# Efficient Intervention Design for Causal Discovery with Latents

Raghavendra Addanki* Shiva Prasad Kasiviswanathan† Andrew McGregor‡
Cameron Musco§

**Abstract**

We consider recovering a causal graph in presence of latent variables, where we seek to minimize the cost of interventions used in the recovery process. We consider two intervention cost models: (1) a linear cost model where the cost of an intervention on a subset of variables has a linear form, and (2) an identity cost model where the cost of an intervention is the same, regardless of what variables it is on, i.e., the goal is just to minimize the number of interventions. Under the linear cost model, we give an algorithm to identify the ancestral relations of the underlying causal graph, achieving within a 2-factor of the optimal intervention cost. This approximation factor can be improved to $1+\epsilon$ for any $\epsilon > 0$ under some mild restrictions. Under the identity cost model, we bound the number of interventions needed to recover the entire causal graph, including the latent variables, using a parameterization of the causal graph through a special type of colliders. In particular, we introduce the notion of $p$-colliders, that are colliders between pair of nodes arising from a specific type of conditioning in the causal graph, and provide an upper bound on the number of interventions as a function of the maximum number of $p$-colliders between any two nodes in the causal graph.

## 1 Introduction

Causality has long been a key tool in studying and analyzing various behaviors in fields such as genetics, psychology, and economics [Pearl, 2009]. Causality also plays a pivotal role in helping us build systems that can understand the world around us, and in turn, in helping us understand the behavior of machine learning systems deployed in the real world. Although the theory of causality has been around for more than three decades, for these reasons it has received increasing attention in recent years. In this paper, we study one of the fundamental problems of causality: *causal discovery*. In causal discovery, we want to learn all the causal relations existing between variables (nodes of the causal graph) of our system. It has been shown that, under certain assumptions, observational data alone only lets us recover *the existence of a causal relationship* between two variables, but not the *direction* of all relationships. To recover the directions of causal edges, we use the notion of an *intervention* described in the Structural Causal Models (SCM) framework introduced by [Pearl, 2009].

An intervention requires us to fix a subset of variables to a set of values, inducing a new distribution on the free variables. Such a system manipulation is generally expensive and thus there has been significant interest in trying to minimize the number of interventions and their cost in causal discovery. In a general cost model, intervening on any subset of variables has a cost associated with

*UMass Amherst. raddanki@cs.umass.edu. Part of this work was done while the author was an intern at Amazon.
†Amazon. kasivisw@gmail.com
‡UMass Amherst. mcgregor@cs.umass.edu
§UMass Amherst. cmusco@cs.umass.edu

it, and the goal is to identify all causal relationships and their directions while minimizing the total cost of interventions applied. This captures the fact that some interventions are more expensive than others. For example, in a medical study, intervening on certain variables might be impractical or unethical. In this work, we study two simplifications of this general cost model. In the *linear cost model*, each variable has an intervention cost, and the cost of an intervention on a subset of variables is the sum of costs for each variable in the set [Kocaoglu et al., 2017a; Lindgren et al., 2018]. In the *identity cost model*, every intervention has the same cost, regardless of what variables it contains and therefore minimizing intervention cost is the same as minimizing the number of interventions [Kocaoglu et al., 2017b].

As is standard in the causality literature, we assume that our causal relationship graph satisfies the *causal Markov condition* and *faithfulness* [Spirtes et al., 2000]. We assume that faithfulness holds both in the observational and interventional distributions following [Hauser and Bühlmann, 2014]. As is common, we also assume that we are given access to an oracle that can check if two variables are independent, conditioned on a subset of variables. We discuss this assumption in more detail in Section 2. Unlike much prior work, we *do not make* the causal sufficiency assumption: that there are no unobserved (or latent) variables in the system. Our algorithms apply to the causal discovery problem with the existence of latent variables.

**Results.** Our contributions are as follows. Let $\mathcal{G}$ be a causal graph on both observable variables $V$ and latent variables $L$. A directed edge $(u, v)$ in $\mathcal{G}$ indicates a causal relationship from $u$ to $v$. Let $G$ be the induced subgraph of $\mathcal{G}$ on the $n$ observable variables (referred to as observable graph). See Section 2 for a more formal description.

