the performance of the LCD algorithm on Gaussian LWF CGs. We run our algorithms **PC4LWF** and **STABLE-PC4LWF** as well as the LCD algorithm on randomly generated LWF CGs and we compare the results and report summary error measures. We

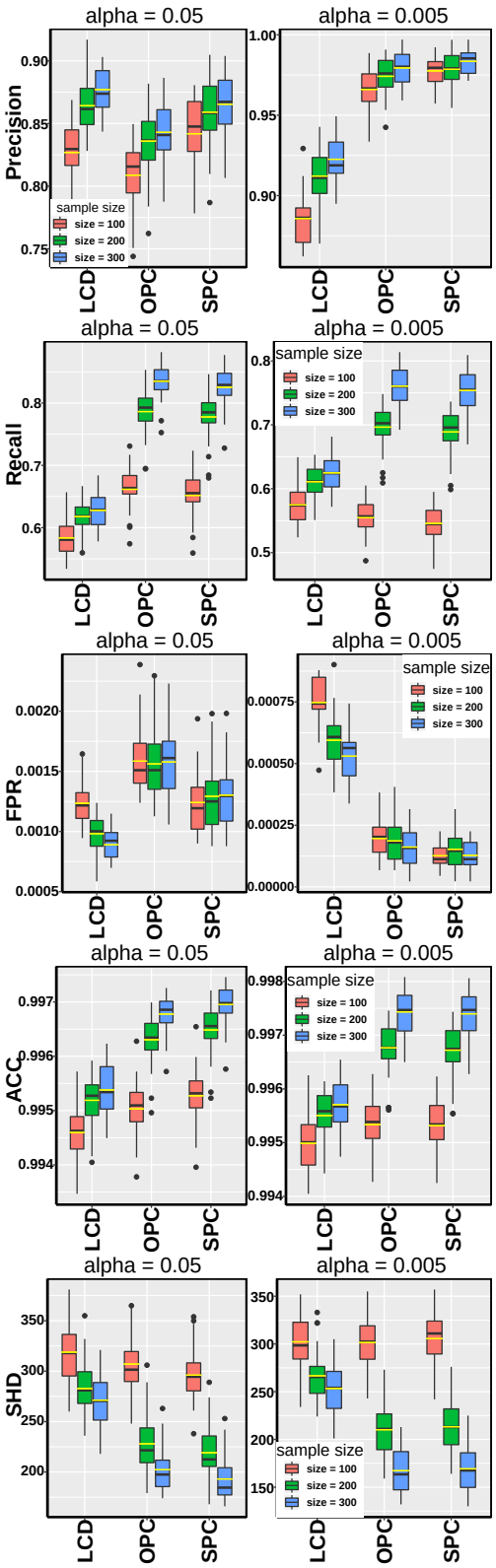


Figure 3: Performance of the LCD and PC-like algorithms (**PC4LWF** (OPC) and **STABLE-PC4LWF** (SPC)) for randomly generated Gaussian chain graph models: over 30 repetitions with 300 variables, expected degree N = 3, and significance levels $\alpha = 0.05, 0.005$.

evaluate the performance of algorithms in terms of the six measurements that are commonly used for constraint-based learning algorithms (Colombo and Maathuis 2014; Kalisch and Bühlmann 2007; Tsamardinos, Brown, and Aliferis 2006), and we report on the first five measurements (see (Javidian, Valtorta, and Jamshidi 2020b) for a more detailed report): (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 (Tsamardinos, Brown, and Aliferis 2006) to compare the structure of the learned and the original graphs), and (f) runtime for the pattern recovery algorithms. In principle, large values of TPR, TDR, and ACC, and small values of FPR and SHD indicate good performance.

Figure 3 illustrate the performance of the algorithms in a high-dimensional setting with 300 variables and samples of size 100, 200, and 300. Figure 3 shows that: **(a)** the performance of the **PC4LWF** algorithms (especially **STABLE-PC4LWF**) in the high-dimensional setting are better than the LCD algorithm (except for the FPR with the p-value $\alpha = 0.05$). This indicates that the **STABLE-PC4LWF** algorithm is computationally feasible and very likely statistically consistent in high-dimensional and sparse setting. **(b)** **STABLE-PC4LWF** shows improved performance in the high-dimensional setting against the original **PC4LWF** algorithm, in particular for error measures precision and FPR (and SHD with the p-value $\alpha = 0.05$). **(c)** In general, the p-value has a very large impact on the performance of the algorithms. Our empirical results suggests that in order to obtain a better precision, FPR, accuracy, and SHD, one can choose a small value (say $\alpha = 0.005$) for the significance level of CI tests. **(d)** While the four error measures TPR, TDR, ACC, and SHD show a clear tendency with increasing sample size, the behavior of FPR is not so clear. The latter seems surprising at first sight but notice that differences are very small with no meaningful indication about the behavior of FPR based on the sample size. Other results in sparse and moderately dense settings are in (Javidian, Valtorta, and Jamshidi 2020b).

