Differential Privacy Preserving Causal Graph Discovery

Depeng Xu*, Shuhan Yuan†, Xintao Wu*
* *University of Arkansas, Fayetteville, AR, USA Email: {depengxu, xintaowu}@uark.edu*
† *Tongji University, Shanghai, China Email: 4e66@tongji.edu.cn*

***Abstract*—Discovering causal relationships by constructing the causal graph provides critical information to researchers and decision makers. Yet releasing causal graphs may risk leakage of individual participant's privacy. It is very under-exploited how to enforce differential privacy in causal graph discovery. In this work, we focus on the PC algorithm, a classic constraint-based causal graph discovery algorithm, and propose a differentially private PC algorithm (PrivPC) for categorical data. PrivPC adopts the exponential mechanism and significantly reduces the number of edge elimination decisions. Therefore, it incurs much less privacy budget than the naive approaches that add privacy protection at each conditional independence test. For numerical data, we further develop a differentially private causal discovery algorithm (PrivPC*). The idea is to add noise once onto the covariance matrix from which partial correlations used for conditional independence test can be derived. Experimental results show that PrivPC and PrivPC* achieve good utility and robustness for different settings of causal graphs. To our best knowledge, this is the first work on how to enforce differential privacy in constraint-based causal graph discovery.**

***Keywords*-differential privacy; causal graph; PC algorithm**

I. Introduction

Understanding the cause-effect relationships helps people make decisions in a rational manner. Traditionally, experimental intervention is conducted to identify causal effect. However, it is often difficult, if not infeasible, to set up intervention experiments for high-dimensional datasets. Hence causal graph modeling has been widely used to discover causal relationships from observed data [1], [2]. Causal graphs, represented as directed acyclic graphs, can conveniently capture the causal structure underlying data. However, it is computationally difficult to derive an optimal causal graph as the number of possible causal graphs increases super-exponentially to the number of nodes.

Constraint-based causal graph discovery methods [3], [4] have been developed to build causal graphs. Among them, the PC algorithm [3] is a classic constraint-based algorithm for causal graph discovery. It first forms a complete and undirected graph and then recursively deletes edges according to conditional independence test. The PC algorithm is computationally feasible and consistent even for sparse causal graphs. It achieves causal graph faithfulness (i.e., the data can be assumed to be simulated from a probability distribution that factorizes according to a causal graph). However, when causal graph discovery is performed on sensitive individual data, it raises individual participants' concerns about their privacy, e.g., in the fields of medical or financial analysis. One solution to protect individual privacy against leakage risk is differential privacy.

Privacy preserving causal modeling is severely under exploited. To the best of our knowledge, there is only one work [5] related to differentially private causal inference. In [5], the authors studied how to achieve differential privacy when deciding on the causal direction between two variables X and Y. They presented how to use Laplace mechanism to privatize dependence scores such as rank correlation coefficient, Kendall rank score, and variance score used in the additive noise framework [6]. However, the work assumed there is no confounding variable Z that commonly causes or is a common effect of X and Y. Our private causal graph analysis is based on the classic causal inference framework [3], [4] that determines the causal relationship between any pair of variables that are a part of a large causal network via conditional independence testing.

In this paper, we focus on developing differential privacy preserving causal graph discovery. One of the challenges is that the number of conditional independence tests used in the PC algorithm is tremendously large. Simply speaking, one conditional independence test is conducted for each edge conditional on any subset of variables in the original PC algorithm. The complexity of the PC algorithm increases exponentially with the dimension and sparseness of a causal graph. The naive idea of adding privacy protection at each conditional independence test incurs a very large privacy budget.

Our idea is to reduce the number of conditional independence tests (or edge elimination decisions) thus incurring less privacy consumption. We determine the conditional independence set of a given node at a particular condition order in one single step. To enforce differential privacy, we develop a procedure to determine the conditional independence set in a differential privacy preserving manner. Our contributions in this work are as follows.

- We develop a differentially private causal graph discovery algorithm (PrivPC) for categorical data. Our PrivPC uses the conditional independence set for edge elimination. We develop a differentially private version of the conditional independence set selection procedure based on the exponential mechanism [7]. To our best knowledge, this is the first work on how to enforce differential privacy in constraint-based causal graph

978-1-5386-1027-5/17 $31.00 © 2017 IEEE
DOI 10.1109/PAC.2017.24

discovery.

- For numerical data, we develop a differentially private PC algorithm (PrivPC*). The idea is to add Laplace noise once to the covariance matrix from which partial correlations used for conditional independence test can be derived.
- We evaluate the proposed algorithms on simulated datasets in both numerical and categorical cases and a real dataset. Experimental results show that PrivPC and PrivPC* achieve good utility and robustness under different settings of causal graphs.

II. Related work

A. Causal graph discovery

A causal graph is a probabilistic graphical model which is widely used for causality representation, reasoning and inference [4]. The constraint-based approaches use data to make categorical decisions about whether particular conditional-independence constraints hold and then pieces these decisions together by looking for those sets of causal structures that are consistent with the constraints. In [3], Spirtes et al. proposed a constraint based method (PC) which consists of two phases, graph skeleton learning and edge orientation. The graph skeleton learning starts from the complete undirected graph, then thins that graph by removing edges conditional independence relations in various orders. The edge orientation is based on the concept of d-separation.

The original PC algorithm [3] is known to be order-dependent, in the sense that the output may depend on the input order of the variables. There are several modified PC algorithms for causal graph discovery. Colombo and Maathuis [8] proposed a simple modification, called "PC-stable", which yields order-independent adjacencies in the skeleton. Ramsey et al. [9] proposed a conservative PC algorithm which has a slight variation in computing all potential v-structures of the PC algorithm. To improve the efficiency, researchers have developed local causal discovery algorithms [10]–[12]. These methods first determine the local structure for a set of chosen attributes and then integrate those local structures in the global causal structure.

For numerical data, Kalisch and Buhlmann [13] showed the use of PC algorithm for estimating the skeleton and equivalence class of a high-dimensional directed acyclic graph with corresponding Gaussian distribution. The method uses partial correlations derived from independent Gaussian observations that are faithful to a suitably sparse causal graph. It was later proved that consistency of Gaussian model can be carried over to a boarder class of Gaussian copula or nonparanormal models which uses rank-based correlation instead of Pearson correlation [14]. In [15], the authors developed an enhanced constraint-based approach for causal structure learning in microarray expression data. The approach uses the independence graph from graphical Gaussian modeling as the input of causal structure learning and uses graphical decomposition techniques to further improve the performance.

Table I: Notations

Symbol	Definition
$Adj(\mathcal{G}, V_i)$	adjacent set of V_i
$Ind(\mathcal{G}, V_i)$	conditional independence set of V_i
$T(ij\|\mathbf{k})$	conditional independence test for V_i, V_j given $\mathbf{V_k}$
$p_{ij\|\mathbf{k}}$	p-value returned by $T(ij\|\mathbf{k})$
$d(ij\|\mathbf{k})$	distance score of $T(ij\|\mathbf{k})$
π, π_t	permutation, the t-th largest element in π
$\|\bullet\|$	size of $\bullet$
$\hat{\bullet}$	differentially private $\bullet$
$\mathcal{M}^{\epsilon}_{q,S(q)}(D,R)$	an exponential mechanism with a privacy budget ε, a quality function q and a sensitivity $S(q)$ on an input dataset D and a output domain R

B. Differential privacy

Differential privacy research has been significantly studied from the theoretical perspective, e.g., [16], [17]. Meanwhile, there have been extensive studies on the applicability of enforcing differential privacy in some particular analysis tasks, e.g., data collection [18], [19], data streams [20], stochastic gradient descents [21], recommendation [22], regression [16], online learning [23], publishing contingency tables [24], spectral graph analysis [25], Bayesian networks [26], and most recently deep learning [27]–[30]. The mechanisms of achieving differential privacy mainly include the classic approach of adding Laplacian noise [31], the exponential mechanism [7], and the functional perturbation approach [16].

Dwork et al. [31] proposed the Laplace mechanism to preserve differential privacy by calibrating the standard deviation of the noise according to the sensitivity of the query function. In addition to the Laplace mechanism, McSherry and Talwar [7] developed the exponential mechanism to guarantee differential privacy in non-numeric sensitive queries by sampling according to a mapping function instead of adding noise. The exponential mechanism has been widely explored in several private learning algorithms, such as logistic regression, support vector machine, k-means clustering [32], [33], decision trees [34], and genetic association framework [35]. Johnson and Shmatikov [35] used the exponential mechanism to privately release the number of significant SNPs associated with a trait and to privately release the location of the significant SNPs.

III. Preliminaries

In this section, we present the technical preliminaries. We introduce causal graph, the classic PC algorithm, and differential privacy. Important notations are summarized in Table I.