**Linear Cost Model.** In the linear cost model, we give an algorithm that given $m = \Omega(\log n)$ where $n$ is the number of observed variables, outputs a set of $m$ interventions that can be used to recover all ancestral relations of the observable graph $G$.[1] We show that cost of interventions generated by the algorithm is at most twice the cost of the optimum set of interventions for this task. Our result is based on a characterization that shows that generating a set of interventions sufficient to recover ancestral relations is equivalent to designing a *strongly separating set system* (Def. 2.2). We show how to design such a set system with at most twice the optimum cost based on a greedy algorithm that constructs intervention sets which includes a variable with high cost in the least number of sets possible.

In the special case when each variable has unit intervention cost [Hyttinen et al., 2013a] give an exact algorithm to recover ancestral relations in $G$ with minimal total cost. Their algorithm is based on the Kruskal-Katona theorem in combinatorics [Kruskal, 1963; Katona, 1966]. We show that a modification of this approach yields a $(1 + \epsilon)$-approximation algorithm in the general linear cost model for any $0 < \epsilon \leq 1$, under mild assumptions on $m$ and the maximum intervention cost on any one variable.

The linear cost model was first considered in [Kocaoglu et al., 2017a] and studied under the causal sufficiency (no latents) assumption. [Lindgren et al., 2018] showed given the *essential graph* of a causal graph, the problem of recovering $G$ with optimal cost under the linear cost model is NP-hard. To the best of our knowledge, our result is the first to minimize intervention cost under the popular linear cost model in the presence of latents, and without the assumption of unit intervention cost on each variable.

We note that, while we give a 2-approximation for recovering ancestral relations in $G$, under the linear cost model, there seems to be no known characterization of the optimal intervention sets needed to recover the entire causal graph $\mathcal{G}$, making it hard to design a good approximation here.

[1] As noted in Section 3, $m \geq \log n$ is a lower bound for any solution.

Tackling this problem in the linear cost model is an interesting direction for future work.

**Identity Cost Model**. In the identity cost model, where we seek to just minimize the number of interventions, recovering ancestral relations in $G$ with minimum cost becomes trivial (see Section 4). Thus, in this case, we focus on algorithms that recover the causal graph $\mathcal{G}$ completely. We start with the notion of *colliders* in causal graphs [Pearl, 2009]. Our idea is to parametrize the causal graph in terms of a specific type of colliders that we refer to as $p$-colliders (Def. 4.2). Intuitively, a node $v_k$ is $p$-collider for a pair of nodes $(v_i, v_j)$ if a) it is a collider on a path between $v_i$ and $v_j$ and b) at least one of the parents $v_i, v_j$ is a descendent of $v_k$. If the graph $\mathcal{G}$ has at most $\tau$ $p$-colliders between every pair of nodes, then our algorithm uses at most $O(n\tau \log n + n \log n)$ interventions. We also present causal graph instances where any non-adaptive algorithm requires $\Omega(n)$ interventions.

The only previous bound on recovering $\mathcal{G}$ in this setting utilized $O(\min\{d \log^2 n, \ell\} + d^2 \log n)$ many interventions where $d$ is the maximum (undirected) node degree and $\ell$ is the length of the longest directed path of the causal graph [Kocaoglu et al., 2017b]. Since we use a different parametrization of the causal graph, a direct comparison with this result is not always possible. We argue that a parameterization in terms of $p$-colliders is inherently more "natural" as it takes the directions of edges in $\mathcal{G}$ into account whereas the maximum degree does not. The presence of a single high-degree node can make the number of interventions required by existing work extremely high, even if the overall causal graph is sparse. In this case, the notion of $p$-colliders is a more global characterization of a causal graph. In Section 5, we experimentally show that our bound improves the bound of [Kocaoglu et al., 2017b] in some popular random graph models.

### 1.1 Other Related Work

Broadly, the problem of causal discovery has been studied under two different settings. In the first, one assumes *causal sufficiency*, i.e., that there are no unmeasured (latent) variables. Most work in this setting focuses on recovering causal relationships based on just observational data. Examples include algorithms like *IC* [Pearl, 2009] and *PC* [Spirtes et al., 2000]. Much work has focused on understanding the limitations and assumptions underlying these algorithms [Hauser and Bühlmann, 2014; Hoyer et al., 2009; Heinze-Deml et al., 2018; Loh and Bühlmann, 2014; Hoyer et al., 2009; Shimizu et al., 2006]. It is well-known, that to disambiguate a causal graph from its equivalence class, interventional, rather than just observational data is required [Hauser and Bühlmann, 2012; Eberhardt and Scheines, 2007; Eberhardt, 2007]. In particular, letting $\chi(\mathcal{G})$ be the chromatic number of $G$, $\Theta(\log \chi(\mathcal{G}))$ interventions are necessary and sufficient for recovery under the causal sufficiency assumption [Hauser and Bühlmann, 2014]. Surprising connections have been found [Hyttinen et al., 2013a; Katona, 1966; Mao-Cheng, 1984] between combinatorial structures and causality. Using these connections, much recent work has been devoted to minimizing intervention cost while imposing constraints such as sparsity or different costs for different sets of nodes [Shanmugam et al., 2015; Kocaoglu et al., 2017a; Lindgren et al., 2018].

