

# REVERSIBLE MCMC ON MARKOV EQUIVALENCE CLASSES OF SPARSE DIRECTED ACYCLIC GRAPHS<sup>1</sup>

BY YANGBO HE, JINZHU JIA AND BIN YU

*Peking University, Peking University and University of California, Berkeley*

Graphical models are popular statistical tools which are used to represent dependent or causal complex systems. Statistically equivalent causal or directed graphical models are said to belong to a Markov equivalent class. It is of great interest to describe and understand the space of such classes. However, with currently known algorithms, sampling over such classes is only feasible for graphs with fewer than approximately 20 vertices. In this paper, we design reversible irreducible Markov chains on the space of Markov equivalent classes by proposing a *perfect* set of operators that determine the transitions of the Markov chain. The stationary distribution of a proposed Markov chain has a closed form and can be computed easily. Specifically, we construct a concrete perfect set of operators on sparse Markov equivalence classes by introducing appropriate conditions on each possible operator. Algorithms and their accelerated versions are provided to efficiently generate Markov chains and to explore properties of Markov equivalence classes of sparse directed acyclic graphs (DAGs) with thousands of vertices. We find experimentally that in most Markov equivalence classes of sparse DAGs, (1) most edges are directed, (2) most undirected subgraphs are small and (3) the number of these undirected subgraphs grows approximately linearly with the number of vertices.

**1. Introduction.** Graphical models based on directed acyclic graphs (DAGs, denoted as  $\mathcal{D}$ ) are widely used to represent causal or dependent relationships in various scientific investigations, such as bioinformatics, epidemiology, sociology and business [12, 13, 19, 20, 24, 32, 35]. A DAG encodes the independence and conditional independence restrictions of variables. However, because different DAGs can encode the same set of independencies or conditional independencies, most of the time we cannot distinguish DAGs via observational data [31].

---

Received September 2012; revised April 2013.

<sup>1</sup>Supported in part by NSFC (11101008, 11101005, 71271211), 973 Program-2007CB814905, DPHEC-20110001120113, US NSF Grants DMS-11-07000, DMS-09-07632, DMS-06-05165, DMS-12-28246 3424, SES-0835531 (CDI), US ARO grant W911NF-11-1-0114 and the Center for Science of Information (CSoI), a US NSF Science and Technology Center, under Grant agreement CCF-0939370. This research was also supported by School of Mathematical Science, the Center of Statistical Sciences, the Key Lab of Mathematical Economics and Quantitative Finance (Ministry of Education), the Key lab of Mathematics and Applied Mathematics (Ministry of Education), and the Microsoft Joint Lab on Statistics and information technology at Peking University.

*MSC2010 subject classifications.* 62H05, 60J10, 05C81.

*Key words and phrases.* Sparse graphical model, reversible Markov chain, Markov equivalence class, Causal inference.

A Markov equivalence class is used to represent all DAGs that encode the same dependencies and independencies [2, 6, 33]. A Markov equivalence class can be visualized (or modeled) and uniquely represented by a completed partial directed acyclic graph (completed PDAG for short) [6] which possibly contains both directed edges and undirected edges [22]. There exists a one-to-one correspondence between completed PDAGs and Markov equivalence classes [2]. The completed PDAGs are also called essential graphs by Andersson et al. [2] and maximally oriented graphs by Meek [26].

A set of completed PDAGs can be used as a model space. The modeling task is to discover a proper Markov equivalence class in the model space [3, 4, 8, 9, 18, 25]. Understanding the set of Markov equivalence classes is important and useful for statistical causal modeling [14, 15, 21]. For example, if the number of DAGs is large for Markov equivalence classes in the model space, searching based on unique completed PDAGs could be substantially more efficient than searching based on DAGs [6, 25, 27]. Moreover, if most completed PDAGs in the model space have many undirected edges (with nonidentifiable directions), many interventions might be needed to identify the causal directions [11, 17].

Because the number of Markov equivalence classes increases superexponentially with the number of vertices (e.g., more than  $10^{18}$  classes with 10 vertices) [15], it is hard to study sets of Markov equivalence classes. To our knowledge, only completed PDAGs with a small given number of vertices ( $\leq 10$ ) have been studied thoroughly in the literature [14, 15, 29]. Moreover, these studies focus on the size of Markov equivalence classes, which is defined as the number of DAGs in a Markov equivalence class. Gillispie and Perlman [15] obtain the true size distribution of all Markov equivalence classes with a given number (10 or fewer) of vertices by listing all classes. Peña [29] designs a Markov chain to estimate the proportion of the equivalence classes containing only one DAG for graphs with 20 or fewer vertices.

In recent years, sparse graphical models have become popular tools for fitting high-dimensional multivariate data. The sparsity assumption introduces restrictions on the model space; a standard restriction is that the number of edges in the graph be less than a small multiple of the number of vertices. It is thus both interesting and important to be able to explore the properties of subsets of graphical models, especially with sparsity constraints on the edges.

In this paper, we propose a reversible irreducible Markov chain on Markov equivalence classes. We first introduce a perfect set of operators that determine the transitions of the chain. Then we obtain the stationary distribution of the chain by counting (or estimating) all possible transitions for each state of the chain. Finally, based on the stationary distribution of the chain (or estimated stationary distribution), we re-weight the samples from the chain. Hence these reweighted samples can be seen as uniformly (or approximately uniformly) generated from the Markov equivalence classes of interest. Our proposal allows the study of properties of the sets that contain sparse Markov equivalence classes in a computationally efficient manner for sparse graphs with thousands of vertices.

1.1. *A Markov equivalence class and its representation.* In this section, we give a short overview for the representations of a Markov equivalence class.

A graph  $\mathcal{G}$  is defined as a pair  $(V, E)$ , where  $V = \{x_1, \dots, x_p\}$  denotes the vertex set with  $p$  variables, and  $E$  denotes the edge set. Let  $n_{\mathcal{G}} = |E|$  be the number of edges in  $\mathcal{G}$ . A directed (undirected) edge is denoted as  $\rightarrow$  or  $\leftarrow$  ( $-$ ). A graph is directed (undirected) if all of its edges are directed (undirected). A sequence  $(x_1, x_2, \dots, x_k)$  of distinct vertices is called a *path* from  $x_1$  to  $x_k$  if either  $x_i \rightarrow x_{i+1}$  or  $x_i \leftarrow x_{i+1}$  is in  $E$  for all  $i = 1, \dots, k - 1$ . A path is partially directed if at least one edge in it is directed. A path is directed (undirected) if all edges are directed (undirected). A *cycle* is a path from a vertex to itself.

A *directed acyclic graph* (DAG), denoted by  $\mathcal{D}$ , is a directed graph which does not contain any directed cycle. Let  $\tau$  be a subset of  $V$ . The *subgraph*  $\mathcal{D}_{\tau} = (\tau, E_{\tau})$  induced by the subset  $\tau$  has vertex set  $\tau$  and edge set  $E_{\tau}$ , the subset of  $E$  which contains the edges with both vertices in  $\tau$ . A subgraph  $x \rightarrow z \leftarrow y$  is called a *v-structure* if there is no edge between  $x$  and  $y$ . A *partially directed acyclic graph* (PDAG), denoted by  $\mathcal{P}$ , is a graph with no directed cycle.

A graphical model consists of a DAG and a joint probability distribution. With the graphical model, in general, the conditional independencies implied by the joint probability distribution can be read from the DAG. A *Markov equivalence class* (MEC) is a set of DAGs that encode the same set of independencies or conditional independencies. Let the *skeleton* of an arbitrary graph  $\mathcal{G}$  be the undirected graph with the same vertices and edges as  $\mathcal{G}$ , regardless of their directions. Verma and Pearl [36] proved the following characterization of Markov equivalence classes:

LEMMA 1 (Verma and Pearl [36]). *Two DAGs are Markov equivalent if and only if they have the same skeleton and the same v-structures.*

This lemma implies that, among DAGs in an equivalence class, some edge orientations may vary, while others will be preserved (e.g., those involved in a v-structure). Consequently, a Markov equivalence class can be represented uniquely by a *completed PDAG*, defined as follows:

DEFINITION 1 (Completed PDAG [6]). *The completed PDAG of a DAG  $\mathcal{D}$ , denoted as  $\mathcal{C}$ , is a PDAG that has the same skeleton as  $\mathcal{D}$ , and an edge is directed in  $\mathcal{C}$  if and only if it has the same orientation in every equivalent DAG of  $\mathcal{D}$ .*

According to Definition 1 and Lemma 1, a completed PDAG of a DAG  $\mathcal{D}$  has the same skeleton as  $\mathcal{D}$ , and it keeps at least the directed edges that occur in the v-structures of  $\mathcal{D}$ . Another popular name of a completed PDAG is “essential graph” introduced by Andersson et al. [2], who introduce four necessary and sufficient conditions for a graph to be an essential graph; see them in Lemma 2, Appendix A.1. One of the conditions shows that all directed edges in a completed PDAG must be “strongly protected,” defined as follows:

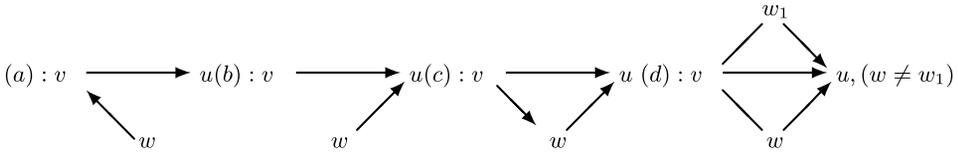


FIG. 1. Four configurations where  $v \rightarrow u$  is strongly protected in  $\mathcal{G}$ .

DEFINITION 2. Let  $\mathcal{G} = (V, E)$  be a graph. A directed edge  $v \rightarrow u \in E$  is strongly protected in  $\mathcal{G}$  if  $v \rightarrow u \in E$  occurs in at least one of the four induced subgraphs of  $\mathcal{G}$  in Figure 1.

If we delete all directed edges from a completed PDAG, we are left with several isolated undirected subgraphs. Each isolated undirected subgraph is a *chain component* of the completed PDAG. Observational data is not sufficient to learn the directions of undirected edges of a completed PDAG; one must perform additional intervention experiments. In general, the size of a chain component is a measure of “complexity” of causal learning; the larger the chain components are, the more interventions will be necessary to learn the underlying causal graph [17].

In learning graphical models [6] or studying Markov equivalence classes [29], Markov chains on completed PDAGs play an important role. We briefly introduce the existing methods to construct Markov chains on completed PDAGs in the next subsection.

1.2. *Markov chains on completed PDAGs.* To construct a Markov chain on completed PDAGs, we need to generate the transitions among them. In general, an *operator* that can modify the initial completed PDAG locally can be used to carry out a transition [6, 27, 29, 34]. Let  $\mathcal{C}$  be a completed PDAG. We consider six types of operators on  $\mathcal{C}$ : inserting an undirected edge (denoted by *InsertU*), deleting an undirected edge (*DeleteU*), inserting a directed edge (*InsertD*), deleting a directed edge (*DeleteD*), making a  $v$ -structure (*MakeV*) and removing a  $v$ -structure (*RemoveV*). We call *InsertU*, *DeleteU*, *InsertD*, *DeleteD*, *MakeV* and *RemoveV* the *types* of operators. An operator on a given completed PDAG is determined by two parts: its type and the modified edges. For example, the operator “*InsertU*  $x - y$ ” on  $\mathcal{C}$  represents inserting an undirected edge  $x - y$  to  $\mathcal{C}$ , and  $x - y$  is the modified edge of the operator. A *modified graph* of an operator is the same as the initial completed PDAG, except for the modified edges of the operator. A modified graph might (not) be a completed PDAG; see Example 1 in Section 2.1, of the Supplementary Material [16].

Madigan et al. [25], Perlman [34] and Peña [29] introduce several Markov chains based on the modified graphs of operators. At each state of these Markov chains, say  $\mathcal{C}$ , they move to the modified graph of an operator on  $\mathcal{C}$  only when the modified graph happens to be a completed PDAG, otherwise, stay at  $\mathcal{C}$ . In order to move to new completed PDAGs, Madigan et al. [25] search the operators

whose modified graphs are completed PDAG by checking Andersson's conditions [2] one by one. Perlman [34] introduces an alternative search approach that is more efficient by "exploiting further" Andersson's conditions.

When the modified graph of an operator on  $\mathcal{C}$  is not a completed PDAG, the operator might result in a transition from one completed PDAG  $\mathcal{C}$  to another. This operator also results in a "valid" transition. To obtain valid transitions, Chickering [6, 7] introduces the concept of *validity* for an operator on  $\mathcal{C}$ . Before defining "valid operator," we need a concept *consistent extension*. A *consistent extension* of a PDAG  $\mathcal{P}$  is a directed acyclic graph (DAG) on the same underlying set of edges, with the same orientations on the directed edges of  $\mathcal{P}$  and the same set of  $v$ -structures [10, 37]. According to Lemma 1, all consistent extensions of a PDAG  $\mathcal{P}$ , if they exist, belong to a unique Markov equivalence class. Hence if the modified graph of an operator is a PDAG and has a consistent extension, it can result in a completed PDAG that corresponds to a unique Markov equivalence class. We call it the *resulting* completed PDAG of the operator. Now a valid operator is defined as below.

**DEFINITION 3 (Valid operator).** An operator on  $\mathcal{C}$  is *valid* if (1) the modified graph of the operator is a PDAG and has a consistent extension, and (2) all modified edges in the modified graph occur in the resulting completed PDAG of the operator.

The first condition in Definition 3 guarantees that a valid operator results in a completed PDAG. The second condition guarantees that the valid operator is "effective;" that is, the change brought about by the operator occurs in the resulting completed PDAG. Here we notice that the second condition is implied by the context in Chickering [6]. Below we briefly introduce how to obtain the resulting completed PDAG of a valid operator from the modified graph.

Verma and Pearl [37] and Meek [26] introduce an algorithm for finding the completed PDAG from a "pattern" (given skeleton and  $v$ -structures). This method can be used to create the completed PDAG from a DAG or a PDAG. They first undirect every edge, except for those edges that participate in a  $v$ -structure. Then they choose one of the undirected edges and direct it if the corresponding directed edge is strongly protected, as shown in Figure 1(a), (c) or (d). The algorithm terminates when there is no undirected edge that can be directed.

Chickering [6] proposes an alternative approach to obtain the completed PDAG of a valid operator from its modified graph; see Example 2, Section 2.1 of the Supplementary Material [16]. The method includes two steps. The first step generates a consistent extension (a DAG) of the modified graph (a PDAG) using the algorithm described in Dor and Tarsi [10]. The second step creates a completed PDAG corresponding to the consistent extension [5, 6]. We describe Dor and Tarsi's algorithm and Chickering's algorithms in Section 1 of the Supplementary Material [16].

The approach proposed by Chickering [5, 6] is "more complicated but more efficient" [26] than Meek's method described above. Hence when constructing a

Markov chain, we use Chickering’s approach to obtain the resulting completed PDAG of a given valid operator from its modified graph.

With a set of valid operators, a Markov chain on completed PDAGs can be constructed. Let  $\mathcal{S}_p$  be the set of all completed PDAGs with  $p$  vertices,  $\mathcal{S}$  be a given subset of  $\mathcal{S}_p$ . For any completed PDAG  $\mathcal{C} \in \mathcal{S}$ , let  $\mathcal{O}_{\mathcal{C}}$  be a set of valid operators of interest to be defined later on  $\mathcal{C}$  in equation (3.2). A set of valid operators on  $\mathcal{S}$  is defined as

$$(1.1) \quad \mathcal{O} = \bigcup_{\mathcal{C} \in \mathcal{S}} \mathcal{O}_{\mathcal{C}}.$$

Here we notice that each operator in  $\mathcal{O}$  is specific to the completed PDAG that the operator applies to. A Markov chain  $\{e_t\}$  on  $\mathcal{S}$  based on the set  $\mathcal{O}$  can be defined as follows.

DEFINITION 4 (A Markov chain  $\{e_t\}$  on  $\mathcal{S}$ ). The Markov chain  $\{e_t\}$  determined by a set of valid operators  $\mathcal{O}$  is generated as follows: start at an arbitrary completed PDAG, denoted as  $e_0 = \mathcal{C}_0 \in \mathcal{S}$ , and repeat the following steps for  $t = 0, 1, \dots$ :

- (1) At the  $t$ th step we are at a completed PDAG  $e_t$ .
- (2) We choose an operator  $o_{e_t}$  uniformly from  $\mathcal{O}_{e_t}$ ; if the resulting completed PDAG  $\mathcal{C}_{t+1}$  of  $o_{e_t}$  is in  $\mathcal{S}$ , move to  $\mathcal{C}_{t+1}$  and set  $e_{t+1} = \mathcal{C}_{t+1}$ ; otherwise we stay at  $e_t$  and set  $e_{t+1} = e_t$ .

Given the same operator set, the Markov chain in Definition 4 has more new transition states for any completed PDAG than those based on the modified graphs of operators [25, 29, 34]. This is because some valid operators will result in new completed PDAGs even if their modified graphs are not completed PDAGs. Consequently, the transitions, which are generated by these operators, are not contained in Markov chains based on the modified graphs.

The set  $\mathcal{S}$  is the finite state space of chain  $\{e_t\}$ . Clearly, the sequence of completed PDAGs  $\{e_t : t = 0, 1, \dots\}$  in Definition 4 is a discrete-time Markov chain [23, 28]. Let  $p_{\mathcal{C}\mathcal{C}'}$  be the one-step transition probability of  $\{e_t\}$  from  $\mathcal{C}$  to  $\mathcal{C}'$  for any two completed PDAGs  $\mathcal{C}$  and  $\mathcal{C}'$  in  $\mathcal{S}$ . A Markov chain  $\{e_t\}$  is *irreducible* if it can reach any completed PDAG starting at any state in  $\mathcal{S}$ . If  $\{e_t\}$  is irreducible, there exists a unique distribution  $\pi = (\pi_{\mathcal{C}}, \mathcal{C} \in \mathcal{S})$  satisfying balance equations (see Theorems 1.7.7 and 1.5.6 in [28])

$$(1.2) \quad \pi_{\mathcal{C}} = \sum_{\mathcal{C}' \in \mathcal{S}} \pi_{\mathcal{C}'} p_{\mathcal{C}'\mathcal{C}} \quad \text{for all } \mathcal{C} \in \mathcal{S}.$$

An irreducible chain  $e_t$  is *reversible* if there exists a probability distribution  $\pi$  such that

$$(1.3) \quad \pi_{\mathcal{C}} p_{\mathcal{C}\mathcal{C}'} = \pi_{\mathcal{C}'} p_{\mathcal{C}'\mathcal{C}} \quad \text{for all } \mathcal{C}, \mathcal{C}' \in \mathcal{S}.$$

It is well known that  $\pi$  is the unique stationary distribution of the discrete-time Markov chain  $\{e_t\}$  if it is finite, reversible, and irreducible; see Lemma 1.9.2 in [28]. Moreover, the stationary probabilities  $\pi_{\mathcal{C}}$  can be calculated efficiently if the Markov chain satisfies equation (1.3).