A. Causal graph discovery

1) Causal graph: A causal graph is a probabilistic graphical model which is specified by a directed acyclic graph (DAG) $\mathcal{G} = (\mathbf{V}, \mathbf{E})$. $\mathbf{V} = \{V_1, \ldots, V_p\}$ is a set of nodes, each of which corresponds to a variable. $\mathbf{E} \subseteq \mathbf{V} \times \mathbf{V}$ is a set of edges, which is a subset of ordered pairs of distinct nodes. We use $V_i \rightarrow V_j$ to denote a directed edge. $Adj(\mathcal{G}, V_i)$ denotes the adjacency set of node V_i on graph $\mathcal{G}$, also called neighbors of V_i.

We use P to denote a probability distribution that is faithful with respect to the graph $\mathcal{G}$. Consider the set of random variables $\mathbf{V} \sim P$, faithfulness of P with respect to $\mathcal{G}$ means: if for any $V_i, V_j \in \mathbf{V}(i \neq j)$ and any set $\mathbf{V_k} \subseteq \mathbf{V}$, V_i and V_j are conditionally independent given $\mathbf{V_k}$, nodes V_i and V_j are d-separated by $\mathbf{V_k}$ [3].

Multiple causal graphs can fit the same distribution. They are called Markov equivalent. Verma and Pearl [36] demonstrated that two causal graphs are equivalent if and only if they have the same skeleton and the same v-structures. The skeleton of a causal graph $\mathcal{G}$ has only undirected edges. The undirected edge between node V_i and V_j is denoted as $V_i - V_j$. A v-structure is an ordered set of nodes (V_i, V_j, V_k) such that $\mathcal{G}$ contains directed edges $V_i \rightarrow V_j$ and $V_k \rightarrow V_j$, and V_i and V_k are not adjacent in $\mathcal{G}$. All causal graphs that are Markov equivalent belong to the same Markov equivalence class. A Markov equivalence class is uniquely represented by a completed, partially directed acyclic graph (CPDAG) [37]. The main goal of estimating a causal graph is to identify the CPDAG, which contains the conditional independence information of node sets.

2) PC algorithm: The PC algorithm is a constraint-based method for causal graph discovery [3]. It assumes that the probability distribution P is Markovian and faithful with respect to the correct graph $\mathcal{G}$, which means all conditional independences in the joint distribution are entailed by the Markov condition. The PC algorithm first estimates the skeleton of the underlying CPDAG, and then determines the orientation of as many edges as possible.

a) Estimate the skeleton given data: The first phase of the PC algorithm is shown in Algorithm 1 (Lines 1−15). Sprites et al [3] proved that this PC algorithm can construct the true skeleton $\mathcal{C}$ of the causal graph given a causal graph $\mathcal{G}$ and a distribution P that is faithful to $\mathcal{G}$. The algorithm starts from the complete undirected graph, then thins that graph by removing edges with zero order conditional independence relations, thins again with first order conditional independence relations, and so on. The set of variables conditioned on need only to be a subset of the set of variables adjacent to the variable conditioned.

In Line 8 of Algorithm 1), we determine whether V_i and V_j are conditionally independent given set $\mathbf{V_k}$ where the number of variables in $\mathbf{V_k}$ is the condition order ord. All the information used in conditional independence test can be extracted from contingency table. Specifically, we use $T(ij|\mathbf{k})$ to denote the conditional independence test of V_i and V_j given set $\mathbf{V_k}$ and use $p_{ij|\mathbf{k}}$ to denote its p-value. For $T(ij|\mathbf{k})$, each cell of the contingency table contains the frequency of corresponding value combination of $V_i, V_j, \mathbf{V_k}$ (denoted as $n_{ij|\mathbf{k}}$). The number of cells in the contingency table is $|V_i| \times |V_j| \times \prod_{h \in \mathbf{k}} |V_h|$ where $|V_i|$ denotes the domain size of variable V_i. We often use two statistics, χ^2 and G^2, for conditional independence test.

$$\chi^2 = \sum \frac{(n_{ij|\mathbf{k}} - \hat{n}_{ij|\mathbf{k}})^2}{\hat{n}_{ij|\mathbf{k}}},$$

$$G^2 = 2 \sum (n_{ij|\mathbf{k}}) ln(\frac{n_{ij|\mathbf{k}}}{\hat{n}_{ij|\mathbf{k}}}),$$

where $\hat{n}_{ij|\mathbf{k}}$ is the estimated number of records (based on independence assumption) of the corresponding cell in the contingency table. The degree of freedom (df) in the test is $(|V_i| - 1) \times (|V_j| - 1) \times \prod_{h \in \mathbf{k}} |V_h|$. If the p-value is less than threshold τ, we remove the edge $V_i - V_j$ due to its conditional independence.

b) Orient the edges given the skeleton structure: The second phase of the PC algorithm is shown in Algorithm 1 (Lines 17−29), which extends skeleton to a CPDAG belonging to the equivalence class of the underlying true causal graph [4].

The complexity of the PC algorithm is bounded by the maximal reached value of ord. Recall that ord is the condition order of a conditional independence test, which is the number of vertices in conditioned set. Let l be the maximal degree of order and let p be the number of vertices. Then in the worst case the number of conditional independence tests needed is bounded by $2\binom{p}{2} \sum_{ord=0}^{l} \binom{p-1}{ord}$, which has a maximal value of $\frac{p^2(p-1)^{l-1}}{(l-1)!}$.

Note that the PC algorithm only accesses input data when conducting conditional independence tests in the first phase. In the second phase, it only uses the structural information to orient edges. Thus we only need to preserve privacy in the first phase. However, the number of conditional independence tests makes it very challenging to achieve differential privacy in the PC algorithm.

B. Differential Privacy

Differential privacy guarantees the output of a query to be insensitive to any particular record's presence or value in the dataset. Let D be a dataset that contains n records with p attributes. We use D' to denote a neighboring dataset which differs in exactly one record from D ($|D - D'| = 1$).

Definition 1: (Differential privacy [31]) A mechanism $\mathcal{M}$ satisfies ε-differential privacy, if for all neighboring datasets D and D' and all subsets Z of $\mathcal{M}$'s range:

$$P(\mathcal{M}(D) \in Z) \leq \exp(\varepsilon) \cdot P(\mathcal{M}(D') \in Z). \quad (1)$$

Algorithm 1: PC algorithm

Input : Vertex set $\mathbf{V}$, Dataset D, Significance threshold τ
Output: Estimated CPDAG $\mathcal{G}$

Form the complete undirected graph $\mathcal{C}_0$ on vertex set $\mathbf{V}$
$ord = 0$; $\mathcal{C} = \mathcal{C}_0$
repeat
  repeat
    Select an edge $V_i - V_j$ that are adjacent in $\mathcal{C}$ s.t. $|Adj(\mathcal{C}, V_i) \setminus \{V_j\}| \geq ord$
    repeat
      Choose $\mathbf{V_k} \subseteq Adj(\mathcal{C}, V_i) \setminus \{V_j\}$ with $|\mathbf{V_k}| = ord$
      if $p_{ij|\mathbf{k}} = T(ij|\mathbf{k}) < \tau$ **then**
        Delete $V_i - V_j$ from $\mathcal{C}$
        break
      end
    until *all* $\mathbf{V_k}$ *have been tested*
  until *all* $V_i - V_j$ *with* $|\mathbf{V_k}| = ord$ *have been tested*
  $ord = ord + 1$
until *for all* V_i $|Adj(\mathcal{C}, V_i) \setminus \{V_j\}| < ord$

Create a partially directed graph with minimal pattern (Identify unshielded colliders)
foreach *pair of non adjacent variables* V_i, V_j *with a common neighbor* V_k **do**
  if V_i, V_j *are not conditionally independent given* V_k **then**
    Orient $V_i - V_k - V_j$ as $V_i \rightarrow V_k \leftarrow V_j$
  end
end
Convert to complete pattern: use rules by Pearl [4]
repeat
  Rule 1: $V_i \rightarrow V_j - V_k$ goes to $V_i \rightarrow V_j \rightarrow V_k$ (No new collider is introduced)
  Rule 2: $V_i \rightarrow V_k \rightarrow V_j$ with $V_i - V_j$, then $V_i - V_j$ goes to $V_i \rightarrow V_j$ (Avoid cycle)
  Rule 3: $V_i - V_j$, $V_i - V_k$, $V_i - V_l$, $V_k \rightarrow V_j$, $V_l \rightarrow V_j$ but V_k and V_l are not connected; then $V_i - V_j$ goes to $V_i \rightarrow V_j$
until *no more edges can be oriented*
return CPDAG $\mathcal{G}$

Definition 2: (Global sensitivity [31]) Given a function f, the sensitivity $S_f(D)$ is defined as

$$S_f(D) = \max_{D,D'} ||f(D) - f(D')||_1. \quad (2)$$

Definition 3: (Laplace mechanism [31]) Given a dataset D and a query f, a mechanism $\mathcal{M}(D) = f(D) + \boldsymbol{\eta}$ satisfies ε-differential privacy, where $\boldsymbol{\eta}$ is a random vector drawn from $Lap(S_f(D)/\varepsilon)$.