In many cases, causal sufficiency is too strong an assumption: it is often contested if the behavior of systems we observe can truly be attributed to measured variables [Pearl, 2000; Bareinboim and Pearl, 2016]. In light of this, many algorithms avoiding the causal sufficiency assumption, such as IC$^*$ [Verma and Pearl, 1992] and FCI [Spirtes et al., 2000], have been developed. The above algorithms only use observational data. However, there is a growing interest in optimal intervention design in this setting [Silva et al., 2006; Hyttinen et al., 2013b; Parviainen and Koivisto, 2011]. We contribute to this line of work, focusing on minimizing the intervention cost required to recover the full intervention graph, or its ancestral graph, without causal sufficiency, i.e., in the presence of latents.

## 2 Preliminaries

**Notation.** Following the SCM framework introduced by Pearl [2009], we represent the set of random variables of interest by $V \cup L$ where $V$ represents the set of endogenous (observed) variables that can be measured and $L$ represents the set of exogenous (latent) variables that cannot be measured. We define a directed graph on these variables where an edge corresponds to a causal relation between the corresponding variables. The edges are directed with an edge $(v_i, v_j)$ meaning that $v_i \to v_j$. As is common, we assume that all causal relations that exist between random variables in $V \cup L$ belong to one of the two categories : (i) $E \subseteq V \times V$ containing causal relations between the observed variables and (ii) $E_L \subseteq L \times V$ containing relations of the form $l \to v$ where $l \in L, v \in V$. Thus, the full edge set of our causal graph is denoted by $\mathcal{E} = E \cup E_L$. We also assume that every latent $l \in L$ influences exactly two observed variables i.e., $(l, u), (l, v) \in E_L$ and no other edges are incident on $l$ following [Kocaoglu et al., 2017b]. We let $\mathcal{G} = \mathcal{G}(V \cup L, \mathcal{E})$ denote the entire causal graph and refer to $G = G(V, E)$ as the *observable graph*.

Unless otherwise specified a path between two nodes is a undirected path. For every observable $v \in V$, let the parents of $v$ be defined as $\text{Pa}(v) = \{w \mid w \in V \text{ and } (w, v) \in E\}$. For a set of nodes $S \subseteq V$, $\text{Pa}(S) = \cup_{v \in S} \text{Pa}(v)$. If $v_i, v_j \in V$, we say $v_j$ is a descendant of $v_i$ (and $v_i$ is an ancestor of $v_j$) if there is a directed path from $v_i$ to $v_j$. $\text{Anc}(v) = \{w \mid w \in V \text{ and } v \text{ is a descendant of } w\}$. We let $\text{Anc}(G)$ denote the *ancestral graph*[2] of $G$ where an edge $(v_i, v_j) \in \text{Anc}(G)$ if and only if there is a directed path from $v_i$ to $v_j$ in $G$. One of our primary interests is in recovering $\text{Anc}(G)$ using a minimal cost set of interventions.

Using Pearl's do-notation, we represent an intervention on a set of variables $S \subseteq V$ as $\text{do}(S = s)$ for a value $s$ in the domain of $S$ and the joint probability distribution on $V \cup L$ conditioned on this intervention by $\Pr[\cdot \mid do(S)]$.

We assume that there exists an oracle that answers queries such as *"Is $v_i$ independent of $v_j$ given $Z$ in the interventional distribution* $\Pr[\cdot \mid \text{do}(S = s)]$*?"*

**Assumption 2.1** (Conditional Independence (CI)-Oracle)**.** *Given any $v_i, v_j \in V$ and $Z, S \subseteq V$ we have an oracle that tests whether $v_i \perp\!\!\!\perp v_j \mid Z, \text{do}(S = s)$.*

Such conditional independence tests have been widely investigated with sublinear (in domain size) bounds on the sample size needed for implementing this oracle [Canonne et al., 2018; Zhang et al., 2011].