The properties of the Markov chain  $\{e_t\}$  given in Definition 4 depend on the operator set  $\mathcal{O}$ . To implement score-based searching in the whole set of Markov equivalence classes, Chickering [6] introduces a set of operators with types of InsertU, DeleteU, InsertD, DeleteD, MakeV or ReverseD (reversing the direction of a directed edge), subject to some validity conditions. Unfortunately, the Markov chain in Definition 4 is not reversible if the set of Chickering's operators is used. Our goal is to design a reversible Markov chain, as it makes it easier to compute the stationary distribution, and thereby to study the properties of a subset of Markov equivalence classes.

In Section 2, we first discuss the properties of an operator set  $\mathcal{O}$  needed to guarantee that the Markov chain is reversible. Section 2 also explains how to use the samples from the Markov chain to study properties of any given subset of Markov equivalence classes. In Section 3 we focus on studying sets of sparse Markov equivalence classes. Finally, in Section 4, we report the properties of directed edges and chain components in sparse Markov equivalence classes with up to one thousand of vertices.

**2. Reversible Markov chains on Markov equivalence classes.** Let  $\mathcal{S}$  be any subset of the set  $\mathcal{S}_p$  that contains all completed PDAGs with  $p$  vertices, and  $\mathcal{O}$  be a set of operators on  $\mathcal{S}$  defined in equation (1.1). As in Definition 4, we can obtain a Markov chain denoted by  $\{e_t\}$ . We first discuss four properties of  $\mathcal{O}$  that guarantee that  $\{e_t\}$  is reversible and irreducible. They are *validity*, *distinguishability*, *irreducibility* and *reversibility*. We call a set of operators *perfect* if it satisfies these four properties. Then we give the stationary distribution of  $\{e_t\}$  when  $\mathcal{O}$  is perfect and show how to use  $\{e_t\}$  to study properties of  $\mathcal{S}$ .

2.1. *A reversible Markov chain based on a perfect set of operators.* Let  $p_{\mathcal{C}\mathcal{C}'}$  be a one-step transition probability of  $\{e_t\}$  from  $\mathcal{C}$  to  $\mathcal{C}'$  for any two completed PDAGs  $\mathcal{C}$  and  $\mathcal{C}'$  in  $\mathcal{S}$ . In order to formulate  $p_{\mathcal{C}\mathcal{C}'}$  clearly, we introduce two properties of  $\mathcal{O}$ : Validity and Distinguishability.

**DEFINITION 5 (Validity).** Given  $\mathcal{S}$  and any completed PDAG  $\mathcal{C}$  in  $\mathcal{S}$ , a set of operators  $\mathcal{O}$  on  $\mathcal{S}$  is valid if for any operator  $o_{\mathcal{C}}$  ( $o$  without confusion below) in  $\mathcal{O}_{\mathcal{C}}$ ,  $o$  is valid according to Definition 3 and the resulting completed PDAG obtained by applying  $o$  to  $\mathcal{C}$ , which is different from  $\mathcal{C}$ , is also in  $\mathcal{S}$ .

According to Definition 5, if a set of operators  $\mathcal{O}$  on  $\mathcal{S}$  is valid, we can move to a new completed PDAG in each step of  $\{e_t\}$  and the one-step transition probability of any completed PDAG to itself is zero:

$$(2.1) \quad p_{\mathcal{C}\mathcal{C}} = 0 \quad \text{for any completed PDAG } \mathcal{C} \in \mathcal{S}.$$

For a set of valid operators  $\mathcal{O}$  and any completed PDAG  $\mathcal{C}$  in  $\mathcal{S}$ , we define the resulting completed PDAGs of the operators in  $\mathcal{O}_{\mathcal{C}}$  as the *direct successors* of  $\mathcal{C}$ . For any direct successor of  $\mathcal{C}$ , denoted by  $\mathcal{C}'$ , we obtain  $p_{\mathcal{C}\mathcal{C}'}$  clearly as in equation (2.2) if  $\mathcal{O}$  has the following property.

**DEFINITION 6 (Distinguishability).** A set of valid operators  $\mathcal{O}$  on  $\mathcal{S}$  is distinguishable if for any completed PDAG  $\mathcal{C}$  in  $\mathcal{S}$ , different operators in  $\mathcal{O}_{\mathcal{C}}$  will result in different completed PDAGs.

If  $\mathcal{O}$  is distinguishable, for any direct successor of  $\mathcal{C}$ , denoted by  $\mathcal{C}'$ , there is a unique operator in  $\mathcal{O}_{\mathcal{C}}$  that can transform  $\mathcal{C}$  to  $\mathcal{C}'$ . Thus, the number of operators in  $\mathcal{O}_{\mathcal{C}}$  is the same as the number of direct successors of  $\mathcal{C}$ . Sampling operators from  $\mathcal{O}_{\mathcal{C}}$  uniformly generates a uniformly random transition from  $\mathcal{C}$  to its direct successors. By denoting  $M(\mathcal{O}_{\mathcal{C}})$  as the number of operators in  $\mathcal{O}_{\mathcal{C}}$ , we have

$$(2.2) \quad p_{\mathcal{C}\mathcal{C}'} = \begin{cases} 1/M(\mathcal{O}_{\mathcal{C}}), & \mathcal{C}' \text{ is a direct successor of } \mathcal{C} \in \mathcal{S}; \\ 0, & \text{otherwise.} \end{cases}$$

We introduce this property because it makes computation of  $p_{\mathcal{C}\mathcal{C}'}$  efficient: if  $\mathcal{O}$  is distinguishable, we know  $p_{\mathcal{C}\mathcal{C}'}$  right away from  $M(\mathcal{O}_{\mathcal{C}})$ .

In order to make sure the Markov chain  $\{e_t\}$  is irreducible and reversible, we introduce two more properties of  $\mathcal{O}$ : irreducibility and reversibility.

**DEFINITION 7 (Irreducibility).** A set of operators  $\mathcal{O}$  on  $\mathcal{S}$  is irreducible if for any two completed PDAGs  $\mathcal{C}, \mathcal{C}' \in \mathcal{S}$ , there exists a sequence of operators in  $\mathcal{O}$  such that we can obtain  $\mathcal{C}'$  from  $\mathcal{C}$  by applying these operators sequentially.

If  $\mathcal{O}$  is irreducible, starting at any completed PDAG in  $\mathcal{S}$ , we have positive probability to reach any other completed PDAG via a sequence of operators in  $\mathcal{O}$ . Thus, the Markov chain  $\{e_t\}$  is irreducible.

**DEFINITION 8 (Reversibility).** A set of operators  $\mathcal{O}$  on  $\mathcal{S}$  is reversible if for any completed PDAG  $\mathcal{C} \in \mathcal{S}$  and any operator  $o \in \mathcal{O}_{\mathcal{C}}$  with  $\mathcal{C}'$  being the resulting completed PDAG of  $o$ , there is an operator  $o' \in \mathcal{O}_{\mathcal{C}'}$  such that  $\mathcal{C}$  is the resulting completed PDAG of  $o'$ .

If the set of operators  $\mathcal{O}$  on  $\mathcal{S}$  is valid, distinguishable and reversible, for any pair of completed PDAGs  $\mathcal{C}, \mathcal{C}' \in \mathcal{S}$ ,  $\mathcal{C}$  is also a direct successor of  $\mathcal{C}'$  if  $\mathcal{C}'$  is a direct successor of  $\mathcal{C}$ . For any  $\mathcal{C} \in \mathcal{S}$  and any of its direct successors  $\mathcal{C}'$ , we have

$$(2.3) \quad p_{\mathcal{C}\mathcal{C}'} = 1/M(\mathcal{O}_{\mathcal{C}}) \quad \text{and} \quad p_{\mathcal{C}'\mathcal{C}} = 1/M(\mathcal{O}_{\mathcal{C}'}).$$

Let  $\mathcal{T} = \sum_{\mathcal{C} \in \mathcal{S}} M(\mathcal{O}_{\mathcal{C}})$ , and define a probability distribution as

$$(2.4) \quad \pi_{\mathcal{C}} = M(\mathcal{O}_{\mathcal{C}})/\mathcal{T}.$$

Clearly, equation (1.3) holds for  $\pi_{\mathcal{C}}$  in equation (2.4) if  $\mathcal{O}$  is valid, distinguishable and reversible.  $\pi_{\mathcal{C}}$  is the unique stationary distribution of  $\{e_t\}$  if it is also irreducible [1, 23, 28].

In the following proposition, we summarize our results about the Markov chain  $\{e_t\}$  on  $\mathcal{S}$ , and give its stationary distribution.

**PROPOSITION 1** (Stationary distribution of  $\{e_t\}$ ). *Let  $\mathcal{S}$  be any given set of completed PDAGs. The set of operators is defined as  $\mathcal{O} = \bigcup_{\mathcal{C} \in \mathcal{S}} \mathcal{O}_{\mathcal{C}}$  where  $\mathcal{O}_{\mathcal{C}}$  is a set of operators on  $\mathcal{C}$  for any  $\mathcal{C}$  in  $\mathcal{S}$ . Let  $M(\mathcal{O}_{\mathcal{C}})$  be the number of operators in  $\mathcal{O}_{\mathcal{C}}$ . For the Markov chain  $\{e_t\}$  on  $\mathcal{S}$  generated according to Definition 4, if  $\mathcal{O}$  is perfect, that is, the properties—validity, distinguishability, reversibility and irreducibility—hold for  $\mathcal{O}$ , then:*

- (1) *the Markov chain  $\{e_t\}$  is irreducible and reversible;*
- (2) *the distribution  $\pi_{\mathcal{C}}$  in equation (2.4) is the unique stationary distribution of  $\{e_t\}$  and  $\pi_{\mathcal{C}} \propto M(\mathcal{O}_{\mathcal{C}})$ .*

The challenge is to construct a concrete perfect set of operators. In Section 3, we carry out such a construction for a set of Markov equivalence classes with sparsity constraints and provide algorithms to obtain a reversible Markov chain. We now show that a reversible Markov chain can be used to compute interesting properties of a completed PDAG set  $\mathcal{S}$ .

**2.2. Estimating the properties of  $\mathcal{S}$  by a perfect Markov chain.** For any  $\mathcal{C} \in \mathcal{S}$ , let  $f(\mathcal{C})$  be a real function describing any property of interest of  $\mathcal{C}$ , and the random variable  $u$  be uniformly distributed on  $\mathcal{S}$ . In order to understand the property of interest, we compute the distribution of  $f(u)$ .

Let's consider one example in the literature. The proportion of Markov equivalence classes of size one (equivalently, completed PDAGs that are directed) in  $\mathcal{S}_p$  is studied in the literature [14, 15, 29]. For this purpose, we can define  $f(u)$  as the size of Markov equivalence classes represented by  $u$  and obtain the proportion by computing the probability of  $\{f(u) = 1\}$ .

Let  $A$  be any subset of  $\mathbb{R}$ , the probability of  $\{f(u) \in A\}$  is

$$(2.5) \quad \mathbb{P}(f(u) \in A) = \frac{|\{\mathcal{C} : f(\mathcal{C}) \in A, \mathcal{C} \in \mathcal{S}\}|}{|\mathcal{S}|} = \frac{\sum_{\mathcal{C} \in \mathcal{S}} I_{\{f(\mathcal{C}) \in A\}}}{|\mathcal{S}|},$$

where  $|\mathcal{S}|$  is the number of elements in the set  $\mathcal{S}$  and  $I$  is an indicator function.

Let  $\{e_t\}_{t=1, \dots, N}$  be a realization of Markov chain  $\{e_t\}$  on  $\mathcal{S}$  based on a perfect operator set  $\mathcal{O}$  according to Definition 4 and  $M_t = M(\mathcal{O}_{e_t})$ . Let  $\pi(e_t)$  be the stationary probability of Markov chain  $\{e_t\}$ . From Proposition 1, we have  $\pi(e_t) \propto M_t$  for  $t = 1, \dots, N$ . We can use  $\{e_t, M_t\}_{t=1, \dots, N}$  to estimate the probability of  $\{f(u) \in A\}$  by

$$(2.6) \quad \hat{\mathbb{P}}_N(f(u) \in A) = \frac{\sum_{t=1}^N I_{\{f(e_t) \in A\}} M_t^{-1}}{\sum_{t=1}^N M_t^{-1}}.$$

From the ergodic theory of Markov chains (see Theorem 1.10.2 in [28]), we can get Proposition 2 directly.

PROPOSITION 2. *Let  $\mathcal{S}$  be a given set of completed PDAGs, and assume the set of operators  $\mathcal{O}$  on  $\mathcal{S}$  is perfect. The Markov chain  $\{e_t\}_{t=1,\dots,N}$  is obtained according to Definition 4. Then the estimator  $\hat{\mathbb{P}}_N(\{f(u) \in A\})$  in equation (2.6) converges to  $\mathbb{P}(\{f(u) \in A\})$  in equation (2.5) with probability one, that is,*

$$(2.7) \quad \mathbb{P}(\hat{\mathbb{P}}_N(f(u) \in A) \rightarrow \mathbb{P}(f(u) \in A) \text{ as } N \rightarrow \infty) = 1.$$

Proposition 2 shows that the estimator defined in equation (2.6) is a consistent estimator of  $\mathbb{P}(f(u) \in A)$ . We can study any given subset of Markov equivalence classes via equation (2.6) if we can obtain  $\{e_t\}_{t=1,\dots,N}$  and  $\{M_t\}_{t=1,\dots,N}$ . We now turn to construct a concrete perfect set of operators for a set of completed PDAGs with sparsity constraints and then introduce algorithms to run a reversible Markov chain.

**3. A Reversible Markov chain on completed PDAGs with sparsity constraints.** We define a set of Markov equivalence classes  $\mathcal{S}_p^n$  with  $p$  vertices and at most  $n$  edges as follows:

$$(3.1) \quad \mathcal{S}_p^n = \{\mathcal{C} : \mathcal{C} \text{ is a completed PDAG with } p \text{ vertices and } n_{\mathcal{C}} \leq n\},$$

where  $n_{\mathcal{C}}$  is the number of edges in  $\mathcal{C}$ . Recall that  $\mathcal{S}_p$  denotes the set of all completed PDAGs with  $p$  vertices. Clearly,  $\mathcal{S}_p^n = \mathcal{S}_p$  when  $n \geq p(p-1)/2$ .

We now construct a perfect set of operators on  $\mathcal{S}_p^n$ . Notice that our constructions can be extended to adapt to some other sets of completed PDAGs, say, a set of completed PDAGS with a given maximum degree. In Section 3.1, we construct the perfect set of operators for any completed PDAG in  $\mathcal{S}_p^n$ . In Section 3.2, we propose algorithms and their accelerated version for efficiently obtaining a Markov chain based on the perfect set of operators.

3.1. *Construction of a perfect set of operators on  $\mathcal{S}_p^n$ .* In order to construct a perfect set of operators, we need to define the set of operators on each completed PDAG in  $\mathcal{S}_p^n$ . Let  $\mathcal{C}$  be a completed PDAG in  $\mathcal{S}_p^n$ . We consider six types of operators on  $\mathcal{C}$  that were introduced in Section 1.2: InsertU, DeleteU, InsertD, DeleteD, MakeV and RemoveV. The operators on  $\mathcal{C}$  with the same type but different modified edges constitute a set of operators. We introduce six sets of operators on  $\mathcal{C}$  denoted by  $InsertU_{\mathcal{C}}$ ,  $DeleteU_{\mathcal{C}}$ ,  $InsertD_{\mathcal{C}}$ ,  $DeleteD_{\mathcal{C}}$ ,  $MakeV_{\mathcal{C}}$  and  $RemoveV_{\mathcal{C}}$  in Definition 9. In addition to the conditions that guarantee validity, for each type of operators, we also introduce other constraints to make sure that all operators are reversible.

First we explain some notation used in Definition 9. Let  $x$  and  $y$  be any two distinct vertices in  $\mathcal{C}$ . The neighbor set of  $x$  denoted by  $N_x$  consists of every vertex

$y$  with  $x - y$  in  $\mathcal{C}$ . The common neighbor set of  $x$  and  $y$  is defined as  $N_{xy} = N_x \cap N_y$ .  $x$  is a *parent* of  $y$  and  $y$  is a *child* of  $x$  if  $x \rightarrow y$  occurs in  $\mathcal{C}$ . A vertex  $u$  is a common child of  $x$  and  $y$  if  $u$  is a child of both  $x$  and  $y$ .  $\Pi_x$  represents the set of all parents of  $x$ .

**DEFINITION 9** (Six sets of operators on  $\mathcal{C}$ ). Let  $\mathcal{C}$  be a completed PDAG in  $\mathcal{S}_p^n$  and  $n_{\mathcal{C}}$  be the number of edges in  $\mathcal{C}$ . We introduce six sets of operators on  $\mathcal{C}$ :  $InsertU_{\mathcal{C}}$ ,  $DeleteU_{\mathcal{C}}$ ,  $InsertD_{\mathcal{C}}$ ,  $DeleteD_{\mathcal{C}}$ ,  $MakeV_{\mathcal{C}}$  and  $RemoveV_{\mathcal{C}}$  as follows.

(a) For any two vertices  $x, y$  that are not adjacent in  $\mathcal{C}$ , the operator “InsertU  $x - y$ ” on  $\mathcal{C}$  is in  $InsertU_{\mathcal{C}}$  if and only if **(iu<sub>1</sub>)**  $n_{\mathcal{C}} < n$ ; **(iu<sub>2</sub>)** “InsertU  $x - y$ ” is valid; **(iu<sub>3</sub>)** for any  $u$  that is a common child of  $x, y$  in  $\mathcal{C}$ , both  $x \rightarrow u$  and  $y \rightarrow u$  occur in the resulting completed PDAG of “InsertU  $x - y$ .”

(b) For any undirected edge  $x - y$  in  $\mathcal{C}$ , the operator “DeleteU  $x - y$ ” on  $\mathcal{C}$  is in  $DeleteU_{\mathcal{C}}$  if and only if **(du<sub>1</sub>)** “DeleteU  $x - y$ ” is valid.