Laplace mechanism is a popular method to achieve differential privacy. It adds identical independent noise into each output value of $f(D)$. We use $S_f(D)$ to denote sensitivity of $f(D)$. It measures the maximum possible change in $f(D)$ when one record in the dataset changes.

Differential privacy has two useful properties: (a) Composability: let $\mathcal{M}_1$ be an ε_1-differentially private mechanism, and let $\mathcal{M}_2$ be an ε_2-differentially private mechanism. Then their combination is $\varepsilon_1 + \varepsilon_2$-differentially private. (b) Security under post-processing: any transformation or query over a differentially private result $g(\mathcal{M}(D))$ is still differentially private, as long as it does not acquire additional access to input dataset D.

IV. Differentially Private Causal Graph Discovery Algorithm for categorical data

This section describes our differentially private PC algorithm (PrivPC) for private causal graph discovery. Our algorithm is based on the exponential mechanism.

A. Exponential mechanism

The exponential mechanism is a differentially private method which selects one outcome from a set of potential outcomes based on some probability distribution. For a given dataset D and a privacy budget ε, the quality function induces a probability distribution over the output domain, from which the outcomes are exponentially chosen. It favors higher scoring classes, while guaranteeing ε-differential privacy.

Definition 4: (L_1-sensitivity of quality function [7]) Let $q : D \rightarrow R$ be a quality function that scores each output class $r \in R$. The sensitivity of this function is defined as

$$S(q(D, r)) = \max_{D,D',r \in R} ||q(D', r) - q(D, r)||_1. \quad (3)$$

Definition 5: (Exponential mechanism [7]) Given a dataset D, the exponential mechanism $\mathcal{M}$ randomly selects a potential outcome r based on the following probability, then the mechanism $\mathcal{M}^{\epsilon}_{q,S(q)}(D, R)$ is ε-differentially private:

$$P(r \in R \text{ is selected}) \propto \exp\left(\frac{\varepsilon q(D, r)}{2S(q)}\right). \quad (4)$$

Simply speaking, assume the goal is to output a class $r \in R$ from its known output domain R. We use a quality function $q(D, r)$ that measures the quality of a potential output r, given a dataset D. The exponential mechanism $\mathcal{M}^{\epsilon}_{q,S(q)}(D, R)$ outputs $r \in R$ with probability proportional to $\exp\left(\frac{\varepsilon q(D, r)}{2S(q)}\right)$ and ensures ε-differential privacy. The mechanism assigns the highest probability to the best answer, and the probability assigned to any other drops off exponentially with the decline of its quality function.

B. Differentially Private PC (PrivPC) Algorithm

The PC algorithm accesses input data to estimate the skeleton $\mathcal{C}$ of the causal graph in the first phase. More specifically, input data is queried at each conditional independence test (Line 8 in Algorithm 1). The naive method, which attempts to achieve differential privacy at each conditional independence test, incurs a tremendous amount of the privacy budget ε due to the huge number of conditional independence tests.

The motivation for PrivPC is to reduce the number of conditional independence tests (or edge elimination decisions) thus incurring less privacy consumption. One conditional independence test is conducted for each edge $V_i - V_j$ given each set $V_{\mathbf{k}}$ with condition order ord in the original PC algorithm. Our idea is to determine the conditional independence set $Ind(\mathcal{C}, V_i)$ of node V_i with order ord in one single step. Any node V_j in the determined conditional independence set $Ind(\mathcal{C}, V_i)$ is conditionally independent with V_i. Hence we can simply remove the edge $V_i - V_j$. To enforce differential privacy, we develop a procedure, *findPrivInd*, to determine the conditional independence set in a differential privacy preserving manner. Our idea is based on the exponential mechanism.

Algorithm 2 shows the pseudo code of our PrivPC. Our key procedure *findPrivInd* is shown in Line 6. It takes node V_i,$Adj(\mathcal{C}, V_i)$,ord, dataset D, and two privacy parameters ϵ_1 and ϵ_2 as input and returns $\widehat{Ind}(\mathcal{C}, V_i)$ that contains the differentially private conditional independence set of node V_i. We will discuss in the following subsections how to use the exponential mechanism to enforce differential privacy in *findPrivInd*. Note that the set $\widehat{Ind}(\mathcal{C}, V_i)$ is conditionally independent to node V_i given some $\mathbf{V_k}$ with $|\mathbf{V_k}| = ord$. Once we get the conditional independence set $\widehat{Ind}(\mathcal{C}, V_i)$, we delete $V_i - V_j$ for each node $V_j \in \widehat{Ind}(\mathcal{C}, V_i)$, as shown in Lines 7-8 of Algorithm 2.

We emphasize that PrivPC makes the decision on edge elimination at each round of determining the conditional independence set for each node instead of making the decision by applying conditional independence test for each edge. As a result, it expects to incur less privacy budget. The potential utility loss is because the differentially private $\widehat{Ind}(\mathcal{C}, V_i)$ may not be the same as the true $Ind(\mathcal{C}, V_i)$. Our PrivPC algorithm finishes the skeleton construction if $|Adj(\mathcal{C}, V_i)| < ord + 1$ for all V_i or all privacy budget ε is used up. It then continues the second phase by running the Algorithm 1 Lines 17-29 to orient edges. Note that if the early termination occurs in the first phase, only the high-order conditional independence tests are compromised. The early termination introduces acceptable errors, because the high-order conditional independence is rare in many cases.

Algorithm 2: PrivPC algorithm

Input : Vertex set $\mathbf{V}$, Dataset D, Privacy budget ε
Output: Estimated CPDAG $\hat{\mathcal{G}}$

1 Form the complete undirected graph $\mathcal{C}_0$ on vertex set $\mathbf{V}$
2 $ord = 0$; $\mathcal{C} = \mathcal{C}_0$
3 **repeat**
4 **repeat**
5 Select a node V_i s.t. $|Adj(\mathcal{C}, V_i)| \geq ord + 1$
6 Find the private conditional independence set, $\widehat{Ind}(\mathcal{C}, V_i) = findPrivInd\Big(V_i, Adj(\mathcal{C}, V_i), ord, D, \epsilon_1, \epsilon_2\Big)$
7 **for** *all* $V_j \in \widehat{Ind}(\mathcal{C}, V_i)$ **do**
8 Delete $V_i - V_j$ from $\mathcal{C}$
9 **end**
10 **until** *all nodes* V_i *with* $|Adj(\mathcal{C}, V_i)| \geq ord + 1$ *have been selected*
11 $ord = ord + 1$
12 **until** $|Adj(\mathcal{C}, V_i)| < ord + 1$ *for all* V_i*, or all privacy budget* ε *is used up*
13 Run Algorithm 1 Lines 17-29 to orient edges

C. Differentially private conditional independence set selection procedure: Details

To achieve differential privacy in the conditional independence set selection, we use the exponential mechanism to privately select nodes from the adjacent set $Adj(\mathcal{C}, V_i)$ into the private conditional independence set $\widehat{Ind}(\mathcal{C}, V_i)$.

Algorithm 3 shows the pseudo code of *findPrivInd*. The high level idea works as follows. For each node V_j from $Adj(\mathcal{C}, V_i)$, we find the minimum p-value $p_{ij|\mathbf{k}}$ from all the conditional independence tests $T(ij|\mathbf{k})$ (Line 5) and assign a quality function value $q_1(D, V_j)$ for each node V_j (Line 6) based on our designed distance function (Line 5). Note that the smaller $p_{ij|\mathbf{k}}$ is, the higher quality value $q_1(D, V_j)$ is assigned. The exponential mechanism has a higher probability to select node V_j into the conditional independence set $\widehat{Ind}(\mathcal{C}, V_i)$. Hence it is more likely that the edge $V_i - V_j$ is removed from the causal graph. In Lines 12-16, we apply the exponential mechanism $\mathcal{M}_{q_1,1}^{\epsilon_1/\hat{\beta}}(D, R_1)$ to select all conditional independence nodes. In each selection, the output domain is all the nodes in $Adj(\mathcal{C}, V_i)$. But those nodes that have larger q_1 values (smaller p-values) will be more likely chosen. However, the number of conditional independence nodes (i.e., the iteration number of the node selection β) may also incur privacy leakage. Thus, we also need to privately determine the number of conditional independence nodes. Lines 8-10 show how to get the differentially private version of the number of conditional independence node ($\hat{\beta}$). The domain of $\hat{\beta}$ is from 0 to $|Adj(\mathcal{C}, V_i)|$. We use the exponential mechanism $\mathcal{M}_{q_2,1}^{\epsilon_2}(D, R_2)$ to determine $\hat{\beta}$ according to the quality function q_2. Thus, all decisions

made in the conditional independence set selection procedure ensure differential privacy.