**Intervention Cost Models.** We study the causal discovery problem under two cost models:

1. *Linear Cost Model.* In this model, each node $v \in V$ has a different cost $c(v) \in \mathbb{R}^+$ and the cost of intervention on a set $S \subset V$ is defined as $\sum_{v \in S} c(v)$ (akin to [Lindgren et al., 2018]). That is, interventions that involve a larger number of, or more costly nodes, are more expensive. Our goal is to find an intervention set $\mathcal{S}$ minimizing $\sum_{S \in \mathcal{S}} \sum_{v \in S} c(v)$. We constrain the number of interventions to be upper bounded by some budget $m$. Without such a bound, we can observe that for ancestral graph recovery, the optimal intervention set is $\mathcal{S} = \{\{v_1\}, \{v_2\}, \ldots, \{v_n\}\}$ with cost $\sum_{v \in V} c(v)$ as intervention on every variable is necessary. The optimality of $\mathcal{S}$ here follows from a characterization of any feasible set system we establish in Lemma 3.1.

2. *Identity Cost Model.* As an intervention on a set of variables requires controlling the variables, and generating a new distribution, we want to use as few interventions as possible. In this

[2]We note that the term *ancestral graph* has also been previously used in a different context, see e.g., [Richardson et al., 2002].

cost model, an intervention on any set of observed variables has *unit cost* (no matter how many variables are in the set). We assume that for any intervention, querying the CI-oracle comes free of cost. This model is akin to the model studied in Kocaoglu et al. [2017b].

**Causal Discovery Goals.** We will study two variations of the causal discovery problem. In the first, we aim to recover the ancestral graph $\text{Anc}(G)$, which contains all the causal ancestral relationships between the observable variables $V$. In the second, our goal is to recover all the causal relations in $\mathcal{E}$, i.e., learn the entire causal graph $\mathcal{G}(V \cup L, \mathcal{E})$. We aim to perform both tasks using a set of intervention sets $\mathcal{S} = \{S_1, \ldots, S_m\}$ (each $S_i \subseteq V$) with minimal cost, with our cost models defined above.

For ancestral graph recovery, we will leverage a simple characterization of when a set of interventions $\mathcal{S} = \{S_1, \ldots, S_m\}$ is sufficient to recover $\text{Anc}(G)$. In particular, $\mathcal{S}$ is sufficient if it is a *strongly separating set system* [Kocaoglu et al., 2017b].

**Definition 2.2** (Strongly Separating Set System)**.** *A collection of subsets $\mathcal{S} = \{S_1, \cdots, S_m\}$ of the ground set $V$ is a strongly separating set system if for every distinct $u, v \in V$ there exists $S_i$ and $S_j$ such that $u \in S_i \setminus S_j$ and $v \in S_j \setminus S_i$.*

Ancestral graph recovery using a strongly separating set system is simple: we intervene on each of the sets $S_1, \ldots, S_m$. Using CI-tests we can identify for every pair of $v_i$ and $v_j$, if there is a path from $v_i$ to $v_j$ or not in $G$ using the intervention corresponding to $S \in \mathcal{S}$ with $v_i \in S$ and $v_j \notin S$. We add an edge to $\text{Anc}(G)$ if the test returns dependence. Finally, we take the transitive closure and output the resulting graph as $\text{Anc}(G)$. In Lemma 3.1, we show that in fact being strongly separating is *necessary* for any set of interventions to be used to identify $\text{Anc}(G)$.

# 3 Linear Cost Model

We begin with our results on recovering the ancestral graph $\text{Anc}(G)$ in the linear cost model. Recall that, given a budget of $m$ interventions, our objective is to find a set of interventions $\mathcal{S} = \{S_1, S_2, \cdots S_m\}$ that can be used to identify $\text{Anc}(G)$ while minimizing $\sum_{S \in \mathcal{S}} \sum_{v \in S} c(v)$.

As detailed in Section 2, a strongly separating set system is sufficient to recover the ancestral graph. We show that it also necessary: a set of interventions to discover $\text{Anc}(G)$ must be a strongly separating set system (Definition 2.2). See proof in Appendix A.

**Lemma 3.1.** *Suppose $\mathcal{S} = \{S_1, S_2, \cdots, S_m\}$ is a collection of subsets of $V$. For a given causal graph $G$ if $\text{Anc}(G)$ is recovered using CI-tests by intervening on the sets $S_i \in \mathcal{S}$. Then, $\mathcal{S}$ is a strongly separating set system.*

Given this characterization, the problem of constructing the ancestral graph $\text{Anc}(G)$ with minimum linear cost *reduces* to that of constructing a strongly separating set system with minimum cost. In developing our algorithm for finding such a set system, it will be useful to represent a set system by a binary matrix, with rows corresponding to observable variables $V$ and columns corresponding to interventions (sets $S_1, \ldots, S_m$).