In summary, empirical simulations show that our proposed algorithms achieve competitive results with the LCD learning algorithm; in particular, in the Gaussian case the **STABLE-PC4LWF** algorithm recovers a better-quality structure than the LCD and original **PC4LWF** algorithms, especially in high-dimensional sparse settings. Besides resolving the order dependence problem, the **STABLE-PC4LWF** algorithm has the advantage that it is easily *parallelizable* and very likely *consistent in high-dimensional settings* (conditions for consistency would need to be investigated as future work) under the same conditions as the original **PC4LWF** algorithm.

## Acknowledgments

This work is partially supported by NASA (RASPBERRY-SI Grant No. 80NSSC20K1720) and NSF (SmartSight Award 2007202).

## References

[Bhattacharya, Malinsky, and Shpitser 2019] Bhattacharya, R.; Malinsky, D.; and Shpitser, I. 2019. Causal inference under interference and network uncertainty. In *Proceedings of UAI 2019*.

[Cheng, Bell, and Liu 1997] Cheng, J.; Bell, D. A.; and Liu, W. 1997. Learning belief networks from data: An information theory based approach. In *Proceedings of the 6th CIKM*, 325–331.

[Colombo and Maathuis 2014] Colombo, D., and Maathuis, M. H. 2014. Order-independent constraint-based causal structure learning. *The Journal of Machine Learning Research* 15(1):3741–3782.

[Frydenberg 1990] Frydenberg, M. 1990. The chain graph Markov property. *Scandinavian Journal of Statistics* 17(4):333–353.

[Javidian, Valtorta, and Jamshidi 2020a] Javidian, M. A.; Valtorta, M.; and Jamshidi, P. 2020a. Learning LWF chain graphs: A Markov blanket discovery approach. volume 124 of *Proceedings of Machine Learning Research*, 1069–1078. Virtual: PMLR.

[Javidian, Valtorta, and Jamshidi 2020b] Javidian, M. A.; Valtorta, M.; and Jamshidi, P. 2020b. Learning LWF chain graphs: an order independent algorithm. https://arxiv.org/abs/2005.14037.

[Kalisch and Bühlmann 2007] Kalisch, M., and Bühlmann, P. 2007. Estimating high-dimensional directed acyclic graphs with the PC-algorithm. *J. Mach. Learn. Res.* 8:613–636.

[Lauritzen and Wermuth 1989] Lauritzen, S., and Wermuth, N. 1989. Graphical models for associations between variables, some of which are qualitative and some quantitative. *The Annals of Statistics* 17(1):31–57.

[Ma, Xie, and Geng 2008] Ma, Z.; Xie, X.; and Geng, Z. 2008. Structural learning of chain graphs via decomposition. *Journal of Machine Learning Research* 9:2847–2880.

[Peña, Sonntag, and Nielsen 2014] Peña, J. M.; Sonntag, D.; and Nielsen, J. 2014. An inclusion optimal algorithm for chain graph structure learning. *In Proceedings of the 17th International Conference on Artificial Intelligence and Statistics* 778–786.

[Ramsey, Spirtes, and Zhang 2006] Ramsey, J.; Spirtes, P.; and Zhang, J. 2006. Adjacency-faithfulness and conservative causal inference. In *Proceedings of UAI Conference*, 401–408.

[Sonntag et al. 2015] Sonntag, D.; Järvisalo, M.; Peña, J. M.; and Hyttinen, A. 2015. Learning optimal chain graphs with answer set programming. In *Proceedings of the 31st UAI Conference*, 822–831.

[Spirtes, Glymour, and Scheines 2000] Spirtes, P.; Glymour, C.; and Scheines, R. 2000. *Causation, Prediction and Search, second ed.* MIT Press, Cambridge, MA.

[Studený 1997] Studený, M. 1997. A recovery algorithm for chain graphs. *International Journal of Approximate Reasoning* 17:265–293.

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

[Verma and Pearl 1991] Verma, T., and Pearl, J. 1991. Equivalence and synthesis of causal models. In *Proceedings of the Sixth UAI Conference*, UAI '90, 255–270.

[Wang, Liu, and Zhu 2019] Wang, J.; Liu, S.; and Zhu, M. 2019. Local structure learning of chain graphs with the false discovery rate control. *Artif. Intell. Rev.* 52(1):293–321.