Algorithm 3: Privately find the conditional independence set of V_i

1 **Procedure** `findPrivInd()`
  Input : V_i, $Adj(\mathcal{C}, V_i)$,ord, Dataset D, ϵ_1, ϵ_2
  Output: $\widehat{Ind}(\mathcal{C}, V_i)$
2 $R_1 = Adj(\mathcal{C}, V_i)$
3 **for** *each* $V_j \in Adj(\mathcal{C}, V_i)$ **do**
4 Find $\mathbf{V_k} \subseteq Adj(\mathcal{C}, V_i) \setminus \{V_j\}$ with $|\mathbf{V_k}| = ord$ such that $p_{ij|\mathbf{k}} = T(ij|\mathbf{k})$ is the minimum
5 Calculate $d(ij|\mathbf{k})$ by Equation 5
6 $q_1(D, V_j) = d(ij|\mathbf{k})$
7 **end**
8 $R_2 = \{0, 1, \ldots, |Adj(\mathcal{C}, V_i)|\}$
9 Calculate q_2 for $0, 1, \ldots, |Adj(\mathcal{C}, V_i)|$ by Equations 6,7,8
10 $\hat{\beta} = \mathcal{M}^{\epsilon_2}_{q_2,1}(D, R_2)$
11 $\widehat{Ind}(\mathcal{C}, V_i) = \emptyset$
12 **for** $t = 1, \ldots, \hat{\beta}$ **do**
13 $V_r = \mathcal{M}^{\epsilon_1/\hat{\beta}}_{q_1,1}(D, R_1)$
14 $\widehat{Ind}(\mathcal{C}, V_i) = \widehat{Ind}(\mathcal{C}, V_i) \bigcup \{V_r\}$
15 $q_1(D, V_r) = -\infty$
16 **end**
17 **return** $\widehat{Ind}(\mathcal{C}, V_i)$

Distance score: We need to define a quality function $q : D \to R$ that assigns a value to each potential output. However, the p-value of the conditional independence test has high sensitivity, which makes it infeasible to produce accurate yet private result via the exponential mechanism. Our idea is to derive a distance score that can approximate the change of p-value. This idea was first proposed in [35] to determine most significant SNPs from genome wide association studies.

$$d(ij|\mathbf{k}) = \begin{cases} \min\{d = |D - D'|\} \Rightarrow p'_{ij|\mathbf{k}} \geq \tau \\ \quad \text{if} p_{ij|\mathbf{k}} \leq \tau, \\ -\min\{d = |D - D'|\} \Rightarrow p'_{ij|\mathbf{k}} < \tau \\ \quad \text{otherwise,} \end{cases} \tag{5}$$

where D' denotes a d-distance neighboring dataset that has d records different from D and $p'_{ij|\mathbf{k}}$ denotes the p-value of $T(ij|\mathbf{k})$ over D'. The value of $d(ij|\mathbf{k})$ shows the minimum number of records in D that must be changed s.t. the p-value is changed from above to below the significance threshold τ or vice versa. The sign of $d(ij|\mathbf{k})$ indicates the significance of the conditional independence test for edge $V_i - V_j$. The value of $d(ij|\mathbf{k})$ has a good approximation to p-value in terms of the conditional independence information, but it has a small sensitivity. Hence, we can define the quality functions based on this distance score. For example, a conditional independence test $T(ij|\mathbf{k})$ on dataset D returns a below-threshold p-value (a significant conditional independence); $d(ij|\mathbf{k})$ finds the minimum d-distance neighboring dataset D' s.t. $T(ij|\mathbf{k})$ on D' returns an above-threshold p-value, and gives it a positive sign.

Privately select a node into the conditional independence set: We use $\mathcal{M}^{\epsilon_1/\hat{\beta}}_{q_1,1}(D, R_1)$ (Line 13 in Algorithm 3) to select a node into $\widehat{Ind}(\mathcal{C}, V_i)$. The output domain R_1 is all nodes in $Adj(\mathcal{C}, V_i)$; the quality function q_1 for each node $V_j \in Adj(\mathcal{C}, V_i)$ is the distance score $d(ij|\mathbf{k})$ corresponding to its minimum $T(ij|\mathbf{k})$. The node V_j with a higher distance score has a higher probability to be selected by the exponential mechanism. Once node V_r is returned from the exponential mechanism, we include it in the conditional independence set $\widehat{Ind}(\mathcal{C}, V_i)$ (Line 14) and set its quality function value as $-\infty$ (Line 15) so that V_r will not be selected in the following iterations.

Corollary 1: The L_1-sensitivity $S(q_1(D, r))$ of quality function q_1 is 1. The private node selection achieves ϵ_1-differential privacy by applying $\mathcal{M}^{\epsilon_1/\hat{\beta}}_{q_1,1}(D, R_1)$.

Proof: Distance score is based on the number of records. As adding or removing one record can only change a counting query result at most by 1, the distance score (Equation 5) can only be changed at most by 1. In the selection of one node, the sensitivity of q_1 is 1. Then we iterate the private node selection for $\hat{\beta}$ times from the remaining nodes in R_1. In each round of the conditional independence set selection, $\mathcal{M}^{\epsilon_1/\hat{\beta}}_{q_1,1}(D, R_1)$ uses ϵ_1 out of the total privacy budget ε. Each iteration uses $\epsilon_1/\hat{\beta}$ to select one node into the conditional independence set.

Privately determine the number of conditional independence nodes: We use the exponential mechanism $\mathcal{M}^{\epsilon}_{q_2,1}(D, R_2)$ to privately return $\hat{\beta}$ (Line 10 in Algorithm 3). Let π be a permutation that sorts all choices of $V_j \in Adj(\mathcal{C}, V_i)$ by decreasing distances $d(ij|\mathbf{k})$, where π_t denotes the t-th largest element in π. Thus, $d(\pi_t)$ finds the t-th largest $d(ij|\mathbf{k})$. The output domain R_2 of $\hat{\beta}$ is $\{0, 1, \ldots, \alpha\}$ ($\alpha = |Adj(\mathcal{C}, V_i)|$). We assign a score of the quality function q_2 to each potential output as follows:
For $0 < t < \alpha$,

$$q_2(D, t) = \begin{cases} \min\Big(d(\pi_t), -d(\pi_{t+1})\Big) - 1, \\ \text{if } \Big(\text{p-value}(\pi_t) < \tau\Big) \wedge \Big(\text{p-value}(\pi_{t+1}) \geq \tau\Big); \\ -d(\pi_{t+1}, \\ \text{if } \Big(\text{p-value}(\pi_t) < \tau\Big) \wedge \Big(\text{p-value}(\pi_{t+1}) < \tau\Big); \\ d(\pi_t), \text{ otherwise.} \end{cases} \tag{6}$$

and

$$q_2(D, 0) = \begin{cases} -d(\pi_1) - 1, \text{ if p-value}(\pi_1) \geq \tau; \\ -d(\pi_1), \text{ otherwise.} \end{cases} \tag{7}$$

$$q_2(D, \alpha) = \begin{cases} d(\pi_\alpha) - 1, \text{ if p-value}(\pi_\alpha) < \tau; \\ d(\pi_\alpha), \text{ otherwise.} \end{cases} \tag{8}$$

The quality function q_2 is based on the sorted distance scores. The exponential mechanism has a higher probability to select the number in the permutation at which the distance score flips from positive to negative. Thus, it has the higher probability to return the correct number of the conditional independence nodes.

Corollary 2: The L_1-sensitivity $S(q_2(D,r))$ of quality function q_2 is 1. The private selection of the iteration number achieves ϵ_2-differential privacy by applying $\mathcal{M}^{\epsilon}_{q_2,1}(D,R_2)$.

Proof: Similar to Corollary 1, one record can only change the distance score (Equation 5) by 1. Because q_2 provides a desired lower bound on the distance score (Equation 5), q_2 changes at most 1 when the distance is changed by 1. Thus, the sensitivity of q_2 is at most 1. In each round of the conditional independence set selection, $\mathcal{M}^{\epsilon_2}_{q_2,1}(D,R_2)$ uses ϵ_2 out of total privacy budget ε.