(c) For any two vertices  $x, y$  that are not adjacent in  $\mathcal{C}$ , the operator “InsertD  $x \rightarrow y$ ” on  $\mathcal{C}$  is in  $InsertD_{\mathcal{C}}$  if and only if **(id<sub>1</sub>)**  $n_{\mathcal{C}} < n$ ; **(id<sub>2</sub>)** “InsertD  $x \rightarrow y$ ” is valid; **(id<sub>3</sub>)** for any  $u$  that is a common child of  $x, y$  in  $\mathcal{C}$ ,  $y \rightarrow u$  occurs in the resulting completed PDAG of “InsertD  $x \rightarrow y$ .”

(d) For any directed edge  $x \rightarrow y$  in  $\mathcal{C}$ , operator “DeleteD  $x \rightarrow y$ ” on  $\mathcal{C}$  is in  $DeleteD_{\mathcal{C}}$  if and only if **(dd<sub>1</sub>)** “DeleteD  $x \rightarrow y$ ” is valid; **(dd<sub>2</sub>)** for any  $v$  that is a parent of  $y$  but not a parent of  $x$ , directed edge  $v \rightarrow y$  in  $\mathcal{C}$  occurs in the resulting completed PDAG of “DeleteD  $x \rightarrow y$ .”

(e) For any subgraph  $x - z - y$  in  $\mathcal{C}$ , the operator “MakeV  $x \rightarrow z \leftarrow y$ ” on  $\mathcal{C}$  is in  $MakeV_{\mathcal{C}}$  if and only if **(mv<sub>1</sub>)** “MakeV  $x \rightarrow z \leftarrow y$ ” is valid.

(f) For any  $v$ -structure  $x \rightarrow z \leftarrow y$  of  $\mathcal{C}$ , the operator “RemoveV  $x \rightarrow z \leftarrow y$ ” on  $\mathcal{C}$  is in  $RemoveV_{\mathcal{C}}$  if and only if **(rv<sub>1</sub>)**  $\Pi_x = \Pi_y$ ; **(rv<sub>2</sub>)**  $\Pi_x \cup N_{xy} = \Pi_z \setminus \{x, y\}$ ; **(rv<sub>3</sub>)** every undirected path between  $x$  and  $y$  contains a vertex in  $N_{xy}$ .

Munteanu and Bendou [27] discuss the constraints for the first five types of operators such that each one can transform one completed PDAG to another. Chickering [6] introduces the necessary and sufficient conditions such that these five types of operators are valid. We list the conditions introduced by Chickering [6] in Lemma 3, Appendix A.1, and employ them to guarantee that the conditions **iu<sub>2</sub>**, **du<sub>1</sub>**, **id<sub>2</sub>**, **dd<sub>1</sub>** and **mv<sub>1</sub>** in Definition 9 hold.

The set of operators on  $\mathcal{C}$  denoted by  $\mathcal{O}_{\mathcal{C}}$  is defined as follows:

$$(3.2) \quad \mathcal{O}_{\mathcal{C}} = InsertU_{\mathcal{C}} \cup DeleteU_{\mathcal{C}} \cup InsertD_{\mathcal{C}} \cup DeleteD_{\mathcal{C}} \cup MakeV_{\mathcal{C}} \cup RemoveV_{\mathcal{C}}.$$

Taking the union over all completed PDAGs in  $\mathcal{S}_p^n$ , we define the set of operators on  $\mathcal{S}_p^n$  as

$$(3.3) \quad \mathcal{O} = \bigcup_{\mathcal{C} \in \mathcal{S}_p^n} \mathcal{O}_{\mathcal{C}},$$

where  $\mathcal{O}_C$  is the set of operators in equation (3.2). In the main result of this paper, we show that  $\mathcal{O}$  in equation (3.3) is a perfect set of operators on  $\mathcal{S}_p^n$ .

**THEOREM 1** (A perfect set of operators on  $\mathcal{S}_p^n$ ).  *$\mathcal{O}$  defined in equation (3.3) is a perfect set of operators on  $\mathcal{S}_p^n$ .*

Here we notice that **iu**<sub>3</sub>, **id**<sub>3</sub> and **dd**<sub>2</sub> are key conditions in Definition 9 to guarantee that  $\mathcal{O}$  is reversible. Without these three conditions, there are operators that are not reversible; see Example 3, Section 2.1 in the Supplementary Material [16]. We provide a proof of Theorem 1 in Appendix A.2.

The preceding section showed how to construct a perfect set of operators. A toy example is provided as Example 4 in Section 2.1 of the Supplementary Material [16]. Based on the perfect set of operators we can obtain a finite irreducible reversible discrete-time chain. In the next subsection, we provide detailed algorithms for obtaining a Markov chain on  $\mathcal{S}_p^n$  and their accelerated version.

**3.2. Algorithms.** In this subsection, we provide the algorithms in detail to generate a Markov chain on  $\mathcal{S}_p^n$ , defined in Definition 4 based on the perfect set of operators defined in (3.3). A sketch of Algorithm 1 is shown below; some steps of this algorithm are further explained in the subsequent algorithms.

Step A of Algorithm 1 constructs the sets of operators on completed PDAGs in the chain  $\{e_t\}$ . It is the most difficult step and dominates the time complexity of Algorithm 1. Step B and Step C can be implemented easily after  $\mathcal{O}_{e_t}$  is obtained. Step D can be implemented via Chickering’s method [6] that was mentioned in Section 1.2. We will show that the time complexity of obtaining a Markov chain

---

**Algorithm 1:** Road map to construct a Markov chain on  $\mathcal{S}_p^n$

---

**Input:**

$p$ , the number of vertices;  $n$ , the maximum number of edges;  $N$ , the length of Markov chain.

**Output:**

$\{e_t, M_t\}_{t=1, \dots, N}$ , where  $\{e_t\}$  is Markov chain and  $M_t$  is the number of operators in  $\mathcal{O}_{e_t}$ .

- 1 Initialize  $e_0$  as any completed PDAG in  $\mathcal{S}_p^n$
  - 2 **for**  $t \leftarrow 0$  **to**  $N$  **do**
    - Step A** Construct the set of operators  $\mathcal{O}_{e_t}$  in equation (3.2) via Algorithm 1.1;
    - Step B** Let  $M_t$  be the number of operators in  $\mathcal{O}_{e_t}$ ;
    - Step C** Randomly choose an operator  $o$  uniformly from  $\mathcal{O}_{e_t}$ ;
    - Step D** Apply operator  $o$  to  $e_t$ . Set  $e_{t+1}$  as the resulting completed PDAG of  $o$ .
  - 3 **return**  $\{e_t, M_t\}_{t=1, \dots, N}$ .
-

on  $S_p^n$  with length  $N$  ( $\{e_t\}_{t=1,\dots,N}$ ) is approximate  $O(Np^3)$  if  $n$  is the same order of  $p$ . For large  $p$ , we also provide an accelerated version that, in some cases, can run hundreds of times faster.

The rest of this section is arranged as follows. In Section 3.2.1, we first introduce the algorithms to implement Step A. In Section 3.2.2 we discuss the time complexity of our algorithm, and provide an acceleration method to speed up Algorithm 1.

3.2.1. *Implementation of Step A in Algorithm 1.* A detailed implementation of Step A (to construct  $\mathcal{O}_{e_t}$ ) is described in Algorithm 1.1. To construct  $\mathcal{O}_{e_t}$  in Algorithm 1.1, we go through all possible operators on  $e_t$  and choose those satisfying the corresponding conditions in Definition 9.

The conditions in Algorithm 1.1 include:  $\mathbf{iu}_1, \mathbf{iu}_2, \mathbf{iu}_3, \mathbf{du}_1, \mathbf{id}_1, \mathbf{id}_2, \mathbf{id}_3, \mathbf{dd}_1, \mathbf{dd}_2, \mathbf{rm}_1, \mathbf{rv}_1, \mathbf{rv}_2$  and  $\mathbf{mv}_1$ . For each possible operator, we check the corresponding conditions shown in Algorithm 1.1 one-by-one until one of them fails. Below, we introduce how to check these conditions.

---

**Algorithm 1.1:** Construct  $\mathcal{O}_{e_t}$  for a completed PDAG  $e_t$ .

---

**Input:** A completed PDAG  $e_t$  with  $p$  vertices.

**Output:** Operator set  $\mathcal{O}_{e_t}$ .

// All sets of possible modified edges of  $e_t$  used below,  
for example,  $\text{Undirected-edges}_{e_t}$ , are generated according  
to Definition 9.

- 1 Set  $\mathcal{O}_{e_t}$  as empty set
  - 2 **for** each undirected edge  $x - y$  in  $\text{Undirected-edges}_{e_t}$  **do**
  - 3      $\lfloor$  consider operator DeleteU  $x - x$ , add it to  $\mathcal{O}_{e_t}$  if  $\mathbf{du}_1$  holds,
  - 4 **for** each directed edge  $x \rightarrow y$  in  $\text{Directed-edges}_{e_t}$  **do**
  - 5      $\lfloor$  consider DeleteD  $x \rightarrow x$ , add it to  $\mathcal{O}_{e_t}$  if both  $\mathbf{dd}_1$  and  $\mathbf{dd}_2$  hold;
  - 6 **for** each  $v$ -structure  $x \rightarrow z \leftarrow y$  in  $\text{V-structures}_{e_t}$  **do**
  - 7      $\lfloor$  consider RemoveV  $x_k \rightarrow x_i \leftarrow x_l$ , add it to  $\mathcal{O}_{e_t}$  if  $\mathbf{rv}_1, \mathbf{rv}_2$  and  $\mathbf{rv}_3$  hold,
  - 8 **for** each undirected  $v$ -structure  $x - z - y$  in  $\text{Undirected-v-structures}_{e_t}$  **do**
  - 9      $\lfloor$  consider MakeV  $x_k \rightarrow x_i \leftarrow x_l$ , add it to  $\mathcal{O}_{e_t}$  if  $\mathbf{mv}_1$  holds,
  - 10 **if**  $n_{e_t} < n$  (i.e.,  $\mathbf{iu}_1$  or  $\mathbf{id}_1$  holds) **then**
  - 11     **for** each pair  $(x, y)$  in  $\text{Pairs-nonadj}_{e_t}$  **do**
  - 12          $\lfloor$  consider InsertU  $x - y$ , add it to  $\mathcal{O}_{e_t}$  if  $\mathbf{iu}_1, \mathbf{iu}_2$ , and  $\mathbf{iu}_3$  hold;
  - 13          $\lfloor$  consider InsertD  $x \rightarrow y$ , add it to  $\mathcal{O}_{e_t}$ , if  $\mathbf{id}_1, \mathbf{id}_2$  and  $\mathbf{id}_3$  hold;
  - 14          $\lfloor$  consider InsertD  $x \leftarrow y$ , add it to  $\mathcal{O}_{e_t}$  if  $\mathbf{id}_1, \mathbf{id}_2$  and  $\mathbf{id}_3$  hold.
  - 15 **return**  $\mathcal{O}_{e_t}$
-

The conditions  $\mathbf{id}_3$ ,  $\mathbf{id}_3$  and  $\mathbf{dd}_2$  in Algorithm 1.1 depend on both  $e_t$  and the resulting completed PDAGs of the operators. Intuitively, checking  $\mathbf{id}_3$ ,  $\mathbf{id}_3$  or  $\mathbf{dd}_2$  requires that we obtain the corresponding resulting completed PDAGs. We know that the time complexity of getting a resulting completed PDAG of  $e_t$  is  $O(pn_{e_t})$  [6, 10], where  $n_{e_t}$  is the number of edges in  $e_t$ . To avoid generating resulting completed PDAG, in the Supplementary Material [16], we provide three algorithms to check  $\mathbf{id}_3$ ,  $\mathbf{id}_3$  and  $\mathbf{dd}_2$  only based on  $e_t$  and in an efficient manner.

The other conditions can be tested via classical graph algorithms. These tests include: (1) whether two vertex sets are equal or not, (2) whether a subgraph is a clique or not and (3) whether all partially directed paths or all undirected paths between two vertices contain at least one vertex in a given set. Checking the first two types of conditions is trivial and very efficient because the sets involved are small for most completed PDAGs in  $S_p^n$  when  $n$  is of the same order of  $p$ . To check the conditions with the third type, we just need to check whether there is a partially directed path or undirected path between two given vertices not through any vertices in the given set. We check this using a depth-first search from the source vertex. When looking for an undirected path, we can search within the corresponding chain component that includes both the source and the destination vertices.

3.2.2. *Time complexity of Algorithm 1 and an accelerated version.* We now discuss the time complexity of Algorithm 1. For  $e_t \in S_p^n$ , let  $p$  and  $n_t$  be the number of vertices and edges in  $e_t$ , respectively,  $k_t$  be the number of  $v$ -structures in  $e_t$ , and  $k'_t$  be the number of undirected  $v$ -structures (subgraphs  $x - y - z$  with  $x$  and  $z$  nonadjacent) in  $e_t$ . To construct  $\mathcal{O}_{e_t}$ , in Step A of Algorithm 1 (equivalently, Algorithm 1.1), all possible operators we need to go through:  $n_t$  deleting operators (DeleteU and DeleteD),  $3(p(p - 1)/2 - n_t)$  inserting operators (InsertU and InsertD) when the number of edges in  $e_t$  is less than  $n$ ,  $k_t$  RemoveV operators and  $k'_t$  MakeV operators. There are at most  $Q_t = 1.5p(p - 1) - 2n_t + k_t + k'_t$  possible operators for  $e_t$ . Among all conditions in Algorithm 1.1, the most time-consuming one, which takes time  $O(p + n_t)$  [6], is to look for a path via the depth-first search for an operator with type of InsertD. We have that the time complexity of constructing  $\mathcal{O}_{e_t}$  in Algorithm 1.1 is  $O(Q_t(p + n_t))$  in the worst case and the time complexity of Algorithm 1 is  $O(\sum_{t=1}^N Q_t(p + n_t))$  in the worst case, where  $N$  is the length of Markov chain in Algorithm 1. We know that  $k_t$  and  $k'_t$  reach the maxima  $(p - 2)/2 * \text{floor}(p/2) * \text{ceil}(p/2)$  when  $e_t$  is a evenly divided complete bipartite graphs [15]. Consequently, the time complexity of Algorithm 1 are  $O(Np^4)$  in the worst case. Fortunately, when  $n$  is a few times of  $p$ , say  $n = 2p$ , all completed PDAGs in  $S_p^n$  are sparse and our experiments show  $k_t$  and  $k'_t$  are much less than  $O(p^2)$  for most completed PDAGs in Markov chain  $\{e_t\}_{t=1, \dots, N}$ . Hence the time complexity of Algorithm 1 is approximate  $O(Np^3)$  on average when  $n$  is a few times of  $p$ .

We can implement Algorithm 1 efficiently when  $p$  is not large (less or around 100 in our experiments). However, when  $p$  is larger, we need large  $N$  to guarantee the estimates reach convergence. Experiments in Section 4 show  $N = 10^6$  is suitable. In this case, cubic complexity ( $O(Np^3)$ ) of Algorithm 1 is unacceptable. We need to speed up the algorithms for a very large  $p$ .

Notice that in Algorithm 1, we obtain an irreducible and reversible Markov chain  $\{e_t\}$  and a sequence of numbers  $\{M_t\}$  by checking all possible operators on each  $e_t$ . The sequence  $\{M_t\}$  are used to compute the stationary probabilities of  $\{e_t\}$  according to Proposition 1. We now introduce an accelerated version of Algorithm 1 to generate irreducible and reversible Markov chains on  $S_p^n$ . The basic idea is that we do not check all possible operators but check some random samples. These random samples are then used to estimate  $\{M_t\}$ .

We first explain some notation used in the accelerated version. For each completed PDAG  $e_t$ , if  $n_{e_t} < n$ ,  $\mathcal{O}_{e_t}^{(all)}$  is the set of all possible operators on  $e_t$  with types of InsertU, DeleteU, InsertD, DeleteD, MakeV and RemoveV. If  $n_{e_t} = n$ , the number of edges in  $e_t$  reaches the upper bound  $n$ , no more edges can be inserted into  $e_t$ . Let  $\mathcal{O}_{e_t}^{(-insert)}$  be the set of operators obtained by removing operators with types of InsertU and InsertD from  $\mathcal{O}_{e_t}^{(all)}$ .  $\mathcal{O}_{e_t}^{(-insert)}$  is the set of all possible operators on  $e_t$  when  $n_{e_t} = n$ . We can obtain  $\mathcal{O}_{e_t}^{(all)}$  and  $\mathcal{O}_{e_t}^{(-insert)}$  easily via all possible modified edges introduced in Algorithm 1.1. The accelerated version of Algorithm 1 is shown in Algorithm 2.

In Algorithm 2,  $\mathcal{O}'_{e_t}$  (either  $\mathcal{O}_{e_t}^{(all)}$  or  $\mathcal{O}_{e_t}^{(-insert)}$ ) is the set of all possible operators on  $e_t$ ,  $\alpha \in (0, 1]$  is an acceleration parameter that determines how many operators in  $\mathcal{O}'_{e_t}$  are checked,  $\mathcal{O}_{e_t}^{(check)}$  is a set of checked operators that are randomly sampled without replacement from  $\mathcal{O}'_{e_t}$  and  $\tilde{\mathcal{O}}_{e_t}$  is the set of all perfect operators in  $\mathcal{O}_{e_t}^{(check)}$ . When  $\alpha = 1$ ,  $\tilde{\mathcal{O}}_{e_t} = \mathcal{O}_{e_t}$  and Algorithm 2 becomes back to Algorithm 1.

In Algorithm 2, because the operators in  $\tilde{\mathcal{O}}_{e_t}$  are i.i.d. sampled from  $\mathcal{O}_{e_t}$  in Step A' and operator  $o$  is chosen uniformly from  $\tilde{\mathcal{O}}_{e_t}$  in Step C', clearly,  $o$  is also chosen uniformly from  $\mathcal{O}_{e_t}$ . We have that the following Corollary 1 holds according to Proposition 1.