**Definition 3.2** (Strongly Separating Matrix)**.** *Matrix $U \in \{0,1\}^{n \times m}$ is a* strongly separating matrix *if $\forall i, j \in [n]$ there exists $k, k' \in [m]$ such that $U(i,k) = 1, U(j,k) = 0$ and $U(i,k') = 0, U(j,k') = 1$.*

Note that given a strongly separating set system $\mathcal{S}$, if we let $U$ be the matrix where $U(i,k) = 1$ if $v_i \in S_k$ and 0 otherwise, $U$ will be a strongly separating matrix. The other direction is also true.

Let $U(j)$ denote the $j$th row of $U$. Using Definition 3.2 and above connection between recovering $\mathrm{Anc}(G)$ and strongly separating set system, we can reformulate the problem at hand as:

$$\min_U \sum_{j=1}^{n} c(v_j) \cdot \|U(j)\|_1 \tag{1}$$
$$\text{s.t. } U \in \{0,1\}^{n \times m} \text{ is a strongly separating matrix.}$$

We can thus view our problem as finding an assignment of vectors in $\{0,1\}^m$ (i.e., rows of $U$) to nodes in $V$ that minimizes (1). Throughout, we will call $\|U(j)\|_1$ the *weight* of row $U(j)$, i.e., the number of 1s in that row. It is easy to see that $m \geq \log n$ is necessary for a feasible solution to exist as each row must be distinct.

We start by giving a 2-approximation algorithm for (1). In Section 3.2, we show how to obtain an improved approximation under certain assumptions.

### 3.1 2-approximation Algorithm

In this section, we present an algorithm (Algorithm SSMatrix) that constructs a strongly separating matrix (and a corresponding intervention set) which minimizes (1) to within a 2-factor of the optimum. Missing details from section are collected in Appendix A.1.

**Outline.** Let $U_{\mathrm{OPT}}$ denote a strongly separating matrix minimizing (1). Let $c_{\mathrm{OPT}} = \sum_{j=1}^{n} c(v_j)\|U_{\mathrm{OPT}}(j)\|_1$ denote the objective value achieved by this optimum $U_{\mathrm{OPT}}$. We start by relaxing the constraint on $U$ so that it does *not need to be strongly separating*, but just must have unique rows, where none of the rows is all zero. In this case, we can optimize (1) very easily. We simply take the rows of $U$ to be the $n$ unique binary vectors in $\{0,1\}^m \setminus \{0^m\}$ with lowest weights. That is, $m$ rows will have weight 1, $\binom{m}{2}$ will have weight 2, etc. We then assign the rows to the nodes in $V$ in descending order of their costs. So the $m$ nodes with the highest costs will be assigned the weight 1 rows, the next $\binom{m}{2}$ assigned weight 2 rows, etc. The cost of this assignment is only lower than $c_{\mathrm{OPT}}$, as we have only relaxed the constraint in (1).

We next convert this relaxed solution into a valid strongly separating matrix. Given $m + \log n$ columns, we can do this easily. Since there are $n$ nodes, in the above assignment, all rows will have weight at most $\log n$. Let $\bar{U} \in \{0,1\}^{m+\log n}$ have its first $m$ columns equal to those of $U$. Additionally, use the last $\log n$ columns as 'row weight indicators': if $\|U(j)\|_1 = k$ then set $\bar{U}(j, m+k) = 1$. We can see that $\bar{U}$ is a strongly separating matrix. If two rows have different weights $k, k'$ in $\bar{U}$, then the last $\log n$ columns ensure that they satisfy the strongly separating condition. If they have the same weight in $\bar{U}$, then they already satisfy the condition, as to be unique in $U$ they must have a at least 2 entries on which they differ.

To turn the above idea into a valid approximation algorithm that outputs $\bar{U}$ with just $m$ (not $m + \log n$) columns, we argue that we can 'reserve' the last $\log n$ columns of $\bar{U}$ to serve as weight indicator columns. We are then left with just $m - \log n$ columns to work with. Thus we can only assign $m - \log n$ weight 1 rows, $\binom{m-\log n}{2}$ weight 2 rows, etc. Nevertheless, if $m \geq \gamma \log n$ (for a constant $\gamma > 1$), this does not affect the assignment much: for any $i$ we can still 'cover' the $\binom{m}{i}$ weight $i$ rows in $U$ with rows of weight $\leq 2i$. Thus, after accounting for the weight indicator columns, each weight $k$ row in $U$ has weight $\leq 2k+1$ in $\bar{U}$. Overall, this gives us a 3-approximation algorithm: when $k$ is 1 the weight of a row may become as large as 3.