Theorem 1: The PrivPC algorithm achieves ε-differential privacy by recursively applying private conditional independence set selection procedure.

Proof: The proof follows by a combination of Corollary 1 and Corollary 2. PrivPC only queries D in the private conditional independence set selection procedure. One round of the private conditional independence set selection achieves $(\epsilon_1+\epsilon_2)$-differential privacy. PrivPC repeats this procedure until all of total budget ε is used up.

Let l be the maximal degree of order and let p be the number of vertices. The number of rounds of conditional independence set selections is bounded by $l \times p$, which is much less than the bound of the number of conditional independence tests $\frac{p^2(p-1)^{l-1}}{(l-1)!}$. Hence, PrivPC requires a much smaller privacy budget than the naive method for the same level of performance.

D. Comparison with a baseline method (LapMech)

An improved method (LapMech) on the naive method is to add noise once on the input dataset D. A basic contingency table contains adequate information of D for all possible conditional independence tests. We add Laplace noise $Lap(0,1/\varepsilon)$ on the basic contingency table. Any following conditional independence tests only access the protected basic contingency table instead of the original input dataset D. For each conditional independence test, the corresponding contingency table can be directly summarized from the protected basic contingency table. This algorithm only accesses the input data once. The basic contingency table has a sensitivity of only 1, since one data point can change the count at most by 1. We use the LapMech method as a baseline against our PrivPC algorithm for evaluation.

However, the basic contingency table is computationally infeasible for high-dimensional data or multiple-category variables. It suffers from over-fitting, as a large portion of value combinations may not be observed in the dataset D. The size of basic contingency table is $\prod_{i=1}^{p} Cat_i$, which is larger than n in most cases. Moreover, summarizing the count numbers $N_{ij|\mathbf{k}}$ from the basic contingency table to the corresponding contingency table is inefficient, and it accumulates noises when summarizing to small tables.

Applying the exponential mechanism on conditional independence set selection has more advantages over the baseline method (LapMech). Because it uses contingency table, it does not encounter the problems of oversize, overfitting or accumulated noise. At each round of conditional independence set selection, it is independent on the dimension p or the number of categories. We expect our PrivPC algorithm has better performance than the LapMech mechanism, especially in high-dimensional, complex cases.

V. Differentially Private Causal Graph Discovery algorithm for Numerical Data

In this section, we propose a differentially private PC (PrivPC*) algorithm for numerical data.

When all nodes are random variables with a multivariate normal distribution, conditional independence tests can be inferred from partial correlations [2]. Kalisch and Buhlmann [13] proposed a modified PC algorithm using partial correlation, which has an uniform consistency for Gaussian samples.

PrivPC* achieves the differential privacy based on the following procedure. Given a dataset D, we first calculate the covariance matrix Σ. Suppose Σ has a lower triangular structure (as Σ is symmetric), with variance σ_i^2 on diagonal and covariance $Cov(V_i,V_j)$ off diagonal. We then add Laplace noise onto the covariance matrix Σ to ensure differential privacy. The noise is randomly drawn from $Lap(0,S_f(D)/\varepsilon)$. Based on the private covariance matrix $\hat{\Sigma}$, the output of any following conditionally independence test (Line 8 in Algorithm 1) between V_i and V_j given $\mathbf{V_k}$ is also differentially private, as it only queries the private covariance matrix $\hat{\Sigma}$ instead of the input dataset D. Thus, differential privacy is ensured in any transformation or query over the private covariance matrix $\hat{\Sigma}$.

Using the private covariance matrix $\hat{\Sigma}$ in PrivPC*: After we get the private covariance matrix $\hat{\Sigma}$, we convert the private $\hat{\Sigma}$ to the private correlation matrix $\hat{\mathcal{P}}$, where $\hat{\rho}_{ij} = \frac{\widehat{Cov}(V_i,V_j)}{\hat{\sigma}_i\hat{\sigma}_j}$ denotes the correlation between V_i and V_j. To calculate the partial correlation $\hat{\rho}_{ij|\mathbf{k}}$ between V_i and V_j given $\mathbf{V_k}$, we use the sub-correlation matrix with entries from corresponding rows and columns and find the respective entries $\hat{\rho}^{-1}_{ij}, \hat{\rho}^{-1}_{ii}, \hat{\rho}^{-1}_{jj}$ from the inverse of the sub-matrix. The partial correlation $\hat{\rho}_{ij|\mathbf{k}}$ is calculated by

$$\hat{\rho}_{ij|\mathbf{k}} = -\frac{\hat{\rho}^{-1}_{ij}}{\sqrt{\hat{\rho}^{-1}_{ii}\hat{\rho}^{-1}_{jj}}}.$$

Then, we apply Fisher's z-transform to get private test

statistics $\hat{Z}(ij|\mathbf{k})$:

$$\hat{Z}(ij|\mathbf{k}) = \frac{1}{2}\log\left(\frac{1+\hat{\rho}_{ij|\mathbf{k}}}{1-\hat{\rho}_{ij|\mathbf{k}}}\right)$$

Thus, the decision boundary on the conditional independence in Line 8 of Algorithm 1 becomes:
"If $\sqrt{n-|\mathbf{V_k}|-3}|\hat{Z}(ij|\mathbf{k})| \leq \Phi^{-1}(1-\tau/2)$, then V_i and V_j are conditionally independent on $\mathbf{V_k}$",
where n is the sample size; Φ is the CDF of $N(0,1)$; τ is the significance level.

Theorem 2: The PrivPC* algorithm achieves ε-differential privacy by applying the Laplace mechanism on the covariance matrix Σ. The global sensitivity $S_f(D)$ of the covariance matrix Σ is $\frac{p(p+1)}{2(n-1)}$.

Proof: We use $\mathbf{x}_u$ to denote the u-th record in dataset D. We use x_{ui} to denote the value of variable V_i in the u-th record ($i = 1, 2, \ldots, p$). Suppose it follows standard normal distribution $x_{ui} \sim N(0,1)$.

$$\begin{aligned}
&S_f = ||\Sigma(D) - \Sigma(D')||_1 \\
&= \sum_{i=1}^{p}\left(\frac{\sum_{u=1}^{n} x_{ui}^2}{n-1} - \frac{\sum_{u=1}^{n-1} x_{ui}^2}{n-2}\right) + \\
&\sum_{i=1}^{p}\sum_{j=1}^{p}\left(\frac{\sum_{u=1}^{n} x_{ui}x_{uj}}{n-1} - \frac{\sum_{u=1}^{n-1} x_{ui}x_{uj}}{n-2}\right) \\
&\leq \sum_{i=1}^{p}\left(\frac{\sum_{u=1}^{n} x_{ui}^2}{n-1} - \frac{\sum_{u=1}^{n-1} x_{ui}^2}{n-1}\right) + \\
&\sum_{i=1}^{p}\sum_{j=1}^{p}\left(\frac{\sum_{u=1}^{n} x_{ui}x_{uj}}{n-1} - \frac{\sum_{u=1}^{n-1} x_{ui}x_{uj}}{n-1}\right) \\
&= \sum_{i=1}^{p}\left(\frac{x_{ni}^2}{n-1}\right) + \sum_{i=1}^{p}\sum_{j=1}^{p}\left(\frac{x_{ni}x_{nj}}{n-1}\right) \\
&= \frac{2\sum_{i=1}^{p} x_{ni}^2 + 2\sum_{i=1}^{p}\sum_{j=1}^{p} x_{ni}x_{nj}}{2(n-1)} = \frac{\sum_{i=1}^{p} x_{ni}^2 + \left(\sum_{i=1}^{p} x_{ni}\right)^2}{2(n-1)} \\
&\leq \frac{p + \left(p \times \sqrt{\frac{p}{p}}\right)^2}{2(n-1)} = \frac{p(p+1)}{2(n-1)}
\end{aligned}$$

where $i \neq j$, $\sum_{i=1}^{p} x_{ni}^2 = p$ because $\sigma^2 = \frac{\sum_{i=1}^{p} x_{ni}^2}{p} = 1$.

The global sensitivity S_f of $\Sigma(D)$ is $\frac{p(p+1)}{2(n-1)}$. When n is large, the added Laplace noise would be relatively small. □

Compared to a basic contingency table, the correlation matrix has a much smaller size ($p \times p$). Hence, conditional independence test via the private covariance matrix $\hat{\Sigma}$ does not introduce extra noise to test statistics, whereas the basic contingency table wastes privacy budget on under-represented entries in its cells. Hence, it is plausible to add noise on the covariance matrix for numerical data.