**COROLLARY 1** (Stationary distribution of  $\{e_t\}$  on  $S_p^n$ ). *Let  $S_p^n$ , defined in equation (3.1), be the set of completed PDAGs with  $p$  vertices and maximum  $n$  of edges,  $\mathcal{O}_{e_t}$ , defined in equation (3.2), be the set of operators on  $e_t$ , and  $M_t$  be the number of operators in  $\mathcal{O}_{e_t}$ . For the Markov chain  $\{e_t\}$  on  $S_p^n$  obtained via Algorithms 1 or 2, then:*

- (1) *the Markov chain  $\{e_t\}$  is irreducible and reversible;*
- (2) *the Markov chain  $\{e_t\}$  has a unique stationary distribution  $\pi$  and  $\pi(e_t) \propto M_t$ .*

---

**Algorithm 2:** An accelerated version of Algorithm 1.

---

**Input:**

$\alpha \in (0, 1]$ : an acceleration parameter;  $p, n$  and  $N$ , the same as input in Algorithm 1

**Output:**

$\{e_t, \hat{M}_t\}_{t=1, \dots, N}$ , where  $\hat{M}_t$  is an estimation of  $M_t = |\mathcal{O}_{e_t}|$

```

1 Initialize  $e_0$  as any completed PDAG in  $\mathcal{S}_p^n$ 
2 for  $t \leftarrow 0$  to  $N$  do
3   Step A':
4   | if  $n_{e_t} < n$  then
5   | | Set  $\mathcal{O}'_{e_t} = \mathcal{O}_{e_t}^{(\text{all})}$ 
6   | else
7   | | Set  $\mathcal{O}'_{e_t} = \mathcal{O}_{e_t}^{(-\text{insert})}$ 
8   | | Set  $m_t = |\mathcal{O}'_{e_t}|$ 
9   | | Randomly sample  $[\alpha m_t]$  operators without replacement from  $\mathcal{O}'_{e_t}$  to
   | | generate a set  $\mathcal{O}_{e_t}^{(\text{check})}$ , where  $[\alpha m_t]$  is the integer closest to  $\alpha m_t$ .
10  | | Check all operators in  $\mathcal{O}_{e_t}^{(\text{check})}$ , and choose perfect operators from it to
   | | construct a set of operators  $\tilde{\mathcal{O}}_{e_t}$ .
11  | | Set  $m_t^{(\tilde{\mathcal{O}})} = |\tilde{\mathcal{O}}_{e_t}|$ . If  $m_t^{(\tilde{\mathcal{O}})} = 0$ , go to line 9.
12  | end
13  Step B':
14  | | Let  $\hat{M}_t = m_t \frac{m_t^{(\tilde{\mathcal{O}})}}{[\alpha m_t]}$ ,
15  | end
16  Step C':
17  | | Randomly choose an operator  $o$  uniformly from  $\tilde{\mathcal{O}}_{e_t}$ .
18  | end
19  Step D:
20  | | Apply operator  $o$  to  $e_t$ . Set  $e_{t+1}$  as the resulting completed PDAG of  $o$ .
21  | end
22 return  $\{e_t, \hat{M}_t\}_{t=1, \dots, N}$ .
```

---

In Algorithm 2, we provide an estimate of  $M_t$  instead of calculating it exactly in Algorithm 1. Let  $|\mathcal{O}'_{e_t}| = m_t$ ,  $|\mathcal{O}_{e_t}^{(\text{check})}| = [\alpha m_t]$  and  $|\tilde{\mathcal{O}}_{e_t}| = m_t^{(\tilde{\mathcal{O}})}$ . Clearly, the ratio  $m_t^{(\tilde{\mathcal{O}})} / [\alpha m_t]$  is an unbiased estimator of the population proportion  $M_t / m_t$  via sampling without replacement. We can estimate  $M_t = |\mathcal{O}_{e_t}|$  in Step B' as

$$(3.4) \quad \hat{M}_t = m_t \frac{m_t^{(\tilde{\mathcal{O}})}}{[\alpha m_t]}.$$

We have that when  $[\alpha m_t]$  is large, the estimator  $\hat{M}_t$  has an approximate normal distribution with mean equal to  $M_t = |\mathcal{O}_{e_t}|$ .

Let the random variable  $u$  be uniformly distributed on  $\mathcal{S}_p^n$ ,  $f(u)$  be a real function describing a property of interest of  $u$  and  $A$  be a subset of  $\mathbb{R}$ . By replacing  $M_t$  with  $\hat{M}_t$  in equation (2.6), we estimate  $\mathbb{P}_N(\{f(u) \in A\})$  via  $\{e_t, \hat{M}_t\}_{t=1, \dots, N}$  as follows:

$$(3.5) \quad \hat{\mathbb{P}}'_N(f(u) \in A) = \frac{\sum_{t=1}^N I_{\{f(e_t) \in A\}} \hat{M}_t^{-1}}{\sum_{t=1}^N \hat{M}_t^{-1}},$$

where  $\mathbb{P}_N(f(u) \in A)$  is defined in equation (2.5).

In the accelerated version, only  $100\alpha\%$  of all possible operators on  $e_t$  are checked. In Section 4, our experiments on  $\mathcal{S}_{100}^{150}$  show that the accelerated version can speed up the approach nearly  $\frac{1}{\alpha}$  times, and that equation (3.5) provides almost the same results as equation (2.6) in which  $\{e_t, M_t\}_{t=1, \dots, N}$  from Algorithm 1 are used. Roughly speaking, if we set  $\alpha = 1/p$ , the time complexity of our accelerated version can reduce to  $O(Np^2)$ .

**4. Experiments.** In this section, we conduct experiments to illustrate the reversible Markov chains proposed in this paper and their applications for studying Markov equivalence classes. The main points obtained from these experiments are as follows:

(1) For  $\mathcal{S}_p$  with small  $p$ , the estimations of our proposed are very close to true values. For  $\mathcal{S}_p^n$  with large  $p$  (up to 1000), the accelerated version of our proposed approach is also very efficient, and the estimations in equations (2.6) and (3.5) converge quickly as the length of Markov chain increases.

(2) For completed PDAGs in  $\mathcal{S}_p^n$  with sparsity constraints ( $n$  is a small multiple of  $p$ ), we see that (i) most edges are directed, (ii) the sizes of maximum chain components (measured by the number of vertices) are very small (around ten) even for large  $p$  (around 1000) and (iii) the number of chain components grows approximately linearly with  $p$ .

As we know, under the assumption that there are no latent or selection variables present, causal inference based on observational data will give a completed PDAG. Interventions are needed to infer the directions of the undirected edges in the completed PDAG. Our results show that if the underlying completed PDAG is sparse, in the model space of Markov equivalence classes, most graphs have few undirected edges and small chain components. They give hope for learning causal relationships via observational data and for inferring the directions of the undirected edges via interventions.

In Section 4.1, we evaluate our methods by comparing the size distributions of Markov equivalence classes in  $\mathcal{S}_p$  with small  $p$  to true distributions ( $p = 3, 4$ ) or Gillispie’s results ( $p = 6$ ) [15]. In Section 4.2, we report the proportion of directed

edges and the properties of chain components of Markov equivalence classes under sparsity constraints. In Section 4.3, we show experimentally that Algorithm 2 is much faster than Algorithm 1, and that the difference in the estimates obtained is small. Finally, we study the asymptotic properties of our proposed estimators in Section 4.4.

4.1. *Size distributions of Markov equivalence classes in  $\mathcal{S}_p$  for small  $p$ .* We consider size distributions of completed PDAGs in  $\mathcal{S}_p$  for  $p = 3, 4$  and 6, respectively. There are 11 Markov equivalence classes in  $\mathcal{S}_3$ , and 185 Markov equivalence classes in  $\mathcal{S}_4$ . Here we can get the true size distributions for  $\mathcal{S}_3$  and  $\mathcal{S}_4$  by listing all the Markov equivalence classes and calculating the size of each explicitly. Gillespie and Perlman calculate the true size probabilities for  $\mathcal{S}_6$  by listing all classes; these are denoted as GP-values. We estimate the size probabilities via equation (2.6) with the Markov chains from Algorithm 1. We ran ten independent Markov chains using Algorithm 1 to calculate the mean and standard deviation of each estimate. The results are shown in Table 1, where  $N$  is the sample size (length of Markov chain). We can see that the means are very close to true values or GP-values, and the standard deviations are also very small.

We implemented our proposed method (Algorithm 1, the version without acceleration) in Python, and ran it on a computer with a 2.6 GHZ processor. In Table 1,  $T$  is the time used to estimate the size distribution for  $\mathcal{S}_3$ ,  $\mathcal{S}_4$  or  $\mathcal{S}_6$ . These results were obtained within at most tens of seconds. In comparison, a MCMC method in [30] took more than one hour (in C++ on a 2.6 GHZ computer) in order to get similar estimates of the proportions of Markov equivalence classes of size one. It is worth noting that our estimates are based on a single Markov chain, while the results in [30] are based on  $10^4$  independent Markov chains with  $10^6$  steps.

4.2. *Markov equivalence classes with sparsity constraints.* We now study the sets  $\mathcal{S}_p^n$  of Markov equivalence classes defined in equation (3.1). The number of vertices  $p$  is set to 100, 200, 500 or 1000, and the maximum edge constraint  $n$  is set to  $rp$  where  $r$  is the ratio of  $n$  to  $p$ . For each  $p$ , we consider three ratios: 1.2, 1.5 and 3. The completed PDAGs in  $\mathcal{S}_p^{rp}$  are sparse since  $r \leq 3$ . Define the size of a chain component as the number of vertices it contains. In this section, we report four distributions for completed PDAGs in  $\mathcal{S}_p^{rp}$ : the distribution of proportions of directed edges, the distribution of the numbers of chain components and the distribution of the maximum size of chain components. The results about the distribution of the numbers of  $v$ -structures are reported in the Supplementary Material [16]. In each simulation, given  $p$  and  $r$ , a Markov chain with length of  $10^6$  on  $\mathcal{S}_p^{rp}$  is generated via Algorithm 2 to estimate the distributions via equation (3.5). The acceleration parameter  $\alpha$  is set to 0.1, 0.05, 0.01 and 0.001 for  $p = 100, 200, 500$  and 1000, respectively.

In Figure 2, twelve distributions of proportions of directed edges are reported for  $\mathcal{S}_p^{rp}$  with different  $p$  and ratio  $r$ . We mark the minimums, 5% quartiles (solid

TABLE 1  
*Size distributions for  $S_p$  with  $p = 3, 4$  and  $6$ , respectively.  $N$  is the sample size,  $T$  is the time (seconds) used to estimate the size distributions with a Markov chain, GP-values are obtained by Gillispie and Perlman [15]*

$p = 3, N = 10^4, T = 2 \text{ sec}$			$p = 4, N = 10^4, T = 3 \text{ sec}$		
Size	True value	Mean (Std)	Size	True value	Mean (Std)
1	0.36363*	0.36422 (0.00540)	1	0.31892*	0.31859 (0.00946)
2	0.27273	0.27160 (0.00412)	2	0.25946	0.25929 (0.00590)
3	0.27273	0.27274 (0.00217)	3	0.19460	0.19572 (0.00635)
6	0.0909	0.09144 (0.00262)	4	0.10270	0.10229 (0.00395)
			6	0.02162	0.02162 (0.00145)
			8	0.06486	0.06464 (0.00291)
			10	0.03243	0.03249 (0.00202)
			24	0.00540	0.00536 (0.00078)

$p = 6, N = 10^5, T = 60 \text{ sec}$					
Size	GP-value	Mean (Std)	Size	GP-value	Mean (Std)
1	0.28667*	0.28588 (0.00393)	48	0.00013	0.00013 (0.00004)
2	0.25858	0.25897 (0.00299)	50	0.00034	0.00034 (0.00007)
3	0.17064	0.17078 (0.00248)	52	0.00017	0.00018 (0.00003)
		⋮	54	0.00017	0.00018 (0.00004)
28	0.00017	0.00017 (0.00004)	60	0.00019	0.00020 (0.00004)
30	0.00169	0.00170 (0.00017)	72	0.00006	0.00006 (0.00002)
32	0.00236	0.00238 (0.00017)	88	0.00004	0.00004 (0.00001)
36	0.00052	0.00053 (0.00008)	144	0.00009	0.00009 (0.00003)
38	0.00034	0.00035 (0.00004)	156	0.00006	0.00006 (0.00003)
40	0.00118	0.00120 (0.00010)	216	0.00001	0.00001 (0.00002)
42	0.00051	0.00052 (0.00009)			

circles below boxes), 1st quartiles, medians, 3rd quartiles and maximums of these distributions. We can see that for a fixed  $p$ , the proportion of directed edges increases with the number of edges in the completed PDAG. For example, when the ratio  $r = 1.2$ , the medians (red lines in boxes) of proportions are near 92%; when the ratio  $r = 1.5$ , the medians are near 95%; when ratio  $r = 3$ , the medians are near 98%.

The distributions of the numbers of chain components of completed PDAGs in  $S_p^{r,p}$  are shown in Figure 3. We plot the distributions for  $S_p^{1.5,p}$  in the main window and the distributions for  $r = 1.2$  and  $r = 3$  in two sub-windows. We can see that the medians of the numbers of chain components are close to 5, 10, 20, and 40 for completed PDAGs in  $S_p^{1.5,p}$  with  $p = 100, 200, 500$  and  $1000$ , respectively. It seems that there is a linear relationship between the number of chain compo-

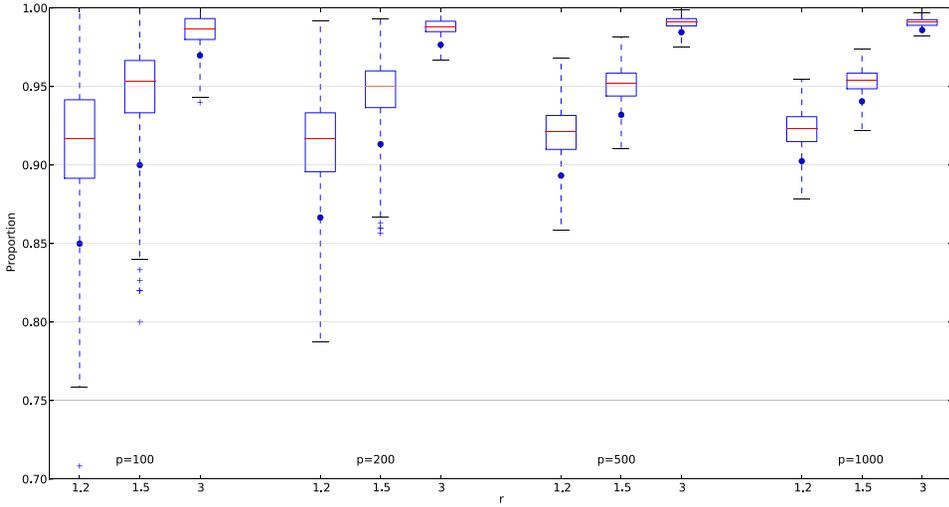


FIG. 2. Distribution of proportion of directed edges in completed PDAGs in  $\mathcal{S}_p^{r,p}$ . The lines in the boxes and the solid circles under the boxes indicate the medians and the 5% quartiles, respectively.

nents and the number of vertices  $p$ . In the insets, similar results are shown in the distributions for  $r = 1.2$  and  $r = 3$ .

The distributions of the maximum sizes of chain components of completed PDAGs in  $\mathcal{S}_p^{r,p}$  are shown in Figure 4. For  $\mathcal{S}_p^{1.5,p}$  in the main window, the medians of the four distributions are approximately 4, 5, 6 and 7 for  $p = 100, 200, 500$

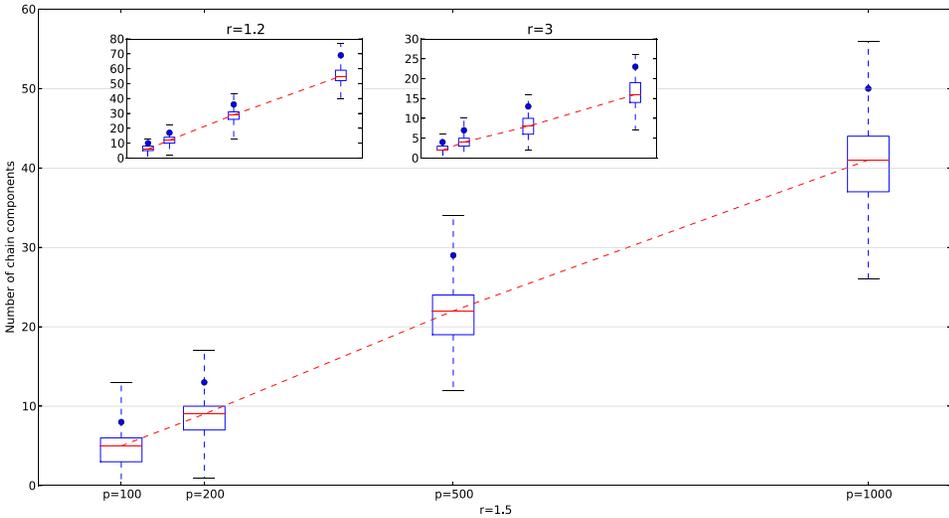


FIG. 3. Distributions of numbers of chain components of completed PDAGs in  $\mathcal{S}_p^{r,p}$ . The lines in the boxes and the solid circles above the boxes indicate the medians and the 95% quartiles, respectively.

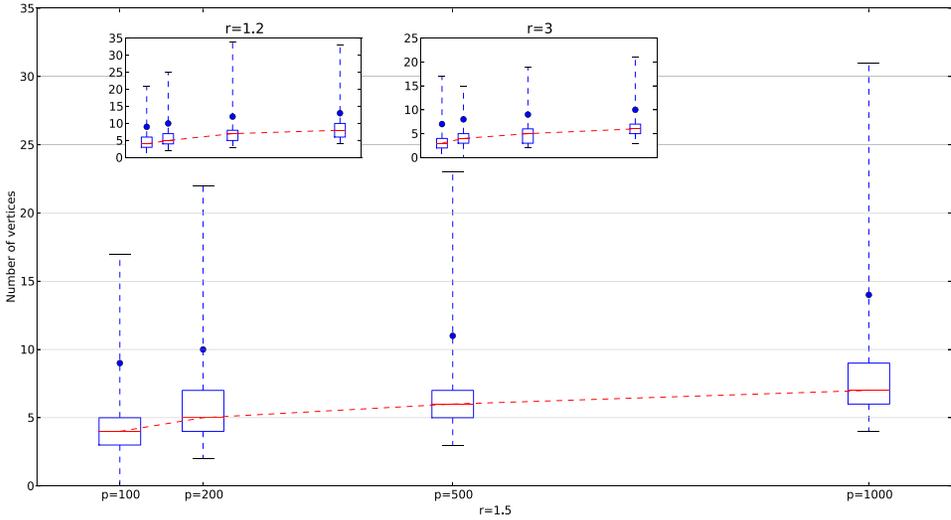


FIG. 4. The distributions of the maximum sizes of chain components of completed PDAGs in  $S_p^{r,p}$ . The lines in the boxes and the solid circles above the boxes indicate the medians and the 95% quartiles, respectively.

and 1000, respectively. This shows that the maximum size of chain components in a competed PDAG increases very slowly with  $p$ . In particular, from the 95% quartiles (solid circles above boxes), we can see that the maximum chain components of more than 95% completed PDAGs in  $S_p^{1.5p}$  have at most 8, 9, 10 and 13 vertices for  $p = 100, 200, 500$  and  $1000$ , respectively. This result implies that sizes of chain components in most sparse completed PDAGs are small.