To improve the approximation to a 2-approximation we *guess* the number of weight 1 vectors $a_1$ in the optimum solution $U_{\mathrm{OPT}}$ and assign the $a_1$ highest cost variables to weight 1 vectors, achieving optimal cost for these variables. There are $O(m)$ possible values for $a_1$ and so trying all guesses is still efficient. We then apply our approximation algorithm to the remaining $m - a_1$ available

**Algorithm 1** SSMatrix$(V, m)$

$c_{U_{min}} \leftarrow \infty$
**for** $a_1 \in \{0, 1, \cdots, 2m/3\}$ **do**
  $U \in \{0,1\}^{n \times m}$ be initialized with all zeros
  Assign the highest cost $a_1$ nodes with unit weight vectors such that $U(i,i) = 1$ for $i \leq a_1$
  Set $m' \leftarrow m - a_1$
  Mark all vectors of weight *at least* 1 in $\{0,1\}^{m' - \log n}$ as available
  **for** unassigned $v_i \in V$ (in decreasing order of cost) **do**
    Set $U(i, (a_1+1) : m - \log n)$ to smallest available weight vector in $\{0,1\}^{m' - \log n}$ and make this vector unavailable. Let the weight of the assigned vector be $k$
    Set 'row weight indicator' $U(i, m' - \log n + k) = 1$
  **end for**
  Compute cost of objective for $U$ be $c_U$
  **if** $c_U < c_{U_{min}}$ **then**
    $c_{U_{min}} \leftarrow c_U, U_{min} \leftarrow U$
  **end if**
**end for**
Return $U_{min}$

columns of $U$ and $n - a_1$ variables. Since no variables are assigned weight 1 in this set, we achieve a tighter 2-approximation using our approach. The resulting matrix has the form:

$$U = \left(\begin{array}{c|c|c} \mathbb{I}_{a_1} & 0 & 0 \\ \hline 0 & C_1 & M_1 \\ \hline 0 & C_2 & M_2 \\ \hline \vdots & \vdots & \vdots \end{array}\right)$$

where $\mathbb{I}_{a_1}$ is the $a_1 \times a_1$ identity matrix, the rows of $C_w$ are all weight $w$ binary vectors of length $m - \log n - a_1$, and the rows of $M_w$ are length $\log n$ binary vectors with 1's in the $w$th column. The entire approach is presented in Algorithm SSMatrix and a proof of the approximation bound in Theorem 3.3 is present in Appendix A.1.

**Theorem 3.3.** *Let $m \geq \gamma \log n$ for constant $\gamma > 1$ and $U$ be the strongly separating matrix returned by* SSMatrix.[3] *Let $c_U = \sum_{j=1}^{n} c(v_j) \|U(j)\|_1$. Then, $c_U \leq 2 \cdot c_{\text{OPT}}$, where $c_{\text{OPT}}$ is the objective value associated with optimum set of interventions corresponding to $U_{\text{OPT}}$.*

Using the interventions from the matrix $U$ returned by Algorithm SSMatrix, we obtain a cost within twice the optimum for recovering $\text{Anc}(G)$.

### 3.2 $(1+\epsilon)$-approximation Algorithm

In [Hyttinen et al., 2013a], the authors show how to construct a collection $\mathcal{A}$ of $m$ strongly separating intervention sets with minimum average set size, i.e., $\sum_{A \in \mathcal{A}} |A|/m$. This is equivalent to minimizing the objective (1) in the linear cost model when the cost of intervening on any node equals 1. In this section, we analyze an adaptation of their algorithm to the general linear cost model, and obtain a $(1+\epsilon)$-approximation for any given $0 < \epsilon \leq 1$, an improvement over the 2-approximation of Section 3.1. Our analysis requires mild restrictions on the number of interventions and an upper bound on the maximum cost. The algorithm will not depend on $\epsilon$ but these bounds will. Missing details from this section are collected in Appendix A.2.

[3] In our proof, $\gamma = 66$ but this can likely be decreased.