VI. Experiments

We conduct empirical evaluations using both synthetic datasets and the UCI Adult dataset [38]. We implement our private PC algorithms (PrivPC and PrivPC*) based on the R-package *pcalg* [39]. Each of private and non-private PC algorithms outputs a CPDAG for a given dataset. We evaluate their performance by comparing the CPDAGs with the ground-truth causal graph based on three metrics:

$$\text{True Positive Rate (TPR)} = \frac{\text{correctly found edges}}{\text{true edges}}$$

$$\text{False Positive Rate (FPR)} = \frac{\text{misplaced edges}}{\text{true gaps (no edge)}}$$

$$\text{True Discovery Rate (TDR)} = \frac{\text{correctly found edges}}{\text{found edges}}$$

For TPR and TDR, higher values indicate better performance. For FPR, lower values indicate better performance. We use G^2 on corresponding contingency tables for the conditional independence tests in our experiments. We use the same significance level $\tau = 0.01$ for conditional independence tests in all our experiments.

A. Categorical data

We evaluate the performance of our PrivPC against LapMech and the non-private PC algorithm under different parameter settings. Table II shows the parameter values, with the default values in bold. The parameter setting for causal graphs includes two parameters: the number of nodes p and the sparseness s $\left(2|\mathbf{E}|/p(p-1)\right)$. For each parameter setting (p, s), we use the $TETRAD$ [40] to generate one causal graph $\mathcal{G}$ that is considered as the ground-truth. Each node is assigned as a categorical variable with the number of domain values ranging from 2 to 5. Conditional probability tables corresponding to the structure of $\mathcal{G}$ are constructed. We then randomly generate a dataset D with n records according to the structure and conditional probability tables of $\mathcal{G}$. We conduct comparison based on the three metrics. For each graph setting, we repeat five times. In all experiments, ϵ_1 (privacy budget for selecting nodes into the conditional independence set) is simply set to be equal to ϵ_2 (privacy budget for determining the number of conditional independence nodes) in the *findPrivInd* procedure of our PrivPC. Specifically,$\epsilon_1 = \epsilon_2 = \frac{1}{30}\epsilon$ It is worth pointing out that the accuracy of our PrivPC could be improved by tuning the values of ϵ_1 and ϵ_2. For example, the selection of ϵ_1 and ϵ_2 can be heuristically made based on the potential iteration

Table II: Experimental parameters and values for categorical data

Parameter	Values
Privacy Budget ε	1, 3, **5**, 10
Sample Size n	1k, 3k, **5k**, 10k
Number of Nodes p	5, **10**, 15
Sparseness s	**0.2**, 0.4, 0.6

Table III: The experimental results for different privacy budgets ε on categorical data ($n = 10000, p = 10, s = 0.2$)

ε		Non-private PC	LapMech	PrivPC
10	TPR	1.000±0.000	0.862±0.060	**0.896±0.136**
	FPR	0.006±0.012	0.011±0.016	**0.006±0.012**
	TDR	0.980±0.045	0.960±0.055	**0.980±0.044**
5	TPR	1.000±0.000	0.742±0.117	**0.873±0.125**
	FPR	0.006±0.012	0.012±0.017	**0.006±0.013**
	TDR	0.980±0.045	0.960±0.055	**0.980±0.045**
3	TPR	1.000±0.000	0.748±0.108	**0.841±0.121**
	FPR	0.006±0.012	0.018±0.017	**0.017±0.025**
	TDR	0.980±0.045	**0.940±0.055**	0.940±0.089
1	TPR	1.000±0.000	0.613±0.157	**0.817±0.206**
	FPR	0.006±0.012	0.105±0.024	**0.073±0.035**
	TDR	0.980±0.045	0.640±0.089	**0.740±0.134**

Table IV: The experimental results for different numbers of nodes p on categorical data ($\varepsilon = 5, n = 5000, s = 0.2$)

p		Non-private PC	LapMech	PrivPC
5	TPR	1.000±0.000	**1.000±0.000**	**1.000±0.000**
	FPR	0.025±0.056	**0.025±0.056**	**0.025±0.056**
	TDR	0.933±0.149	**0.933±0.149**	**0.933±0.149**
10	TPR	1.000±0.000	0.738±0.158	**0.884±0.121**
	FPR	0.011±0.015	0.035±0.036	**0.023±0.013**
	TDR	0.960±0.055	0.880±0.130	**0.920±0.045**
15	TPR	0.990±0.022	0.455±0.174	**0.815±0.128**
	FPR	0.019±0.016	0.181±0.080	**0.053±0.021**
	TDR	0.926±0.061	0.227±0.093	**0.798±0.073**

number (β) which is related to the parameter setting of the graph (p and s).

Table III to Table VI show the experimental results when one parameter varies and the others are fixed with default values. We highlight the better values between LapMech and PrivPC.

Table III shows the performance of non-private PC, LapMech, and PrivPC algorithms for different privacy budgets ε. When $\varepsilon = 10$, both LapMech and PrivPC achieve high TPR and TDR. However, when a stronger privacy is enforced (smaller ε), LapMech incurs much larger utility loss than PrivPC. For example, when $\varepsilon = 1$, LapMech loses 25% on TPR and 30% on TDR. On the contrary, our PrivPC only loses 10% on TPR and 20% on TDR. This result demonstrates that our PrivPC is robust to different privacy budgets ε.

Table V: The experimental results for different sample sizes n on categorical data ($\varepsilon = 5, p = 10, s = 0.2$)

n		Non-private PC	LapMech	PrivPC
10k	TPR	1.000±0.000	0.742±0.117	**0.873±0.125**
	FPR	0.006±0.012	0.012±0.017	**0.006±0.013**
	TDR	0.980±0.045	0.960±0.055	**0.980±0.045**
5k	TPR	1.000±0.000	0.738±0.158	**0.884±0.121**
	FPR	0.011±0.015	0.035±0.036	**0.023±0.013**
	TDR	0.960±0.055	0.880±0.130	**0.920±0.045**
3k	TPR	0.982±0.041	0.812±0.129	**0.894±0.161**
	FPR	0.011±0.015	0.050±0.035	**0.045±0.038**
	TDR	0.960±0.055	0.820±0.130	**0.840±0.134**
1k	TPR	1.000±0.000	**0.836±0.157**	0.812±0.077
	FPR	0.049±0.021	0.140±0.033	**0.101±0.032**
	TDR	0.820±0.084	0.440±0.152	**0.620±0.130**

Table VI: The experimental results for different values of sparseness s on categorical data ($\varepsilon = 5, n = 5000, p = 10$)

s		Non-private PC	LapMech	PrivPC
0.2	TPR	1.000±0.000	0.738±0.158	**0.884±0.121**
	FPR	0.011±0.015	0.035±0.036	**0.023±0.013**
	TDR	0.960±0.055	0.880±0.130	**0.920±0.045**
0.4	TPR	1.000±0.000	**0.886±0.080**	0.774±0.062
	FPR	0.069±0.055	0.129±0.062	**0.084±0.041**
	TDR	0.866±0.100	0.741±0.121	**0.846±0.092**
0.6	TPR	1.000±0.000	**0.877±0.113**	0.809±0.073
	FPR	0.464±0.066	0.503±0.048	**0.280±0.081**
	TDR	0.328±0.165	0.266±0.160	**0.764±0.118**

We then evaluate the performance over causal graphs with different dimensions p. Table IV shows the comparison results for different p. When $p = 5$, both LapMech and PrivPC achieve good utility. However, PrivPC significantly outperforms LapMech with the increase of p. For example, LapMech has only 45% on TPR and 22% on TDR when $p = 15$. On the contrary, our PrivPC has about 80% on both FPR and TDR. Furthermore, LapMech is generally infeasible for high-dimensional datasets due to the huge size of contingency tables.

Table V shows the experimental results for different sample sizes n of input data. We observe that the sample size n has a strong impact on the performance of LapMech and PrivPC. When the sample size $n = 10k$, both LapMech and PrivPC have enough data to achieve good utility. When the sample size n drops, accuracy values of both LapMech and PrivPC drop significantly. However, PrivPC can still achieve much better TDRs than LapMech when the sample size n is small. For example, when $n = 1k$, PrivPC achieves 62% on TDR whereas LapMech achieves only 44%.

The maximal degree of order l of a causal graph $\mathcal{G}$ is related to the dimension p and sparseness s. The maximal

Table VII: Experimental parameters and values for numerical data

Parameter	Values
Privacy Budget ε	0.01, 0.1, **1**, 10
Sample Size n	100, 1000, **10000**
Number of Nodes p	10, 20, 50, 100
Sparseness s	0.1, **0.4**, 0.7

degree is roughly bounded by $p \times s$. We further conduct evaluation on causal graphs with various sparseness s. Table VI shows the comparison results with different sparseness s. For a dense causal graph ($s = 0.6$), LapMech has a high FPR and a low TDR as expected. On the contrary, PrivPC has a much lower FPR and a higher TDR. For a sparse causal graph ($s = 0.2$), PrivPC outperforms LapMech in terms of all three metrics.