4.3. *Comparisons between Algorithm 1 and its accelerated version.* In this section, we show experimentally that the accelerated version Algorithm 2 is much faster than Algorithm 1, and the difference of estimates based on two algorithms is small. We have estimated four distributions on  $S_{100}^{150}$  in Section 4.2 via Algorithm 2. The four distributions are the distribution of proportions of directed edges, the distribution of the numbers of chain components, the distribution of maximum size of chain components and the distribution of the numbers of  $v$ -structures. To compare Algorithm 1 with Algorithm 2, we re-estimate these four distributions for completed PDAGs in  $S_{100}^{150}$  via Algorithm 1.

For each distribution, in Figure 5, we report the estimates obtained by Algorithm 1 with lines and the estimates obtained by Algorithm 2 with points in the main windows. The differences of two estimates are shown in the sub-windows. The top panel of Figure 5 displays the cumulative distributions of proportions of directed edges. The second panel of this figure displays the distributions of the numbers of chain components. The third panel displays the distributions of maximum size of chain components. The bottom panel displays the distribution of the

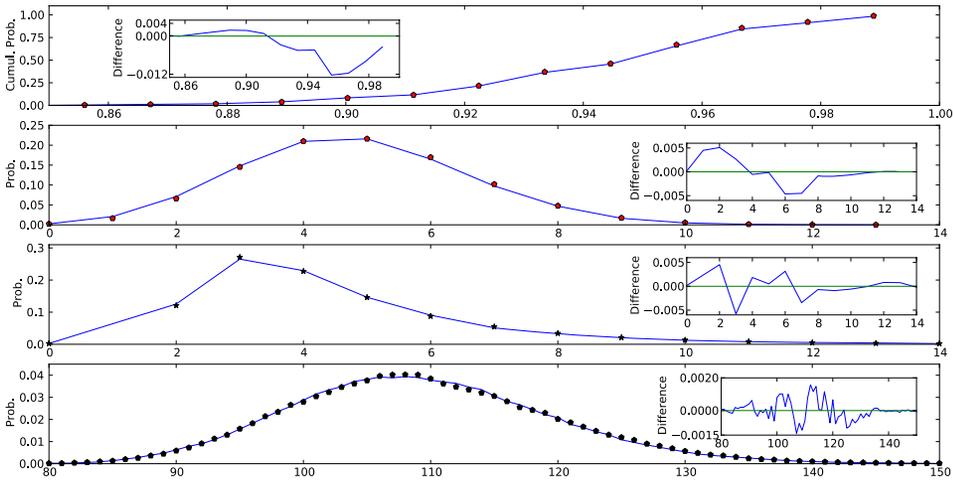


FIG. 5. Distributions for completed PDAGs in  $\mathcal{S}_{100}^{150}$  estimated via Algorithm 1 (plotted in lines) and the accelerated version—Algorithm 2 (plotted in points) are shown in the main windows. The differences are shown in sub-windows. Four panels (from top to bottom) display distributions of directed edges, number of chain components, maximum size of chain components and  $v$ -structures, respectively.

numbers of  $v$ -structures. We can see that the differences of three pairs of estimates are small.

The average times used to generate a state of the Markov chain of completed PDAGs in  $\mathcal{S}_p^{1.5p}$  are shown in Table 2, in which  $\alpha$  is the acceleration parameter used in Algorithm 2. If  $\alpha = 1$ , the Markov chain is generated via Algorithm 1. The results suggest that the accelerated version can speed up the approach nearly  $\frac{1}{\alpha}$  times when  $p = 100$ .

4.4. *Asymptotic properties of proposed estimators.* We further illustrate the asymptotic properties of proposed estimators of sparse completed PDAGs via simulation studies. We consider  $\mathcal{S}_p^{1.5p}$  for  $p = 100, 200, 500$  and  $1000$ , respectively. Let  $f(u)$  be a discrete function of Markov equivalence class  $u$ , where  $u$  is a random variable distributed uniformly in  $\mathcal{S}_p^{1.5p}$ . Let  $\mathbb{E}(f)$  be the expectation of  $f(u)$ ,

TABLE 2  
 The average time used to generate a completed PDAG in  $\mathcal{S}_p^{1.5p}$ ,  
 where  $p$  is the number of vertices,  $\alpha$  is the acceleration parameter,  
 $\kappa$  is the average time (seconds)

$p$	100	100	200	500	1000
$\alpha$	1	0.1	0.05	0.01	0.001
$\kappa$ (seconds)	0.22	0.032	0.113	0.28	0.72

and we have

$$\mathbb{E}(f) = \sum_i i \mathbb{P}(f = i).$$

Proposition 2 shows that the estimator  $\hat{\mathbb{P}}(f = i)$  in equation (2.6) converges to  $\mathbb{P}(f = i)$  with probability one. We also have that the estimator defined as

$$\hat{\mathbb{E}}(f) = \sum_i i \hat{\mathbb{P}}(f = i) = \frac{\sum_i \sum_{t=1}^N i I_{\{f(e_t)=i\}} M_t^{-1}}{\sum_{t=1}^N M_t^{-1}} = \frac{\sum_{t=1}^N f(e_t) M_t^{-1}}{\sum_{t=1}^N M_t^{-1}}$$

converges to  $\mathbb{E}(f)$  with probability one, where  $\{e_t, M_t\}_{t=1, \dots, N}$  is a Markov chain from Algorithm 1.

We generate some sequences of Markov equivalence classes  $\{e_t, \hat{M}_t\}$  with length of  $N = 1.25 \times 10^6$  via Algorithm 2 and divide each sequence into 250 blocks. Set  $f(u)$  to be the proportion of directed edges in  $u$ , we estimate  $\mathbb{E}(f)$  using cumulative data in the first  $k$  blocks as

$$\hat{\mathbb{E}}(f)_k = \left( \sum_{t=1}^{k \times j} f(e_t) \hat{M}_t^{-1} \right) / \sum_{t=1}^{k \times j} \hat{M}_t^{-1},$$

where  $j = 5 \times 10^3$ . The simulation results are shown in Figure 6. We can see that the estimates of proportions of directed edges converge quickly as  $k$  increases.

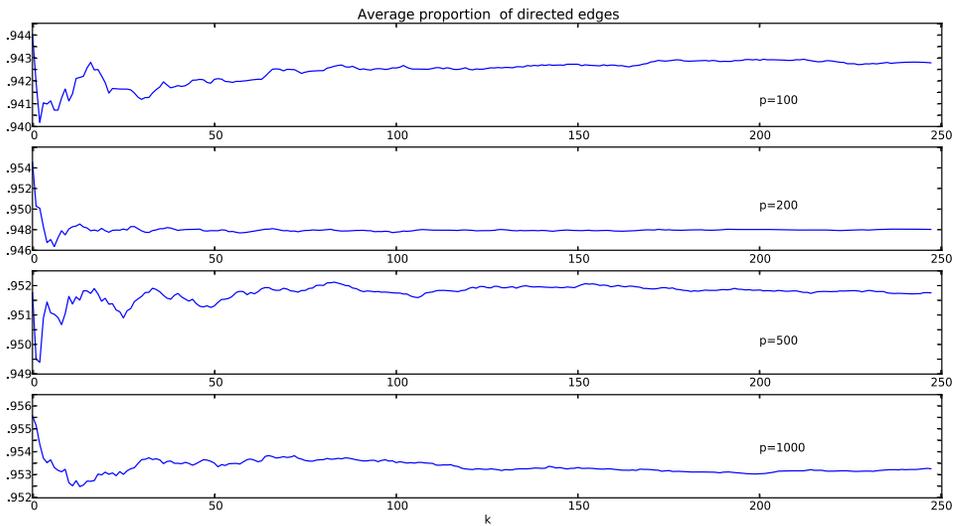


FIG. 6. Four sequences of average proportions of directed edges in completed PDAGs in  $\mathcal{S}_p^{1.5p}$  with  $p = 100, 200, 500$  and  $1000$ , estimated via Algorithm 2 and the first  $5000k$  steps of the Markov chains, where  $k$  is shown in x-axis.

**5. Conclusions and discussions.** In this paper, we proposed a reversible irreducible Markov chain on Markov equivalence classes that can be used to study various properties of a given set of interesting Markov equivalence classes. Our experiments on Markov equivalence classes with sparse constraints reveal useful information. For example, we find that proportions of undirected edges and chain components in sparse completed PDAGs are small even for Markov equivalence classes with thousands of vertices.

When some “important” but very rare equivalence classes are of interest, it will be very hard to sample them in the proposed Markov chain. In this case, we can constrain the space appropriately so that these Markov equivalence classes are easy to be sampled. For example, it is nearly impossible to sample equivalence classes with 300 vertices and 1 edge from  $\mathcal{S}_{300}$ . Fortunately, if we set the space to be  $\mathcal{S}_{300}^2$ , sampling graphs with 1 edge is not difficult.

The sizes of Markov equivalence classes are the property most widely discussed in the literature. Due to space constraints, we have omitted several details in this paper about determining the size of Markov equivalence classes and calculating further properties of edges and vertices. We will discuss these issues in a follow-up paper. The proposed methods can potentially be extended to study other sets of completed PDAGs besides  $\mathcal{S}_n^p$ . Some interesting sets include (1) the completed PDAGs in which each vertex has at most  $d$  adjacent edges; (2) completed PDAGs in which each pair of vertices is connected by a path along edges in the graph.

## APPENDIX: PRELIMINARY RESULTS AND PROOF OF THEOREM 1

In this Appendix, we provide two preliminary results introduced by Andersson [2] and Chickering [5, 6], respectively, in Appendix A.1. These results are necessary to implement our proposed approach technically and will be used in the proof of Theorem 1. Then we provide a proof of the main result of this paper (Theorem 1) in Appendix A.2.

**A.1. Two preliminary results.** Some definitions and notation are introduced first. A graph is called a *chain graph* if it contains no partially directed cycles [22]. A *chord* of a cycle is an edge that joins two nonadjacent vertices in the cycle. An undirected graph is *chordal* if every cycle of length greater than or equal to 4 possesses a chord. A directed edge of a DAG is *compelled* if it occurs in the corresponding completed PDAG, otherwise, the directed edge is *reversible*, and the corresponding parents are reversible parents. Recall  $N_x$  be the set of all neighbors of  $x$ ,  $\Pi_x$  is the set of all parent of  $x$ ,  $N_{xy} = N_x \cap N_y$  and  $\Omega_{x,y} = \Pi_x \cap N_y$  and the concept of “strongly protected” is presented in Definition 2.

Lemma 2 characterizes completed PDAGs that are used to represent Markov equivalence classes [2] and will be used in the proofs in Appendix A.2.

LEMMA 2 (Andersson [2]). *A graph  $C$  is a completed PDAG of a directed acyclic graph  $\mathcal{D}$  if and only if  $C$  satisfies the following properties:*

- (i)  $\mathcal{C}$  is a chain graph;
- (ii) let  $\mathcal{C}_\tau$  be the subgraph induced by  $\tau$ .  $\mathcal{C}_\tau$  is chordal for every chain component  $\tau$ ;
- (iii)  $w \rightarrow u - v$  does not occur as an induced subgraph of  $\mathcal{C}$ ;
- (iv) every arrow  $v \rightarrow u$  in  $\mathcal{C}$  is strongly protected.

Lemma 3 shows the equivalent validity conditions for  $\mathbf{iu}_2$ ,  $\mathbf{du}_1$ ,  $\mathbf{id}_2$ ,  $\mathbf{dd}_1$  and  $\mathbf{mv}_1$  used in Definition 9.

LEMMA 3 (Validity conditions of some operators [6]). *The necessary and sufficient validity conditions of the operators with type of InsertU, DeleteU, InsertD, DeletedD or MakeV are as follows:*

- (InsertU) Let  $x$  and  $y$  be two vertices that are not adjacent in  $\mathcal{C}$ . The operator  $\text{InsertU } x - y$  is valid (equivalently,  $\mathbf{iu}_2$  holds) if and only if (iu2.1)  $\Pi_x = \Pi_y$ , (iu2.2) every undirected path from  $x$  to  $y$  contains a vertex in  $N_{xy}$ .
- (DeleteU) Let  $x - y$  be an undirected edge in completed PDAG  $\mathcal{C}$ . The operator  $\text{DeleteU } x - y$  is valid (equivalently,  $\mathbf{du}_1$  holds) if and only if (du1.1)  $N_{xy}$  is a clique in  $\mathcal{C}$ .
- (InsertD) Let  $x$  and  $y$  be two vertices that are not adjacent in  $\mathcal{C}$ . The operator  $\text{InsertD } x \rightarrow y$  is valid (equivalently,  $\mathbf{id}_2$  holds) if and only if (id2.1)  $\Pi_x \neq \Pi_y$ , (id2.2)  $\Omega_{x,y}$  is a clique, (id2.3) every partially directed path from  $y$  to  $x$  contains at least one vertex in  $\Omega_{x,y}$ .
- (DeletedD) Let  $x \rightarrow y$  be a directed edge in completed PDAG  $\mathcal{C}$ . The operator  $\text{DeletedD}$  of  $x \rightarrow y$  is valid (equivalently,  $\mathbf{dd}_1$  holds) if and only if (dd1.1)  $N_y$  is a clique.
- (MakeV) Let  $x - z - y$  be any length-two undirected path in  $\mathcal{C}$  such that  $x$  and  $y$  are not adjacent. The operator  $\text{MakeV } x \rightarrow z \leftarrow y$  is valid (equivalently,  $\mathbf{mv}_1$  holds) if and only if (mv1.1) every undirected path between  $x$  and  $y$  contains a vertex in  $N_{xy}$ .

**A.2. Proof of Theorem 1.** Let  $\mathcal{O}$  be the operator set defined in equation (3.3); to prove Theorem 1, which shows  $\mathcal{O}$  is a perfect operator set, we need to show  $\mathcal{O}$  satisfies four properties: validity, distinguishability, irreducibility and reversibility. Equivalently, we just need to prove Theorem 2–5 as follows:

THEOREM 2. *The operator set  $\mathcal{O}$  is valid.*

THEOREM 3. *The operator set  $\mathcal{O}$  is distinguishable.*

THEOREM 4. *The operator set  $\mathcal{O}$  is reversible.*

THEOREM 5. *The operator set  $\mathcal{O}$  is irreducible.*

Of the above four theorems, the most important and difficult is to prove Theorem 4. We now show the proofs one by one.

**PROOF OF THEOREM 2.** According to the definition of validity in Definition 5 and the definition of  $\mathcal{O}_C$  in equation (3.2), all operators in  $InsertU_C$ ,  $DeleteU_C$ ,  $InsertD_C$ ,  $DeleteD_C$  and  $MakeV_C$  are valid. We just need to prove Lemma 4, which shows all operators in  $RemoveV_C$  are valid.  $\square$

**LEMMA 4.** *Let  $x \rightarrow z \leftarrow y$  be a  $v$ -structure in completed PDAG  $\mathcal{C}$ . If  $(\mathbf{rv}_1) \Pi_x = \Pi_y$ ,  $(\mathbf{rv}_2) \Pi_x \cup N_{xy} = \Pi_z \setminus \{x, y\}$ , and  $(\mathbf{rv}_3)$  every undirected path between  $x$  and  $y$  contains a vertex in  $N_{xy}$  hold, then the operator  $RemoveV$   $x \rightarrow z \leftarrow y$  is valid and results in a completed PDAG in  $S_p^n$  defined in equation (3.1).*

To prove Lemma 4, we will use Lemma 5 given by Chickering (Lemma 32 in [6]).

**LEMMA 5.** *Let  $\mathcal{C}$  be any completed PDAG, and let  $x$  and  $y$  be any pair of vertices that are not adjacent. Every undirected path between  $x$  and  $y$  passes through a vertex in  $N_{xy}$  if and only if there exists a consistent extension in which (1)  $x$  has no reversible parents, (2) all vertices in  $N_{xy}$  are parents of  $y$  and (3)  $y$  has no other reversible parents.*

We now give a proof of Lemma 4.

**PROOF OF LEMMA 4.** From Lemma 5 and condition  $\mathbf{rv}_3$  in Lemma 4, there exists a consistent extension of  $\mathcal{C}$ , denoted by  $\mathcal{D}$ , in which  $x$  has no reversible parents, and the reversible parents of  $y$  are the vertices in  $N_{xy}$ . Because  $y \rightarrow z$  occurs in the completed PDAG,  $\mathcal{C}$ ,  $N_x$  and  $N_y$  occur in different chain components. We can orient the undirected edges adjacent to  $z$  out of  $z$ . Then all vertices in  $N_z$  are children of  $z$  in  $\mathcal{D}$ . Let  $\mathcal{D}'$  be the graph obtained by reversing  $y \rightarrow z$  in  $\mathcal{D}$  and  $\mathcal{P}'$  be the PDAG obtained by applying the  $RemoveV$  operator to  $\mathcal{C}$ . We will show that  $\mathcal{D}'$  is a consistent extension of  $\mathcal{P}'$ .

Clearly,  $\mathcal{D}'$  and  $\mathcal{P}'$  have the same skeleton.

We have that any  $v$ -structure that occurs in  $\mathcal{D}$  but not in  $\mathcal{P}'$  must include either the edge  $x \rightarrow z$  or  $y \rightarrow z$ . Since  $\mathcal{D}$  is a consistent extension of  $\mathcal{C}$ , we have that all  $v$ -structures in  $\mathcal{D}$  are also in  $\mathcal{C}$ . From condition  $\mathbf{rv}_2$ , all parents of  $z$  other than  $x$  and  $y$  are adjacent to  $x$  and  $y$ . Hence  $x \rightarrow z \leftarrow y$  is the only  $v$ -structure that is directed into  $z$  in  $\mathcal{C}$ . We have that all  $v$ -structures of  $\mathcal{P}'$  are also in  $\mathcal{D}$ , and there is only one  $v$ -structure  $x \rightarrow z \leftarrow y$  that is in  $\mathcal{D}$  but not  $\mathcal{P}'$ .