**Algorithm** $\epsilon$-SSMATRIX **Outline**. The famous Kruskal-Katona theorem in combinatorics forms the basis of the scheme presented in [Hyttinen et al., 2013a] for minimizing the average size of the intervention sets. To deal with with varying costs of node interventions, we augment this approach with a greedy strategy. Let $\mathcal{A}$ denote a set of $m$ interventions sets over the nodes $\{v_1, v_2 \cdots, v_n\}$ obtained using the scheme from [Hyttinen et al., 2013a]. Construct a strongly separating matrix $\widetilde{U}$ from $\mathcal{A}$ with $\widetilde{U}(i,j) = 1$ iff $v_i \in A_j$ for $A_j \in \mathcal{A}$. Let $\zeta$ denote the ordering of rows of $\widetilde{U}$ in the increasing order of weight. Our Algorithm $\epsilon$-SSMATRIX outputs the strongly separating matrix $U$ where, for every $i \in [n]$, $U(i) = \widetilde{U}(\zeta(i))$ and the $i$th row of $U$ corresponds to the node with $i$th largest cost.

Let $c_{max} = \max_{v_i \in V} c(v_i) / \min_{v_i \in V} c(v_i)$ be the ratio of maximum cost to minimum cost of nodes in $V$. For ease of analysis, we assume that the cost of any node is least 1.

**Theorem 3.4.** *Let $U$ be the strongly separating matrix returned by $\epsilon$-SSMATRIX. If $c_{max} \leq \frac{\epsilon n}{3\binom{m}{t}}$ for $0 < \epsilon \leq 1$ where $\binom{m}{k-1} < n \leq \binom{m}{k}$ and $t = \lfloor k - \epsilon k/3 \rfloor$, then,*

$$c_U := \sum_{j=1}^{n} c(v_j) \left\| U(j) \right\|_1 \leq (1+\epsilon) \cdot c_{\text{OPT}} \ ,$$

*where $c_{\text{OPT}}$ is the objective value associated with optimum set of interventions corresponding to $U_{\text{OPT}}$.*

*Proof.* Suppose the optimal solution $U_{\text{OPT}}$ includes $a_q^*$ vectors of weight $q$. Let $S$ be the $a_1^* + a_2^* + \ldots + a_t^*$ nodes with highest cost in $U_{\text{OPT}}$. Since $a_q^* \leq \binom{m}{q}$, it immediately follows that $|S| \leq \sum_{i=q}^{t} \binom{m}{q}$. However, a slightly tighter analysis (see Lemma A.10) implies $|S| \leq \binom{m}{t}$. Let $c_{\text{OPT}}(S)$ be the total contribution of the nodes in $S$ to $c_{\text{OPT}}$. Let $c_U(S)$ denote the sum of contribution of the nodes in $S$ to $c_U$ for the matrix $U$ returned by $\epsilon$-SSMATRIX. Let $\bar{k}_{|S|}$ and $\bar{k}_n$ be the average of the smallest $|S|$ and $n$ respectively of the vector weights assigned by the algorithm. It is easy to observe that $\bar{k}_{|S|} \leq \bar{k}_n$.

$$\begin{aligned} c_U(S) &= \sum_{v_i \in S} c(v_i) \left\| U(i) \right\|_1 \leq c_{max} \sum_{v_i \in S} \left\| U(i) \right\|_1 \\ &= c_{max} \bar{k}_{|S|} |S| \leq c_{max} \bar{k}_{|S|} \binom{m}{t} \leq \epsilon \bar{k}_{|S|} n/3. \end{aligned}$$

As every node in $V \setminus S$ receives weight at least $t = k - \epsilon k/3$ in $U_{\text{OPT}}$ and at most $k$ in $U$ returned by $\epsilon$-SSMATRIX, we have $c_U(V \setminus S) \leq \frac{c_{\text{OPT}}(V \setminus S)}{1 - \epsilon/3}$. Now, we give a lower bound on the cost of the optimum solution $c_{\text{OPT}}(V)$. We know that when costs of all the nodes are 1, then $\epsilon$-SSMATRIXachieves optimum cost denoted by $c'_{\text{OPT}}(V)$ (see Appendix A.2 for more details). As all the nodes of $V$ have costs more than 1, we have:

$$c_{\text{OPT}}(V) \geq c'_{\text{OPT}}(V) = \bar{k}_n \cdot n \geq \bar{k}_{|S|} \cdot n.$$

Hence,

$$\frac{c_U(V)}{c_{\text{OPT}}(V)} \leq \frac{c_U(S)}{\bar{k}_{|S|} n} + \frac{c_U(V \setminus S)}{c_{\text{OPT}}(V \setminus S)} \leq \frac{\epsilon}{3} + \frac{1}{1 - \epsilon/3} \leq 1 + \epsilon.$$

This completes the proof.