We also compare the average execution time of LapMech and PrivPC. When $n = 5000, p = 10, s = 0.2$, LapMech has an average runtime of $111s$, whereas PrivPC only needs $9s$. As sparseness s or dimension p increases, the average runtime of LapMech becomes up to 1000 times worse than PrivPC. Hence, PrivPC requires a significantly less execution time than LapMech.

Overall, PrivPC achieves much better utility than LapMech. Meanwhile, PrivPC is also more robust to a small privacy budget ε, small sample size n, and high dimensional graphs. As discussed in Section IV, PrivPC makes a much smaller number of edge elimination decisions than the number of conditional independence tests in the original PC algorithm. Additionally, PrivPC runs much faster than LapMech.

B. Numerical data

To evaluate the performance of PrivPC*, we first generate a causal graph $\mathcal{G}$ with a given setting (p and s). We then generate the dataset D with n records according to the given causal graph $\mathcal{G}$ (with weights) with Gaussian distribution. We evaluate the performance of our PrivPC* algorithm against the non-private PC algorithm under different parameter settings (Table VII).

Table VIII to Table XI show the experimental results when one parameter varies and the others are fixed at default values. We highlight the values on which PrivPC* has *no significant difference* from the non-private PC.

Table VIII shows the performance for different privacy budgets ε on the same numerical datasets. With $\varepsilon = 1$ or higher, PrivPC* achieves very good performance on all three metrics (no significant difference from the non-private algorithms). With the smaller privacy budget $\varepsilon = 0.1$ or 0.01, the utility of PrivPC* decreases as the trade-off for strong privacy protection. However, PrivPC* still has reasonable utility when $\varepsilon = 0.1$.

Table VIII: The experimental results for different privacy budgets ε on numerical data ($n = 10000, p = 10, s = 0.4$)

ε		Non-private PC	PrivPC*
10	TPR	0.975±0.056	**0.975±0.056**
	FPR	0.094±0.096	**0.094±0.096**
	TDR	0.851±0.129	**0.851±0.129**
1	TPR	0.975±0.056	**0.975±0.056**
	FPR	0.094±0.096	**0.087±0.102**
	TDR	0.851±0.129	**0.865±0.143**
0.1	TPR	0.975±0.056	0.789±0.056
	FPR	0.094±0.096	0.187±0.093
	TDR	0.851±0.129	0.688±0.111
0.01	TPR	0.975±0.056	0.516±0.383
	FPR	0.094±0.096	0.373±0.062
	TDR	0.851±0.129	0.113±0.116

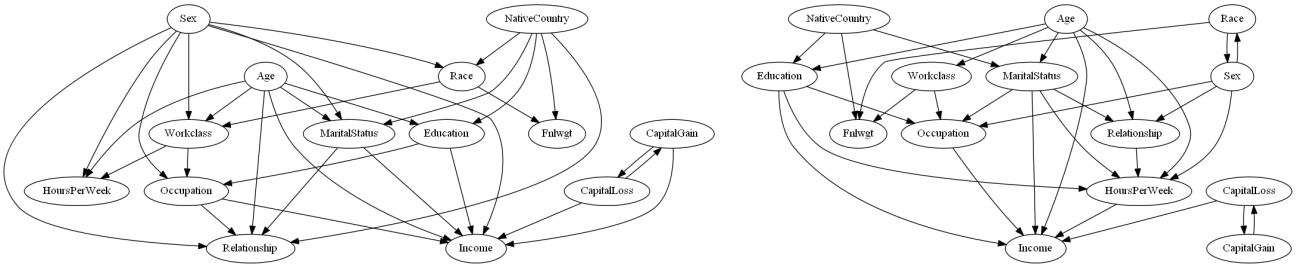
Table IX: The experimental results for different sample sizes n on numerical data ($\varepsilon = 1, p = 10, s = 0.4$)

n		Non-private PC	PrivPC*
10000	TPR	0.975±0.056	**0.975±0.056**
	FPR	0.094±0.096	**0.087±0.102**
	TDR	0.851±0.129	**0.865±0.143**
1000	TPR	1.000±0.000	**0.932±0.075**
	FPR	0.140±0.096	**0.196±0.082**
	TDR	0.746±0.153	**0.632±0.126**
100	TPR	1.000±0.000	0.433±0.435
	FPR	0.255±0.058	0.372±0.064
	TDR	0.463±0.060	0.081±0.091

Table X: The experimental results for different numbers of nodes p on numerical data ($\varepsilon = 1, n = 10000, s = 0.4$)

p		Non-private PC	PrivPC*
10	TPR	0.975±0.056	**0.975±0.056**
	FPR	0.094±0.096	**0.087±0.102**
	TDR	0.851±0.129	**0.865±0.143**
20	TPR	0.886±0.035	**0.853±0.048**
	FPR	0.299±0.024	**0.297±0.025**
	TDR	0.388±0.028	**0.401±0.050**
50	TPR	0.819±0.043	**0.785±0.082**
	FPR	0.374±0.009	**0.378±0.012**
	TDR	0.112±0.008	**0.094±0.019**
100	TPR	0.836±0.025	0.712±0.060
	FPR	0.393±0.008	**0.397±0.008**
	TDR	0.039±0.008	**0.023±0.004**

The sensitivity of the covariance matrix Σ_D in PrivPC* is $\frac{p(p+1)}{2(n-1)}$, which is a function of n and p. We evaluate our PrivPC* algorithm against the non-private algorithm over different sizes n and dimensions p. Table IX shows the comparison results with different n. When $n = 1000$ or larger, PrivPC* can achieve utility even as good as the non-



(a) Estimated CPDAG $\mathcal{G}$ by Non-private PC

(b) Estimated CPDAG $\hat{\mathcal{G}}$ by PrivPC

Figure 1: Case analysis on the UCI Adult dataset ($\varepsilon = 10, n = 48842, p = 14$)

Table XI: The experimental results for different values of sparseness s on numerical data ($\varepsilon = 1, n = 10000, p = 20$)

s		Non-private PC	PrivPC*
0.1	TPR	0.918±0.080	0.701±0.193
	FPR	0.004±0.003	**0.004±0.003**
	TDR	0.971±0.027	**0.971±0.027**
0.4	TPR	0.886±0.035	**0.853±0.048**
	FPR	0.299±0.024	**0.297±0.025**
	TDR	0.388±0.028	**0.401±0.050**
0.7	TPR	0.803±0.102	**0.789±0.078**
	FPR	0.684±0.039	**0.688±0.028**
	TDR	0.190±0.030	**0.179±0.029**

private PC algorithm. When the sample size is very small (e.g., $n = 100$), the effect of noise is magnified and the performance of PrivPC* significantly decreases.

Table X shows the comparison results over graphs with different dimensions p. In the low dimensional setting, PrivPC* can achieve utility as good as the non-private algorithm. In the high dimensional setting ($p = 50$ or more), the TDR values of both non-private and private PC algorithms are very low. When $p = 100$, the TPR of PrivPC* is significantly lower than the non-private PC algorithm. This is expected from our sensitivity analysis as the utility of PrivPC* will be compromised under a large p or small n. Yet, PrivPC* still achieves satisfactory performance in a wide range of cases ($n \geq 1000, p \leq 50$).

The sparseness s is another essential parameter of a causal graph other than the dimension p. We evaluate PrivPC* on causal graphs with same dimension but different sparseness. Table XI shows the comparison results for different sparseness s. For a dense causal graph ($s \geq 0.4$), PrivPC* achieves as good utility as the non-private algorithm. However, for a very sparse causal graph ($s = 0.1$), PrivPC* has a lower TPR than the non-private algorithm.

C. Case analysis on the UCI Adult dataset

We further conduct experiments on the UCI Adult datset [38]. It consists of 48842 records with 14 attributes such as age, education, sex, occupation, and income. For computational feasibility we discretize each attribute's domain values into two to four categories to reduce the domain sizes. There is no ground-truth graph of the Adult dataset. We apply the non-private PC algorithm to get an estimated CPDAG $\mathcal{G}$ (Figure 1a) and use it as the ground-truth; then we apply the PrivPC algorithm to get a differential privacy preserving CPDAG $\hat{\mathcal{G}}$ (Figure 1b) and compare it to $\mathcal{G}$.

The causal graph $\mathcal{G}$ produced by the original PC algorithm has 31 edges. The estimated CPDAG $\hat{\mathcal{G}}$ by the PrivPC algorithm has 28 edges. Most causal relationships discovered by the non-private algorithm are also captured by PrivPC. Specifically, PrivPC achieves 78.6% on TPR, 14.3% on FPR, and 71.0% on TDR.