Since  $y \rightarrow z$  is the unique edge that differs between  $\mathcal{D}$  and  $\mathcal{D}'$ , we have that any  $v$ -structure that exists in  $\mathcal{D}$  but not in  $\mathcal{D}'$  must include the edge  $y \rightarrow z$ , and any  $v$ -structure that exists in  $\mathcal{D}'$  but not in  $\mathcal{D}$  must include the edge  $z \rightarrow y$ . We have shown that  $x \rightarrow z \leftarrow y$  is the only  $v$ -structure in  $\mathcal{D}$  that is directed into  $z$ . From the

construction of  $\mathcal{D}$ , we have that all compelled parents of  $y$  in  $\mathcal{D}'$  are also parents of  $z$ , and all other parents are in  $N_{xy}$ ; from  $\mathbf{rv}_2$ , they also are parents of  $z$ . There is no  $v$ -structure that includes edge  $z \rightarrow y$  in  $\mathcal{D}'$ . Hence, all  $v$ -structures of  $\mathcal{D}'$  are also in  $\mathcal{D}$ , and there is only one  $v$ -structure  $x \rightarrow z \leftarrow y$  that is in  $\mathcal{D}$  but not  $\mathcal{D}'$ .

Hence,  $\mathcal{D}'$  and  $\mathcal{P}'$  have the same  $v$ -structures. It remains to be shown that  $\mathcal{D}'$  is acyclic.

If  $\mathcal{D}'$  contains a cycle, the cycle must contain the edges  $z \rightarrow y$  because  $\mathcal{D}$  is acyclic. This implies there is a directed path from  $y$  to  $z$  in  $\mathcal{D}$ . By construction, all vertices in  $N_z$  are children of  $z$  in  $\mathcal{D}'$ . So, this path must include a compelled parent of  $z$ ; denote it by  $u$ . If  $u \neq x$ , from condition  $\mathbf{rv}_2$ ,  $u \in \Pi_y \cup N_{xy}$ ; by the construction of  $\mathcal{D}$ , we have  $u \in \Pi_y$ . Thus, there is no path from  $y$  to  $z$  that contains  $u$ . If  $u = x$ , by construction, the path must contain a compelled parent  $v$  of  $x$ . From condition  $\mathbf{rv}_1$ ,  $v \in \Pi_y$ . Thus, there is no path from  $y$  to  $z$  contains  $v$ . We get that  $\mathcal{D}'$  is acyclic. Thus  $\mathcal{D}'$  is a consistent extension of  $\mathcal{P}'$  and the operator  $\text{RemoveV } x \rightarrow z \leftarrow y$  is valid.  $\square$

**PROOF OF THEOREM 3.** For any completed  $\mathcal{C} \in \mathcal{S}_p^n$ , we need to show that different operators in  $\mathcal{O}_C$  result in different completed PDAGs. For any valid operator  $o \in \text{InsertU}_C$ , say  $\text{InsertU } x - y$ , denoted as  $o$ , the resulting completed PDAG of  $o$  contains the undirected edge  $x - y$ . We have that all other operators in  $\mathcal{O}_C$  except for  $\text{InsertD } x \rightarrow y$  and  $\text{Insert } x \leftarrow y$  (if they are also valid) will result in completed PDAGs with skeletons different than the resulting completed PDAG of  $o$ . Thus, these operators cannot result in the same completed PDAG as  $o$ . If  $\text{InsertD } x \rightarrow y$  or  $\text{Insert } x \leftarrow y$  is valid, the resulting completed PDAGs of them contain  $x \rightarrow y$  or  $x \leftarrow y$ . These two resulting completed PDAGs have at least a compelled edge different than the resulting completed PDAG of  $o$ . Thus there is no operator in  $\mathcal{O}_C$  that can result in the same completed PDAG as  $o$ .

Similarly, we can show for any operator in  $\mathcal{O}_C$ , different operators will result in different completed PDAGs because they will have distinct skeletons, compelled edges or  $v$ -structures.  $\square$

**PROOF OF THEOREM 4.** Let  $\mathcal{C}$  be any completed PDAG in  $\mathcal{S}_p^n$ ,  $o \in \mathcal{O}_C$  be an operator on  $\mathcal{C}$ . The operator  $o' \in \mathcal{O}$  is the *reversible operator* of  $o$  if  $o'$  can transfer the resulting completed PDAG of  $o$  back to  $\mathcal{C}$ . To prove Theorem 4, we just need to show each operator in  $\mathcal{O}_C$  defined in equation (3.3) has a reversible operator in  $\mathcal{O}$ . Equivalently, we prove Lemmas 6, 7, 8, 9, 10 and 11 to show the reversibility for six types of operators, respectively.  $\square$

**LEMMA 6.** For any operator  $o \in \mathcal{O}_C$  denoted by “ $\text{InsertU } x - y$ ,” the operator “ $\text{DeleteU } x - y$ ” is the reversible operator of  $o$ .

**LEMMA 7.** For any operator  $o \in \mathcal{O}_C$  denoted by “ $\text{DeleteU } x - y$ ,” the operator “ $\text{InsertU } x - y$ ” is the reversible operator of  $o$ .

LEMMA 8. For any operator  $o \in \mathcal{O}_C$  denoted by “InsertD  $x \rightarrow y$ ,” the operator “DeleteD  $x \rightarrow y$ ” is the reversible operator of  $o$ .

LEMMA 9. For any operator  $o \in \mathcal{O}_C$  denoted by “DeleteD  $x \rightarrow y$ ,” the operator “InsertD  $x \rightarrow y$ ” is the reversible operator of  $o$ .

LEMMA 10. For any operator  $o \in \mathcal{O}_C$  denoted by “MakeV  $x \rightarrow z \leftarrow y$ ,” the operator “RemoveV  $x \rightarrow z \leftarrow y$ ” is the reversible operator of  $o$ .

LEMMA 11. For any operator  $o \in \mathcal{O}_C$  denoted by “RemoveV  $x \rightarrow z \leftarrow y$ ,” the operator “MakeV  $x \rightarrow z \leftarrow y$ ” is the reversible operator of  $o$ .

Before giving proofs of these six lemmas, We first provide several results shown in Lemmas 12, 13, 14 and 15.

LEMMA 12. Let graph  $\mathcal{C}$  be a completed PDAG,  $\{w, v, u\}$  be three vertices that are adjacent each other in  $\mathcal{C}$ . If there are two undirected edges in  $\{w, v, u\}$ , then the third edge is also undirected.

PROOF. If the third edge is directed, there is a directed cycle like  $w - v - u \rightarrow w$ . From Lemma 2, we know that  $\mathcal{C}$  is a chain graph, so there is no directed circle in  $\mathcal{C}$ .  $\square$

LEMMA 13. Let  $\mathcal{C}_1$  be the resulting completed PDAG obtained by inserting a new edge between  $x$  and  $y$  in  $\mathcal{C}$ . If there is at least one edge  $v \rightarrow u$  that is directed in  $\mathcal{C}$  but not directed in  $\mathcal{C}_1$ , then there exists a vertex  $h$  that is common child of  $x$  and  $y$  such that  $x \rightarrow h$  and  $y \rightarrow h$  in  $\mathcal{C}$  become undirected in  $\mathcal{C}_1$ .

PROOF. According to Lemma 2, an edge is directed in a completed PDAG if and only if it is strongly protected. Thus, we have that at least one case among (a), (b), (c), (d) in Figure 1 occurs in  $\mathcal{C}$  but not in  $\mathcal{C}_1$  for  $v \rightarrow u$ . We will show that either Lemma 13 holds, or there exists a parent of  $u$ , denoted as  $u_1$ , such that  $u_2 \rightarrow u_1$  occurs in  $\mathcal{C}$  but not in  $\mathcal{C}_1$ , where  $u_2$  is a parent of  $u_1$ . We denote the latter result as (\*).

Suppose case (a) in Figure 1 occurs in  $\mathcal{C}$  but not in  $\mathcal{C}_1$ . Because  $v \rightarrow u$  becomes undirected in  $\mathcal{C}_1$ , we have that  $w \rightarrow v$  must be undirected in  $\mathcal{C}_1$  since  $w$  and  $u$  are not adjacent. Set  $u_1 = v$  and  $u_2 = u$ , and we have that (\*) holds.

Suppose case (b) in Figure 1 occurs in  $\mathcal{C}$  but not in  $\mathcal{C}_1$ . If the pair  $\{v, w\}$  is not  $\{x, y\}$ ,  $v \rightarrow u \leftarrow w$  is a  $v$ -structure in  $\mathcal{C}$ . We have that  $v \rightarrow u$  occurs in  $\mathcal{C}_1$ . This is a contradiction. If  $\{v, w\}$  is  $\{x, y\}$ , we have that Lemma 13 holds ( $h = u$ ).

Suppose case (c) in Figure 1 occurs in  $\mathcal{C}$  but not in  $\mathcal{C}_1$ . Either  $v \rightarrow w$  or  $w \rightarrow u$  occurs in  $\mathcal{C}$  but not in  $\mathcal{C}_1$ . If it is  $v \rightarrow w$ , by setting  $u_2 = v$  and  $u_1 = w$ , we have

(\*) holds. If it is  $w \rightarrow u$ , both  $v - u$  and  $w - u$  in  $\mathcal{C}_1$ , so  $x - u$  also must be in  $\mathcal{C}_1$ . We also have that (\*) holds.

Suppose case (d) in Figure 1 occurs in  $\mathcal{C}$  but not in  $\mathcal{C}_1$ . If the pair  $\{w, w_1\}$  is  $\{x, y\}$ , Lemma 13 holds ( $h = u$ ). Otherwise,  $w \rightarrow u \leftarrow w_1$  must occur in both  $\mathcal{C}_1$  and  $\mathcal{C}$  and the edge  $v \rightarrow u$  is still strongly protected in  $\mathcal{C}_1$ , yielding a contradiction.

If (\*) holds, we have that there is a directed path  $u_2 \rightarrow u_1 \rightarrow u$  such that  $u_2 \rightarrow u_1$  occurs in  $\mathcal{C}$  but not  $\mathcal{C}_1$ . Iterating, we can get a directed path  $u_k \rightarrow u_{k-1} \cdots \rightarrow u$  of length  $k - 1$  without undirected edges such that  $u_k \rightarrow u_{k-1}$  occurs in  $\mathcal{C}$  but not in  $\mathcal{C}_1$  if Lemma 13 does not hold in each step. Because  $\mathcal{C}$  is a chain graph without directed circle, the procedure will stop in finite steps and Lemma 13 will hold eventually.  $\square$

From the proof of Lemma 13, we have that  $u$  should be a descendant of  $x$  and  $y$ , so we can get the following Lemma 14.

LEMMA 14. *Let  $\mathcal{C}$  be any completed PDAG, and let  $\mathcal{P}$  denote the PDAG that results from adding a new edge between  $x$  and  $y$ . For any edge  $v \rightarrow u$  in  $\mathcal{C}$  that does not occur in the resulting completed PDAG extended from  $\mathcal{P}$ , there is a directed path of length zero or more from both  $x$  and  $y$  to  $u$  in  $\mathcal{C}$ .*

LEMMA 15. *Let  $InsertU_{\mathcal{C}}$  and  $DeleteU_{\mathcal{C}}$  be the operator sets defined in Definition 9, respectively. For any  $o$  in  $InsertU_{\mathcal{C}}$  or in  $DeleteU_{\mathcal{C}}$ , where  $\mathcal{P}'$  is the modified graph of  $o$  that is obtained by applying  $o$  to  $\mathcal{C}$ , we have that  $\mathcal{P}'$  is a completed PDAG.*

PROOF. We just need to check whether  $\mathcal{P}'$  satisfies the four conditions in Lemma 2.

(i): For any  $o \in DeleteU_{\mathcal{C}}$ , denoted as  $DeleteD\ x - y$ , let  $\mathcal{P}'$  be the modified graph obtained by deleting  $x - y$  from  $\mathcal{C}$ .

If there is a directed cycle in  $\mathcal{P}'$ , it must be a directed cycle in  $\mathcal{C}$ , which is a contradiction. Thus there is no directed cycle in  $\mathcal{P}'$ , and  $\mathcal{P}'$  is a chain graph.

If there exists an undirected cycle of length greater than 3 without a chord in  $\mathcal{P}'$ , the cycle must contain both  $x$  and  $y$ ; otherwise, this cycle occurs in  $\mathcal{C}$ . If the length of the cycle is 4, the other two vertices are in  $N_{xy}$ ; we have that the cycle has a chord since  $N_{xy}$  is a clique in  $\mathcal{C}$ . If the cycle in  $\mathcal{P}'$  has length greater than 4 without a chord, we have that  $x - y$  is the unique chord of this cycle in  $\mathcal{C}$ . However, this would imply that there is a cycle of length greater than 3 without a chord in  $\mathcal{C}$ , a contradiction. Thus, there is no undirected cycle with length greater than 3 in  $\mathcal{P}'$ , so every chain component of  $\mathcal{P}'$  is chordal.

Suppose that  $\cdot \rightarrow \cdot - \cdot$  occurs as an induced subgraph of  $\mathcal{P}'$ ; it must be  $x \rightarrow \cdot - y$  (or  $y \rightarrow \cdot - x$ ). However, in this case,  $x \rightarrow \cdot - y - x$  (or  $y \rightarrow \cdot - x - y$ ) would be a directed cycle in  $\mathcal{C}$ . Thus the induced subgraph like  $\cdot \rightarrow \cdot - \cdot$  does not occur as an induced subgraph of  $\mathcal{P}'$ .

Finally, all directed edges in  $\mathcal{P}'$  will be strongly protected; by the definition of strong protection, all directed edges in  $\mathcal{C}$  will remain strongly protected when an undirected edge is removed.

(ii): For any  $o \in \text{InsertU}_{\mathcal{C}}$ , denoted as  $\text{InsertU } x - y$ ,  $\mathcal{P}'$  is the modified graph of  $o$ .

If there is a directed cycle in  $\mathcal{P}'$ , it must contain  $x - y$ ; otherwise this cycle is also in  $\mathcal{C}$ . We can suppose that there exists a partially directed path from  $x$  to  $y$  in  $\mathcal{C}$ . Denote the adjacent vertex of  $y$  in the path as  $u$ . Let  $u$  be the vertex adjacent to  $y$  in the path. We have  $u \notin \Pi_y$ ; otherwise, from the condition  $\Pi_x = \Pi_y$  in Lemma 3,  $u$  would also be in  $\Pi_x$ , so there would be a partially directed cycle from  $x$  to  $x$  in  $\mathcal{C}$ . Hence the directed path must have the form  $x \cdots \rightarrow \cdots u - y$ . This would induce a subgraph like  $a \rightarrow b - v$  in  $\mathcal{C}$ , a contradiction. Consequently,  $\mathcal{P}'$  is a chain graph.

If there exists an undirected cycle of length greater than 3 without a chord in  $\mathcal{P}'$ , the cycle must contain  $x$  and  $y$ , and there must be an undirected path from  $x$  to  $y$  in  $\mathcal{C}$ ; otherwise, the cycle would also be in  $\mathcal{C}$ . From Lemma 3, every undirected path from  $x$  to  $y$  contains a vertex in  $N_{xy}$ , so every undirected path of length greater than two has a chord. Thus, every undirected path of length greater than 3 from  $x$  to  $y$  in  $\mathcal{P}'$  has a chord. This implies that every chain component of  $\mathcal{P}'$  is chordal.

Suppose that a subgraph like  $\cdot \rightarrow \cdot - \cdot$  occurs as an induced subgraph of  $\mathcal{P}'$ . Since  $\Pi_x = \Pi_y$  in  $\mathcal{C}$ , the induced subgraph is not  $\cdot \rightarrow x - y$  (or  $\cdot \rightarrow y - x$ ). Thus, the induced subgraph like  $\cdot \rightarrow \cdot - \cdot$  also occurs in  $\mathcal{C}$ . This is a contradiction since  $\mathcal{C}$  is a completed PDAG, yielding a contradiction.

From Lemma 13 and the condition  $\mathbf{iu}_3$  in Definition 9, all directed edges in  $\mathcal{C}$  are also directed in  $\mathcal{C}_1$ . This implies that all directed edges in  $\mathcal{P}$  are still compelled, and are thus strongly protected.  $\square$

We now give proofs of Lemmas 6, 7, 8, 9, 10 and 11, one by one.

**PROOF OF LEMMA 6.** Because the operator “ $\text{InsertU } x - y$ ” =  $o \in \mathcal{O}_{\mathcal{C}}$  is valid and  $\mathcal{C}_1$  is the resulting completed PDAG of  $o$ , we have that  $x - y$  occurs in  $\mathcal{C}_1$ . We just need to show that the common neighbors of  $x$  and  $y$ , denoted as  $N_{xy}$ , form a clique in  $\mathcal{C}_1$ .

If  $N_{xy}$  is empty set or has only one vertex, the condition that  $N_{xy}$  is a clique in  $\mathcal{C}_1$  holds.

If there are two different vertices  $z, u \in N_{xy}$  in  $\mathcal{C}_1$ , we have that  $x - z - y$  and  $x - u - y$  form a cycle of length of 4 in  $\mathcal{C}_1$ . The cycle is also in  $\mathcal{C}$ . Since the edge  $x - y$  does not exist in  $\mathcal{C}$  and  $\mathcal{C}$  is a completed PDAG in which all undirected subgraphs are chordal graphs, we have that  $z - u$  occurs in  $\mathcal{C}$ , so  $z$  and  $u$  are adjacent in  $\mathcal{C}_1$ . Hence the condition that  $N_{xy}$  is a clique in  $\mathcal{C}_1$  holds.  $\square$

**PROOF OF LEMMA 7.** We need to show the operator  $o' := \text{InsertU } x - y$  satisfies the conditions  $\mathbf{iu}_1$ ,  $\mathbf{iu}_2$  and  $\mathbf{iu}_3$  in Definition 9 for completed PDAG  $\mathcal{C}_1$  and that the resulting completed PDAG of  $o'$  is  $\mathcal{C}$ .

The condition  $\mathbf{iu}_1$  clearly holds, since  $x - y$  exists in  $\mathcal{C}_1$  but not in  $\mathcal{C}$ . Lemma 15 implies that the graph obtained by deleting  $x - y$  from  $\mathcal{C}$  is the completed PDAG  $\mathcal{C}_1$ . Thus, the graph obtained by inserting  $x - y$  into  $\mathcal{C}_1$  is  $\mathcal{C}$ . This implies that  $\text{InsertU } x - y$  is valid, and the condition  $\mathbf{iu}_2$  holds.