VII. Conclusions and Future Work

In this work, we have studied how to preserve differential privacy in constraint-based causal graph discovery. The naive differentially private PC approach requires a large privacy budget since it conducts privacy protection at each conditional independence test. To solve this problem, we proposed a differentially private PC algorithm (PrivPC) based on the exponential mechanism. Our PrivPC significantly reduces the number of edge elimination decisions and hence reduces the required privacy budget. Our empirical evaluations demonstrated that PrivPC is robust over a wide range of parameter settings. We also developed a Laplace mechanism based differentially private PC algorithm (PrivPC*) for numerical data. The experimental results also showed that it achieves satisfactory performance.

The widely used PC algorithm still has some limitations including (a) vulnerability to errors in statistical independence test results, (b) no ranking or estimation of the confidence in the causal predictions, and (c) erroneous conclusions due to hidden variables. In future work, we will extend our study to other alternative causal modeling such as local causal discovery [10]–[12] and the Fast Causal Inference [4].

ACKNOWLEDGMENT

This work was supported in part to Depeng Xu and Xintao Wu by U.S. National Institute of Health (1R01GM103309) and National Science Foundation (DGE-1523115 and IIS-1502273) and to Shuhan Yuan by China Scholarship Council. This research was conducted while Shuhan Yuan visited University of Arkansas.

REFERENCES

[1] D. Edwards, *Introduction to graphical modelling*. Springer Science & Business Media, 2012.

[2] S. L. Lauritzen, *Graphical models*. Clarendon Press, 1996, vol. 17.

[3] P. Spirtes, C. N. Glymour, and R. Scheines, *Causation, prediction, and search*. MIT press, 2000.

[4] J. Pearl, *Causality*. Cambridge university press, 2009.

[5] M. J. Kusner, Y. Sun, K. Sridharan, and K. Q. Weinberger, "Private causal inference," *arXiv preprint arXiv:1512.05469*, 2015.

[6] P. O. Hoyer, D. Janzing, J. M. Mooij, J. Peters, and B. Schölkopf, "Nonlinear causal discovery with additive noise models," in *Advances in neural information processing systems*, 2009, pp. 689–696.

[7] F. McSherry and K. Talwar, "Mechanism Design via Differential Privacy," *48th Annual IEEE Symposium on Foundations of Computer Science (FOCS'07)*, pp. 94–103, 2007.

[8] D. Colombo and M. H. Maathuis, "Order-independent constraint-based causal structure learning," *Order-Independent Causal Structure Learning*, vol. 15, no. 2010, pp. 1–40, 2010.

[9] J. Ramsey, J. Zhang, and P. L. Spirtes, "Adjacency-Faithfulness and Conservative Causal Inference," *Proceedings of the Twenty-Second Conference on Uncertainty in Artificial Intelligence*, pp. 401–408, 2006.

[10] C. F. Aliferis, A. Statnikov, I. Tsamardinos, S. Mani, and X. D. Koutsoukos, "Local causal and markov blanket induction for causal discovery and feature selection for classification part i: Algorithms and empirical evaluation," *The Journal of Machine Learning Research*, vol. 11, pp. 171–234, 2010.

[11] ——, "Local causal and markov blanket induction for causal discovery and feature selection for classification part ii: Analysis and extensions," *The Journal of Machine Learning Research*, vol. 11, pp. 235–284, 2010.

[12] A. Statnikov, J. Lemeir, and C. F. Aliferis, "Algorithms for discovery of multiple markov boundaries," *The Journal of Machine Learning Research*, vol. 14, no. 1, pp. 499–566, 2013.

[13] M. Kalisch and P. Bühlmann, "Estimating high-dimensional directed acyclic graphs with the pc-algorithm," *Journal of Machine Learning Research*, vol. 8, no. Mar, pp. 613–636, 2007.

[14] N. Harris and M. Drton, "Pc algorithm for nonparanormal graphical models." *Journal of Machine Learning Research*, vol. 14, no. 1, pp. 3365–3383, 2013.

[15] X. Wu and Y. Ye, "Exploring gene causal interactions using an enhanced constraint-based method," *Pattern Recognition*, vol. 39, no. 12, pp. 2439–2449, 2006.

[16] K. Chaudhuri and C. Monteleoni, "Privacy-preserving logistic regression," in *NIPS'08*, 2008, pp. 289–296.

[17] D. Kifer and A. Machanavajjhala, "No free lunch in data privacy," in *SIGMOD'11*, 2011, pp. 193–204.

[18] U. Erlingsson, V. Pihur, and A. Korolova, "Rappor: Randomized aggregatable privacy-preserving ordinal response," in *CCS'14*, 2014, pp. 1054–1067.

[19] Y. Wang, X. Wu, and D. Hu, "Using randomized response for differential privacy preserving data collection," in *Proceedings of the Workshops of the EDBT/ICDT 2016 Joint Conference, EDBT/ICDT Workshops*, 2016.

[20] T.-H. H. Chan, M. Li, E. Shi, and W. Xu, "Differentially private continual monitoring of heavy hitters from distributed streams," in *PETS'12*, 2012, pp. 140–159.

[21] S. Song, K. Chaudhuri, and A. D. Sarwate, "Stochastic gradient descent with differentially private updates," in *GlobalSIP*, 2013, pp. 245–248.

[22] F. McSherry and I. Mironov, "Differentially Private Recommender Systems," in *KDD*. ACM, 2009.

[23] P. Jain, P. Kothari, and A. Thakurta, "Differentially private online learning," in *COLT*, 2012, pp. 24.1–24.34.

[24] X. Xiao, G. Wang, and J. Gehrke, "Differential privacy via wavelet transforms," in *ICDE'10*, 2010, pp. 225–236.

[25] Y. Wang, X. Wu, and L. Wu, "Differential privacy preserving spectral graph analysis," in *PAKDD (2)*, 2013, pp. 329–340.

[26] J. Zhang, G. Cormode, C. M. Procopiuc, D. Srivastava, and X. Xiao, "Privbayes: Private data release via bayesian networks," in *SIGMOD*. ACM, 2014, pp. 1423–1434.

[27] R. Shokri and V. Shmatikov, "Privacy-preserving deep learning." in *CCS'15*, 2015, pp. 1310–1321.

[28] M. Abadi, A. Chu, I. Goodfellow, H. B. McMahan, I. Mironov, K. Talwar, and L. Zhang, "Deep learning with differential privacy," in *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*. ACM, 2016, pp. 308–318.

[29] N. Phan, Y. Wang, X. Wu, and D. Dou, "Differential privacy preservation for deep auto-encoders: an application of human behavior prediction," in *Thirtieth AAAI Conference on Artificial Intelligence*, 2016.

[30] N. Phan, X. Wu, and D. Dou, "Preserving differential privacy in convolutional deep relief networks," *Machine Learning*, 2017.

[31] C. Dwork, F. McSherry, K. Nissim, and A. Smith, "Calibrating noise to sensitivity in private data analysis," *Theory of Cryptography*, pp. 265–284, 2006.

[32] J. Zhang, X. Xiao, Y. Yang, Z. Zhang, and M. Winslett, "PrivGene: Differentially private model fitting using genetic algorithms," *SIGMOD*, pp. 665–676, 2013.

[33] C. Dwork, "A firm foundation for private data analysis," *Communications of the ACM*, vol. 54, no. 1, p. 86, 2011.

[34] A. Friedman and A. Schuster, "Data mining with differential privacy," *Knowledge discovery and data mining*, pp. 493–502, 2010.

[35] A. Johnson, "Privacy-Preserving Data Exploration in Genome-Wide Association Studies Categories and Subject Descriptors," pp. 1079–1087, 2013.

[36] T. Verma and J. Pearl, "Equivalence and synthesis of causal models," in *Proceedings of Sixth Conference on Uncertainty in Artificial Intelligence*, 1991, pp. 220–227.

[37] S. A. Andersson, D. Madigan, M. D. Perlman *et al.*, "A characterization of markov equivalence classes for acyclic digraphs," *The Annals of Statistics*, vol. 25, no. 2, pp. 505–541, 1997.

[38] M. Lichman, "UCI machine learning repository," 2013. [Online]. Available: http://archive.ics.uci.edu/ml

[39] A. Hauser and P. Bühlmann, "Characterization and greedy learning of interventional Markov equivalence classes of directed acyclic graphs," *Journal of Machine Learning Research*, vol. 13, pp. 2409–2464, 2012.

[40] R. Scheines, P. Spirtes, C. Glymour, C. Meek, and T. Richardson, "The tetrad project: Constraint based aids to causal model specification," *Multivariate Behavioral Research*, vol. 33, no. 1, pp. 65–117, 1998.