Lemma 15 implies that the condition  $\mathbf{iu}_3$  also holds.  $\square$

**PROOF OF LEMMA 8.** I will first show that there is no undirected edge  $y - w$  that occurs in both  $\mathcal{C}$  and  $\mathcal{C}_1$ . If  $w - y$  occurs in  $\mathcal{C}$ , since  $x$  and  $y$  are not adjacent in  $\mathcal{C}$ ,  $x \rightarrow w - y$  does not occur in  $\mathcal{C}$ . There are three possible configurations between  $x$  and  $w$  in  $\mathcal{C}$ : (1)  $x$  is not adjacent to  $w$ , (2)  $w \rightarrow x$  and (3)  $x - w$ . If  $x$  is not adjacent to  $w$  in  $\mathcal{C}$ , inserting  $x \rightarrow y$  will result in  $y \rightarrow w$  in  $\mathcal{C}_1$ . If  $w \rightarrow x$  is in  $\mathcal{C}$ , inserting  $x \rightarrow y$  will result in  $w \rightarrow y$  in  $\mathcal{C}_1$ . If  $x - w$  in  $\mathcal{C}$ , there is an undirected path from  $y$  to  $x$ ; that is, the first condition for  $\text{InsertD}$  to be valid, according to Lemma 3, does not hold. Thus we get that there is no undirected edge  $y - w$  that occurs in both  $\mathcal{C}$  and  $\mathcal{C}_1$ .

For any  $w \in N_y$  in  $\mathcal{C}_1$ , the edge between  $w$  and  $y$  is directed in  $\mathcal{C}$ ; that is, either  $w \rightarrow y$  or  $y \rightarrow w$  occurs in  $\mathcal{C}$ . If  $y \rightarrow w$  is in  $\mathcal{C}$ , there are three possible configurations between  $x$  and  $w$  in  $\mathcal{C}$ : (1)  $x$  is not adjacent to  $w$ , (2)  $w \rightarrow x$  and (3)  $x \rightarrow w$ . If  $x$  and  $w$  are not adjacent in  $\mathcal{C}$ , inserting  $x \rightarrow y$  will result in  $y \rightarrow w$  in  $\mathcal{C}_1$ . If  $w \rightarrow x$  occurs in  $\mathcal{C}$ , inserting  $x \rightarrow y$  is not valid for  $\mathcal{C}$  since there would be a directed path from  $y$  to  $x$ . If  $x \rightarrow w$  occurs in  $\mathcal{C}$ ,  $w$  is common child of  $x$  and  $y$ , so from condition  $\mathbf{id}_3$ ,  $y \rightarrow w$  occurs in  $\mathcal{C}_1$  and  $w \notin N_y$  in  $\mathcal{C}_1$ . Thus, we have that  $w \rightarrow y$  must be in  $\mathcal{C}$ .

If there is another vertex  $v \in N_y$  in  $\mathcal{C}_1$ ,  $v \rightarrow y$  must also be in  $\mathcal{C}$ . If  $v$  and  $w$  are not adjacent,  $v \rightarrow y \leftarrow w$  forms a  $v$ -structure both in  $\mathcal{C}$  and in  $\mathcal{C}_1$ .  $w \rightarrow y$  must occur in  $\mathcal{C}_1$  and, consequently,  $w \notin N_y$  in  $\mathcal{C}_1$  yielding a contradiction. Thus, we know that any two vertices in  $N_y$  are adjacent in  $\mathcal{C}$ .  $N_y$  is therefore a clique in  $\mathcal{C}_1$ , and the operator  $\text{DeleteD } x \rightarrow y$  is valid for  $\mathcal{C}_1$ ; that is, the condition  $\mathbf{id}_1$  in Definition 9 holds.

Denote the modified PDAG of operator  $\text{DeleteD } x \rightarrow y$  of  $\mathcal{C}_1$  as  $\mathcal{P}'$ . We need to show that the corresponding completed PDAG of  $\mathcal{P}'$  is  $\mathcal{C}$ . Equivalently, we just need to show  $\mathcal{P}'$  and  $\mathcal{C}$  have the same skeleton and  $v$ -structures. Clearly,  $\mathcal{P}'$  and  $\mathcal{C}$  have the same skeleton. If there is a  $v$ -structure in  $\mathcal{C}$ , but not in  $\mathcal{C}_1$ , it must be  $x \rightarrow u \leftarrow y$ , where  $u$  is a common child of  $x$  and  $y$ . From condition  $\mathbf{id}_3$  in Definition 9,  $x \rightarrow u$  and  $y \rightarrow u$  also occur in  $\mathcal{C}_1$ , so, these  $v$ -structures also exist in  $\mathcal{P}'$ . This implies that all  $v$ -structures of  $\mathcal{C}$  are also in  $\mathcal{P}'$ . Moreover, the  $v$ -structures in  $\mathcal{C}_1$  but not in  $\mathcal{C}$  must be  $x \rightarrow y \leftarrow v$ , where  $v$  is parent of  $y$ , and  $x$  and  $v$  are not adjacent in  $\mathcal{C}_1$ . Clearly, after we delete  $x \rightarrow y$  from  $\mathcal{C}_1$ , these  $v$ -structures will not exist in  $\mathcal{P}'$ . This implies that all  $v$ -structures of  $\mathcal{P}'$  are in  $\mathcal{C}$ . So,  $\mathcal{P}'$  and  $\mathcal{C}$  have the same  $v$ -structures.

For any  $v \rightarrow y$  in  $\mathcal{C}_1$ , if  $v - y$  is in  $\mathcal{C}$ ,  $v$  must be parent of  $x$ . If  $x$  and  $v$  are not adjacent, inserting  $x \rightarrow y$  to  $\mathcal{C}$  will result in  $y \rightarrow v$  in  $\mathcal{C}_1$ . Moreover,  $x - v - y$  does not exist in  $\mathcal{C}$  since  $\text{InsertD } x \rightarrow y$  is a valid operator, and  $x \rightarrow v - y$  does not

occur in  $\mathcal{C}$ . Thus, for any  $v$  that is a parent of  $y$  but not a parent of  $x$ , the directed edge  $v \rightarrow y$  also occurs in the resulting completed PDAG  $\mathcal{C}$ . That is, the condition  $\text{id}_2$  in Definition 9 holds.  $\square$

**PROOF OF LEMMA 9.** To prove this lemma, we first introduce Lemmas 16 and 17. Let  $L = (u_1, u_2, \dots, u_k)$  be a partially directed path from  $u_1$  to  $u_k$  in a graph. A path  $L_2 = (u^1, \dots, u^k)$  is a sub-path of  $L_1$  if all vertices in  $L_1$  are in  $L$  and have the same order as in  $L$ . We say that a partially directed path is shortest if it has no smaller sub-path.  $\square$

**LEMMA 16.** *Let  $\mathcal{C}$  be a completed PDAG, and let  $L_1$  be a partially directed path from  $y$  to  $x$  in  $\mathcal{C}$ . Then there exists a shortest sub-path of  $L_1$ , denoted as  $L_2 = y - u_1 - \dots - u_k \rightarrow \dots \rightarrow x$ , in which there exists a  $k$  such that all edges occurring before  $u_k$  in the path are undirected, and all edges occurring after  $u_k$  are directed.*

**PROOF.** We just need to show that a directed edge must be followed by a directed edge in the shortest sub-path. If not,  $u_i \rightarrow u_{i+1} - u_{i+2}$  occurs in  $L_2$ . Because  $\mathcal{C}$  is a completed PDAG,  $u_i$  and  $u_{i+2}$  must be adjacent; otherwise  $u_{i+1} \rightarrow u_{i+2}$  occurs in  $\mathcal{C}$ . If  $u_i \rightarrow u_{i+2}$  occurs in  $\mathcal{C}$ ,  $L_2$  is not a shortest path. If  $u_i \leftarrow u_{i+2}$  occurs in  $\mathcal{C}$ ,  $u_{i+1} \leftarrow u_{i+2}$  must be in  $\mathcal{C}$ .  $\square$

**LEMMA 17.** *If the graph  $\mathcal{P}_1$  obtained by deleting  $a \rightarrow b$  from a completed PDAG  $\mathcal{C}$  can be extended to a new completed PDAG,  $\mathcal{C}_1$ , then we have that for any directed edge  $x \rightarrow y$  in  $\mathcal{C}$ , if  $y$  is not  $b$  or a descendant of  $b$ , then  $x \rightarrow y$  occurs in  $\mathcal{C}_1$ .*

**PROOF.** Because  $x \rightarrow y$  occurs in  $\mathcal{C}$ , so it is strongly protected in  $\mathcal{C}$ . If  $x \rightarrow y$  does not occur in  $\mathcal{C}_1$ , it is not strongly protected in  $\mathcal{C}_1$  from Lemma 2. From the definition of strongly protected, we know that the four cases in Figure 1 in which  $v \rightarrow u$  is strongly protected do not involve any descendant of  $u$ . Thus, if  $x \rightarrow y$  is not compelled in  $\mathcal{C}_1$ , there must exist a directed edge  $w \rightarrow z$  between two nondescendants of  $y$  such that the edges between nondescendants of  $z$  are strongly protected, and  $w - z$  is no longer strongly protected in  $\mathcal{P}_1$ . Because  $\mathcal{P}_1$  is obtained by deleting  $a \rightarrow b$ ,  $z$  is nondescendant of  $b$ , we have that  $w \rightarrow z$  is strongly protected in  $\mathcal{P}_1$ , yielding a contraction.  $\square$

We now give a proof of Lemma 9:

**PROOF OF LEMMA 9.** Since  $\mathcal{C} \in \mathcal{S}_p^n$ , we have  $n_{\mathcal{C}_1} < n$ . That is, the condition  $\text{id}_1$  in Definition 9 holds for  $\text{InsertD } x \rightarrow y$  of  $\mathcal{C}_1$ .

For any undirected edge  $w - y$  in  $\mathcal{C}$ ,  $x$  must be parent of  $w$ ; otherwise the edge between  $y$  and  $w$  is directed. Then deleting  $x \rightarrow y$  from  $\mathcal{C}$  will result in  $w \rightarrow y$

in  $\mathcal{C}_1$ . Thus, we have that all  $N_y$  in  $\mathcal{C}$  become parents of  $y$  in  $\mathcal{C}_1$ . From the condition **dd**<sub>2</sub>, the parents of  $y$  but not  $x$  in  $\mathcal{C}$  are also parents of  $y$  in  $\mathcal{C}_1$ . If there is a partially directed path from  $y$  to  $x$  in  $\mathcal{C}_1$ , then the vertex adjacent to  $y$  in this path must be a child of  $y$  or a vertex that is parent of  $y$  and  $x$  in  $\mathcal{C}$ . We will show that if the vertex is not a parent of  $y$  and  $x$  in  $\mathcal{C}$ , there exists a contradiction.

If there is a partially directed path from  $y$  to  $x$  in  $\mathcal{C}_1$ , we can find a shortest partially directed path like  $y - u_1 - \dots - u_k \rightarrow \dots \rightarrow x$  from Lemma 16, denoted as  $L_1$ . Any directed edge, say  $u_i \rightarrow u_{i+1}$ , in  $L_1$  does not become  $u_i \leftarrow u_{i+1}$  in  $\mathcal{C}$ . If  $L_1$  does not include undirected edges in  $\mathcal{C}_1$ , we have that the vertices of  $L_1$  form a partially directed cycle in  $\mathcal{C}$ . We just need to show that the vertices of the undirected path  $L_1$  also form a partially directed path in  $\mathcal{C}$ .

Suppose  $y \rightarrow u_1$  occurs in  $\mathcal{C}$ . If  $u_1 - u_2$  is undirected in  $\mathcal{C}$ , then  $y \rightarrow u_2$  must occur in  $\mathcal{C}$ , and consequently,  $L_1$  will not be shortest in  $\mathcal{C}_1$ . If  $u_2 \rightarrow u_1$  occurs in  $\mathcal{C}$ , there exists a  $v$ -structure  $u_2 \rightarrow u_1 \leftarrow y$  in  $\mathcal{C}_1$ ; otherwise  $u_2$  and  $y$  are adjacent, and  $L_1$  is not the shortest path in  $\mathcal{C}_1$ . Thus,  $u_1 \rightarrow u_2$  must occur in  $\mathcal{C}$ . In this manner, we get that all edges in  $y - u_1 - \dots - u_k \rightarrow \dots \rightarrow x$  are directed in  $\mathcal{C}$  and are directed from  $u_i \rightarrow u_{i+1}$ . This implies that there exists a partially directed cycle in  $\mathcal{C}$ . So,  $u_1$  must be a parent of  $y$  and  $x$  in  $\mathcal{C}$ . We have  $u_1 \in \Omega_{xy}$  and every partially directed path of  $\mathcal{C}_1$  from  $y$  to  $x$  contains at least one vertex in  $\Omega_{xy}$ .

Since all vertices in  $\Omega_{xy}$  in  $\mathcal{C}_1$  are parents of  $x$  and  $y$  in  $\mathcal{C}$ , if there are two vertices, say  $w_1, w_2 \in \Omega_{xy}$ , that are not adjacent, the subgraph  $w_1 \rightarrow y \leftarrow w_2$  could be a  $v$ -structure in  $\mathcal{C}_1$ . So, all vertices in  $\Omega_{xy}$  in  $\mathcal{C}_1$  are adjacent and  $\Omega_{xy}$  is a clique.

We have that the parents of  $y$  in  $\mathcal{C}_1$  ( $(\Pi_y)_{\mathcal{C}_1}$ ) are in the union of the parents and neighbors of  $y$  in  $\mathcal{C}$  ( $(\Pi_y \cup N_y)_{\mathcal{C}_1}$ ). If there is at least one neighbor  $u$  of  $y$  in  $\mathcal{C}$ ,  $u$  must be child of  $x$  in  $\mathcal{C}$  and parent of  $y$  in  $\mathcal{C}_1$ , so parents of  $x$  and  $y$  are not the same. If there is no neighbor of  $y$  in  $\mathcal{C}$ , the parents of  $y$  in  $\mathcal{C}_1$  are the same as in  $\mathcal{C}$ , except those vertices that are parents of  $x$ , that is,  $(\Pi_y - \Pi_x)_{\mathcal{C}_1} = (\Pi_y - \Pi_x)_{\mathcal{C}}$ . At the same time, from Lemma 17, the parents of  $x$  in  $\mathcal{C}_1$  are also the parents of  $x$  in  $\mathcal{C}$ . Thus, the parents of  $x$  and  $y$  are not the same in  $\mathcal{C}_1$ . From Lemma 3, we have that  $\text{InsertD } x \rightarrow y$  is valid for  $\mathcal{C}_1$ , and condition **id**<sub>2</sub> holds.

Denote the modified PDAG of operator  $\text{InsertD } x \rightarrow y$  of  $\mathcal{C}_1$  as  $\mathcal{P}'$ . We need to show that the corresponding completed PDAG of  $\mathcal{P}'$  is  $\mathcal{C}$ . Equivalently, we just need to show that  $\mathcal{P}'$  and  $\mathcal{C}$  have the same skeleton and  $v$ -structures. Clearly,  $\mathcal{P}'$  and  $\mathcal{C}$  have the same skeleton. A  $v$ -structure that is in  $\mathcal{C}$  but not in  $\mathcal{C}_1$  must have the form  $x \rightarrow y \leftarrow u$ , where  $u$  is parent of  $y$  but not adjacent to  $x$ . From condition **dd**<sub>2</sub> in Definition 9,  $u \rightarrow y$  also occurs in  $\mathcal{C}_1$ , so such a  $v$ -structure must also exist in  $\mathcal{P}'$ . This implies that all  $v$ -structures of  $\mathcal{C}$  are also in  $\mathcal{P}'$ . Moreover, the  $v$ -structures in  $\mathcal{C}_1$  but not in  $\mathcal{C}$  must have the form  $x \rightarrow v \leftarrow y$ , where  $v$  is a common child of  $y$  and  $x$  in  $\mathcal{C}_1$ . Clearly, after we insert  $x \rightarrow y$  to  $\mathcal{C}_1$ , this is no longer a  $v$ -structure in  $\mathcal{P}'$  implying that all  $v$ -structures of  $\mathcal{P}'$  are in  $\mathcal{C}$ . Thus,  $\mathcal{P}'$  and  $\mathcal{C}$  have the same  $v$ -structures.

Let the modified graph of DeleteD  $x \rightarrow y$  from  $\mathcal{C}$  be  $\mathcal{P}$ ; we know that  $\mathcal{P}$  and  $\mathcal{C}_1$  have the same  $v$ -structures. Thus, for any  $u$  that is a common child of  $x$  and  $y$  in  $\mathcal{C}_1$ ,  $x \rightarrow u \leftarrow y$  is a  $v$ -structure in  $\mathcal{P}$ . This implies that  $y \rightarrow u$  occurs in  $\mathcal{C}$  and the condition **id**<sub>3</sub> hold.  $\square$

**PROOF OF LEMMA 10.** Since  $x, z$  and  $y$  are in the same chain component of  $\mathcal{C}$ , they have the same parent set in  $\mathcal{C}$ . The modified graph of  $o'$  has the same skeleton and  $v$ -structures as  $\mathcal{C}_1$  because all compelled edges in  $\mathcal{C}$  remain compelled in  $\mathcal{C}_1$ . We just need to prove that the operator  $o'$  is valid and equivalently to prove that the conditions **rm**<sub>1</sub>, **rm**<sub>2</sub> and **rm**<sub>3</sub> hold for  $\mathcal{C}_1$ .

We now show that the condition **rm**<sub>1</sub>,  $x$  and  $y$  have the same parents in  $\mathcal{C}_1$  holds. Because  $x$  and  $y$  have the same parents in  $\mathcal{C}$ , and all directed edges in  $\mathcal{C}$  occur in  $\mathcal{C}_1$ , we just need to consider the neighbors of  $x$  or  $y$ . Let  $w - y$  be any undirected edge in  $\mathcal{C}$ , we consider the edges between  $w$  and  $x$  or  $z$ :

- (1) If both  $w - z$  and  $x - w$  occur in  $\mathcal{C}$ ,  $w - y$  and  $w - x$  must be undirected in  $\mathcal{C}_1$ .
- (2) If  $w - z$  occurs but  $x - w$  does not occur in  $\mathcal{C}$ ,  $z \rightarrow w$  and  $y \rightarrow w$  must be in  $\mathcal{C}_1$ .
- (3) If  $x - w$  occurs but  $w - z$  does not occur in  $\mathcal{C}$ , there is an undirected cycle of length 4 without a chord in  $\mathcal{C}$ . Thus, this case will not occur.
- (4) If neither  $w - z$  nor  $x - w$  occur in  $\mathcal{C}$ , and there is no undirected path other than  $w - y - z$  from  $w$  to  $z$  in  $\mathcal{C}$ , then  $w - y$  occurs in  $\mathcal{C}_1$ . If there exists another undirected path from  $w$  to  $z$ , there must exist an undirected path of length 2 like  $w - u' - z$  in  $\mathcal{C}$ , and  $y$  is adjacent to  $u'$ . In this case,  $y - w$  occurs in  $\mathcal{C}_1$  when  $x - u'$  occurs and  $y \rightarrow w$  occurs when  $x$ , and  $u'$  are not adjacent.

Thus, there are no neighbors of  $y$  in  $\mathcal{C}$  that become parents of  $y$  in  $\mathcal{C}_1$ ; that is,  $y$  has the same parents in both  $\mathcal{C}_1$  and  $\mathcal{C}$ . Similarly,  $x$  has the same parents in both  $\mathcal{C}_1$  and  $\mathcal{C}$ . we get  $x$  and  $y$  have the same parents in  $\mathcal{C}_1$ , and the condition **rm**<sub>1</sub> holds.

All parents of  $x$  must also be parents of  $z$  in  $\mathcal{C}_1$  since they are in the same chain component. For any  $w \in N_{xy}$ ,  $w - z$  also occurs in  $\mathcal{C}$ ; otherwise  $x - z - y - w - x$  would form cycle of length 4 without a chord. We have  $w \rightarrow z$  must be in  $\mathcal{C}_1$ , otherwise a new  $v$ -structure will occur in  $\mathcal{C}_1$ . Thus, we have  $\Pi(x) \cup N_{xy} \subset \Pi(z)$  in  $\mathcal{C}_1$ .

For any  $w \in \Pi(z)$  in  $\mathcal{C}_1$ , if  $w \in \Pi(z)$  in  $\mathcal{C}$ , it must also be parent of  $x, y$  and  $z$  in  $\mathcal{C}_1$ , so  $w \in \Pi(x)$  in  $\mathcal{C}_1$ . If  $w - z$  is an undirected edge in  $\mathcal{C}$ , there exist undirected edges  $w - x$  and  $w - y$  in  $\mathcal{C}$  such that  $w \rightarrow z$  is in  $\mathcal{C}_1$ . Thus,  $w \in N_{xy}$  in  $\mathcal{C}_1$ . We have that  $w \in \Pi(x) \cup N_{xy}$  and  $\Pi(z) \subset \Pi(x) \cup N_{xy}$  in  $\mathcal{C}_1$ . Thus,  $\Pi(z) = \Pi(x) \cup N_{xy}$  in  $\mathcal{C}_1$ , and the condition **rm**<sub>2</sub> holds.

Any undirected path between  $x$  and  $y$  in  $\mathcal{C}_1$  will also be an undirected path in  $\mathcal{C}$ , so these paths contain at least one vertex in  $N_{xy}$  in  $\mathcal{C}$ . From the proof above, any vertex in  $N_{xy}$  in  $\mathcal{C}$  is also a vertex of  $N_{xy}$  in  $\mathcal{C}_1$ . Thus any undirected path between  $x$  and  $y$  contains a vertex in  $N_{xy}$  in  $\mathcal{C}_1$ , and the condition **rm**<sub>3</sub> holds.  $\square$

**PROOF OF LEMMA 11.** From Lemma 5 and the condition  $\mathbf{rm}_3$ , there exists a consistent extension of  $\mathcal{C}$ , denoted by  $\mathcal{D}$ , such that all neighbors of  $x$  in  $\mathcal{C}$  are children of  $x$  in  $\mathcal{D}$ , and all neighbors of  $y$  in  $\mathcal{C}$  are parents of  $x$  in  $\mathcal{D}$ . Changing  $y \rightarrow z$  to  $z \rightarrow y$  in  $\mathcal{D}$ , we obtain a new graph  $\mathcal{D}'$ . From the proof of Lemma 4, we can get that (1)  $\mathcal{D}'$  is a DAG, (2)  $\mathcal{D}'$  is a consistent extension of  $\mathcal{C}_1$ . Thus,  $\mathcal{D}$  is a consistent extension of the PDAG that results from making the  $v$ -structure  $x \rightarrow z \leftarrow y$  in  $\mathcal{C}_1$ . Thus, we can get  $\mathcal{C}$  by applying  $\text{MakeV } x \rightarrow z \leftarrow y$  to  $\mathcal{C}_1$ . This implies that  $\text{MakeV } x \rightarrow z \leftarrow y$  is a valid operator of  $\mathcal{O}_1$  and satisfies the condition  $\text{mv}_1$ .  $\square$

**PROOF OF THEOREM 5.** In order to prove this theorem, we first introduce three results: Lemmas 18, 19 and 20.  $\square$

**LEMMA 18.** *For any completed PDAG  $\mathcal{C}$  containing at least one undirected edge, there exists an undirected edge  $x - y$  for which  $N_{xy}$  is a clique.*

**LEMMA 19.** *For any completed PDAG  $\mathcal{C}$ , if  $x \rightarrow y$  occurs in  $\mathcal{C}$ , then  $\Pi_x \neq \Pi_y \setminus x$ .*

A proof of Lemmas 18 and 19 can be found in Chickering [6].

**LEMMA 20.** *For any completed PDAG  $\mathcal{C}$  containing no undirected edges and at least one directed edge, there exists at least one vertex  $x$  for which any parent of  $x$  has no parent.*

**PROOF.** The following procedure will find the vertex whose parent has no parent. Let  $a \rightarrow b$  be a directed edge in  $\mathcal{C}$ , set  $y = a$  and  $x = b$ .

(1) If  $\Pi_y$  is not empty, choose any vertex  $u$  in  $\Pi_y$ , set  $x = y$  and  $y = u$ . Repeat this step until we find a directed edge  $y \rightarrow x$  for which  $\Pi_y$  is empty.

(2) Since  $\Pi_y$  is empty, from Lemma 19, there exists at least one vertex other than  $y$  in  $\Pi_x$ . If there is a vertex  $u \in \Pi_x$  and  $u \neq y$  such that  $\Pi_u$  is not empty, choose a vertex in  $\Pi_u$ , denoted as  $v$  and set  $y = v$  and  $x = u$ , and go to step 1.

Since  $\mathcal{C}$  is an acyclic graph with finite vertices, above procedure must end at the step in which the parents of  $x$  have no parents.  $\square$

We now show a proof of Theorem 5.

**PROOF OF THEOREM 5.** We need to show that for any two completed PDAGs  $\mathcal{C}_1, \mathcal{C}_2 \in \mathcal{S}$ , there exists a sequence of operators in  $\mathcal{O}$  such that  $\mathcal{C}_2$  can be obtained by applying a sequence of operators to PDAGs, starting from  $\mathcal{C}_1$ . Because  $\mathcal{O}$  is reversible, any operator in  $\mathcal{O}$  has a reversible operator, so we just need to show

that any completed PDAG can be transferred to empty graph without edges. The procedure includes three basic steps.

(1) Deleting all undirected edges.

From Lemma 18, for any completed PDAG containing at least one undirected edge, we can find an operator with type of DeleteU that satisfies the condition  $\mathbf{du}_1$  in Definition 9. We can delete an undirected edge with this operator and get a new completed PDAG whose skeleton is a subgraph of the skeleton of the initial completed PDAG. Repeating this procedure, we can get a completed PDAG, denoted as  $\mathcal{C}_i$ , which contains no undirected edges.

(2) Deleting some directed edges.

From Lemma 20, we can find a vertex, denoted as  $x$ , whose parents have no parents in the completed PDAG  $\mathcal{C}_i$ . If  $\Pi_x$  contains more than two vertices, we can choose a vertex  $u \in \Pi_x$ . Because (1)  $N_x$  is empty in  $\mathcal{C}_i$ , and (2) any other directed edge  $v \rightarrow x$  forms a  $v$ -structure in  $\mathcal{C}_i$ , we have that  $v \rightarrow x$  is also compelled in the completed PDAG obtained by deleting directed edge  $u \rightarrow x$  from  $\mathcal{C}_i$ . We can delete  $v \rightarrow x$  from  $\mathcal{C}_i$  and get a new completed PDAG whose skeleton is a subgraph of the skeleton of the initial one. Thus, the new completed PDAG is in  $\mathcal{S}$ . Repeat this procedure for all other directed edges  $v' \rightarrow x$  in which  $v' \in \Pi_x$  until there are only two vertices in  $\Pi_x$  in the new completed PDAG, denoted as  $\mathcal{C}_j$ .

(3) Removing a  $v$ -structure.

The conditions  $\text{rm}_1$ ,  $\text{rm}_2$  and  $\text{rm}_3$  hold for the  $v$ -structure  $y \rightarrow x \leftarrow u$  in  $\mathcal{C}_j$ , so, we can remove  $y \rightarrow x \leftarrow u$  from  $\mathcal{C}_j$  and get a new completed PDAG whose skeleton is a subgraph of the skeleton of the initial graph. Denote the resulting completed PDAG as  $\mathcal{C}_k$ ; it may still contain some undirected edges.

By repeatedly applying the above the steps in sequence, we can finally obtain a graph without any edges.  $\square$

**Acknowledgments.** This work was partly done when Yangbo He was visiting Department of Statistics in UC Berkeley. Yangbo He would like to thank Prof. Lan Wu for her support of this visit. Jinzhu Jia's work was done when he was a postdoc in UC Berkeley. We are very grateful to Adam Bloniarz for his comments that significantly improved the presentation of our manuscript. We also thank Jasjeet Sekhon, the co-Editor, the Associate Editor and the reviewer for their helpful comments and suggestions.

## SUPPLEMENTARY MATERIAL

**Supplement to “Reversible MCMC on Markov equivalence classes of sparse directed acyclic graphs”** (DOI: 10.1214/13-AOS1125SUPP;.pdf). In this supplementary note, we give some algorithms, examples, an experiment and the proofs of the results in this paper.

## REFERENCES

- [1] ALDOUS, D. and FILL, J. Reversible Markov chains and random walks on graphs. Available at <http://www.stat.berkeley.edu/~aldous/RWG/book.html>.

- [2] ANDERSSON, S. A., MADIGAN, D. and PERLMAN, M. D. (1997). A characterization of Markov equivalence classes for acyclic digraphs. *Ann. Statist.* **25** 505–541. [MR1439312](#)
- [3] CASTELO, R. and PERLMAN, M. D. (2004). Learning essential graph Markov models from data. In *Advances in Bayesian Networks. Studies in Fuzziness and Soft Computing* **146** 255–269. Springer, Berlin. [MR2090887](#)
- [4] CHICKERING, D., GEIGER, D. and HECKERMAN, D. (1995). Learning Bayesian networks: Search methods and experimental results. In *Proceedings of Fifth Conference on Artificial Intelligence and Statistics* 112–128. Ft. Lauderdale, Society for Artificial Intelligence in Statistics, FL.
- [5] CHICKERING, D. M. (1995). A transformational characterization of equivalent Bayesian network structures. In *Proceedings of the Eleventh Conference on Uncertainty in Artificial Intelligence (Montreal, PQ, 1995)* 87–98. Morgan Kaufmann, San Francisco, CA. [MR1615012](#)
- [6] CHICKERING, D. M. (2002). Learning equivalence classes of Bayesian-network structures. *J. Mach. Learn. Res.* **2** 445–498. [MR1929415](#)
- [7] CHICKERING, D. M. (2003). Optimal structure identification with greedy search: Computational learning theory. *J. Mach. Learn. Res.* **3** 507–554. [MR1991085](#)
- [8] COOPER, G. F. and YOO, C. (1999). Causal discovery from a mixture of experimental and observational data. In *Proceedings of the Fifteenth Conference on Uncertainty in Artificial Intelligence* 116–125. Morgan Kaufmann, San Francisco, CA.
- [9] DASH, D. and DRUZDZEL, M. J. (1999). A hybrid anytime algorithm for the construction of causal models from sparse data. In *Proceedings of the Fifteenth Conference on Uncertainty in Artificial Intelligence* 142–149. Morgan Kaufmann, San Mateo, CA.
- [10] DOR, D. and TARSI, M. (1992). A simple algorithm to construct a consistent extension of a partially oriented graph. Technical Report R-185, Cognitive Systems Laboratory, UCLA.
- [11] EBERHARDT, F. and SCHEINES, R. (2007). Interventions and causal inference. *Philos. Sci.* **74** 981–995. [MR2444979](#)
- [12] FINEGOLD, M. and DRTON, M. (2011). Robust graphical modeling of gene networks using classical and alternative  $t$ -distributions. *Ann. Appl. Stat.* **5** 1057–1080. [MR2840186](#)
- [13] FRIEDMAN, N. (2004). Inferring cellular networks using probabilistic graphical models. *Science Signaling* **303** 799.
- [14] GILLISPIE, S. B. (2006). Formulas for counting acyclic digraph Markov equivalence classes. *J. Statist. Plann. Inference* **136** 1410–1432. [MR2253771](#)
- [15] GILLISPIE, S. B. and PERLMAN, M. D. (2002). The size distribution for Markov equivalence classes of acyclic digraph models. *Artificial Intelligence* **141** 137–155. [MR1935281](#)
- [16] HE, Y., JIA, J. and YU, B. (2013). Supplement to “Reversible MCMC on Markov equivalence classes of sparse directed acyclic graphs.” DOI:[10.1214/13-AOS1125SUPP](#).
- [17] HE, Y.-B. and GENG, Z. (2008). Active learning of causal networks with intervention experiments and optimal designs. *J. Mach. Learn. Res.* **9** 2523–2547. [MR2460892](#)
- [18] HECKERMAN, D., GEIGER, D. and CHICKERING, D. M. (1995). Learning Bayesian networks: The combination of knowledge and statistical data. *Machine Learning* **20** 197–243.
- [19] HECKERMAN, D., MEEK, C. and COOPER, G. (1999). A Bayesian approach to causal discovery. In *Computation, Causation, and Discovery* 141–165. AAAI Press, Menlo Park, CA. [MR1689964](#)
- [20] JANSEN, R., YU, H., GREENBAUM, D., KLUGER, Y., KROGAN, N. J., CHUNG, S., EMILI, A., SNYDER, M., GREENBLATT, J. F. and GERSTEIN, M. (2003). A Bayesian networks approach for predicting protein–protein interactions from genomic data. *Science* **302** 449.
- [21] KALISCH, M. and BUHLMANN, P. (2007). Estimating high-dimensional directed acyclic graphs with the pc-algorithm. *J. Mach. Learn. Res.* **8** 613–636.

- [22] LAURITZEN, S. L. and RICHARDSON, T. S. (2002). Chain graph models and their causal interpretations. *J. R. Stat. Soc. Ser. B Stat. Methodol.* **64** 321–361. [MR1924296](#)
- [23] LOVASZ, L. (1993). Random walks on graphs: A survey. *Combinatorics: Paul Erdős Is Eighty* **2** 1–46.
- [24] MAATHUIS, M. H., KALISCH, M. and BÜHLMANN, P. (2009). Estimating high-dimensional intervention effects from observational data. *Ann. Statist.* **37** 3133–3164. [MR2549555](#)
- [25] MADIGAN, D., ANDERSSON, S. A., PERLMAN, M. D. and VOLINSKY, C. T. (1996). Bayesian model averaging and model selection for Markov equivalence classes of acyclic digraphs. *Comm. Statist. Theory Methods* **25** 2493–2519.
- [26] MEEK, C. (1995). Causal inference and causal explanation with background knowledge. In *Proceedings of the Eleventh Conference on Uncertainty in Artificial Intelligence (San Mateo)* 403–410. Morgan Kaufmann, San Francisco, CA.
- [27] MUNTEANU, P. and BENDOU, M. (2001). The eq framework for learning equivalence classes of Bayesian networks. In *Proceedings IEEE International Conference on Data Mining, 2001. ICDM 2001* 417–424. IEEE, San Jose, CA.
- [28] NORRIS, J. R. (1997). *Markov Chains. Cambridge Series in Statistical and Probabilistic Mathematics* **2**. Cambridge Univ. Press, Cambridge.
- [29] PEÑA, J. M. (2007). Approximate counting of graphical models via MCMC. In *Proceedings of the 11th International Conference on Artificial Intelligence* 352–359. San Juan, Puerto Rico; available at <http://jmlr.org/proceedings/papers/v2/pena07a/pena07a.pdf>.
- [30] PEÑA, J. M. (2013). Approximate counting of graphical models via mcmc revisited. Preprint. Available at [arXiv:1301.7189](https://arxiv.org/abs/1301.7189).
- [31] PEARL, J. (1988). *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann, San Mateo, CA. [MR0965765](#)
- [32] PEARL, J. (2000). *Causality: Models, Reasoning, and Inference*. Cambridge Univ. Press, Cambridge. [MR1744773](#)
- [33] PEARL, J. and VERMA, T. S. (1991). A theory of inferred causation. In *Principles of Knowledge Representation and Reasoning (Cambridge, MA, 1991)* 441–452. Morgan Kaufmann, San Mateo, CA. [MR1142173](#)
- [34] PERLMAN, M. D. (2001). Graphical model search via essential graphs. In *Algebraic Methods in Statistics and Probability (Notre Dame, IN, 2000). Contemporary Mathematics* **287** 255–265. Amer. Math. Soc., Providence, RI. [MR1873680](#)
- [35] SPIRITES, P., GLYMOUR, C. N. and SCHEINES, R. (2001). *Causation, Prediction, and Search*. MIT Press, Cambridge.
- [36] VERMA, T. and PEARL, J. (1990). Equivalence and synthesis of causal models. In *Proceedings of the Sixth Annual Conference on Uncertainty in Artificial Intelligence* 270. Elsevier, Amsterdam.
- [37] VERMA, T. and PEARL, J. (1992). An algorithm for deciding if a set of observed independencies has a causal explanation. In *Proceedings of the Eighth International Conference on Uncertainty in Artificial Intelligence* 323–330. Morgan Kaufmann, San Mateo, CA.

Y. HE  
J. JIA  
SCHOOL OF MATHEMATICAL SCIENCES  
AND CENTER OF STATISTICAL SCIENCES  
PEKING UNIVERSITY  
BEIJING 100871  
CHINA  
E-MAIL: [heyb@math.pku.edu.cn](mailto:heyb@math.pku.edu.cn)  
[jzjia@math.pku.edu.cn](mailto:jzjia@math.pku.edu.cn)

B. YU  
DEPARTMENT OF STATISTICS  
UNIVERSITY OF CALIFORNIA  
BERKELEY, CALIFORNIA 94720  
USA  
E-MAIL: [binyu@stat.berkeley.edu](mailto:binyu@stat.berkeley.edu)