□

By bounding the binomial coefficients in Thm. 3.4, we obtain the following somewhat easier to interpret corollary:

**Corollary 3.5.** *If $c_{max} \leq (\epsilon/6)n^{\Omega(\epsilon)}$ and either a) $n^{\epsilon/6} \geq m \geq (2\log_2 n)^{c_1}$ for some constant $c_1 > 1$ or b) $4\log_2 n \leq m \leq c_2 \log_2 n$ for some constant $c_2$ then the Algorithm $\epsilon$-SSMatrix returns an $(1+\epsilon)$-approximation.*

# 4 Identity Cost Model

In this section, we consider the identity cost model, where the cost of intervention for any subset of variables is the same. Our goal is to construct the entire causal graph $\mathcal{G}$, while minimizing the number of interventions. Our algorithm is based on parameterizing the causal graph based on a specific type of collider structure.

Before describing our algorithms, we recall the notion of $d$-separation and introduce this specific type of colliders that we rely on. Missing details from section are collected in Appendix B.

**Colliders.** Given a causal graph $\mathcal{G}(V \cup L, \mathcal{E})$, let $v_i, v_j \in V$ and a set of nodes $Z \subseteq V$. We say $v_i$ and $v_j$ are *d-separated* by $Z$ if and only if every undirected path $\pi$ between $v_i$ and $v_j$ is *blocked* by $Z$. A path $\pi$ between $v_i$ and $v_j$ is blocked by $Z$ if at least one of the following holds.

**Rule 1:** $\pi$ contains a node $v_k \in Z$ such that the path $\pi = v_i \cdots \rightarrow v_k \rightarrow \cdots v_j$ or $v_i \cdots \leftarrow v_k \leftarrow \cdots v_j$.

**Rule 2:** $\pi = v_i \cdots \rightarrow v_k \leftarrow \cdots v_j$ contains a node $v_k$ and both $v_k \notin Z$ and no descendant of $v_k$ is in $Z$.

**Lemma 4.1.** *[Pearl, 2009] If $v_i$ and $v_j$ are d-separated by $Z$, then $v_i \perp\!\!\!\perp v_j \mid Z$.*

For the path $\pi = v_i \cdots \rightarrow v_k \leftarrow \cdots v_j$ between $v_i$ and $v_j$, $v_k$ is called a *collider* as there are two arrows pointing towards it. We say that $v_k$ is a collider for the pair $v_i$ and $v_j$, if there exists a path between $v_i$ and $v_j$ for which $v_k$ is a collider. As shown by Rule 2, colliders play an important role in $d$-separation. We give a more restrictive definition for colliders that we will rely on henceforth.

**Definition 4.2** ($p$-**colliders**). *Given a causal graph $\mathcal{G}(V \cup L, E \cup E_L)$. Consider $v_i, v_j \in V$ and $v_k \in V$. We say $v_k$ is a p-collider for the pair $v_i$ and $v_j$, if there exists a path $v_i \cdots \rightarrow v_k \leftarrow \cdots v_j$ in $\mathcal{G}$ and either $v_k \in \mathrm{Pa}(v_i) \cup \mathrm{Pa}(v_j)$ or has at least one descendant in $\mathrm{Pa}(v_i) \cup \mathrm{Pa}(v_j)$. Let $P_{ij} \subset V$ denote all the p-colliders between $v_i$ and $v_j$.*

Now between every pair of observable variables, we can define a set of $p$-colliders as above. Computing $P_{ij}$ explicitly requires the knowledge of $\mathcal{G}$, however as we show below we can use randomization to overcome this issue. The following parameterization of a causal graph will be useful in our discussions.

**Definition 4.3** ($\tau$-causal graph). *A causal graph $\mathcal{G}(V \cup L, \mathcal{E})$ is a $\tau$-causal graph if for every pair of nodes in $V$ the number of p-colliders is at most $\tau$, i.e., $v_i, v_j \in V$ $(i \neq j)$, we have $|P_{ij}| \leq \tau$.*

Note that every causal graph is at least $n-2$-causal. In practice, we would expect $\tau$ to be significantly smaller. Given a causal graph $\mathcal{G}$, it is easy to determine the minimum values of $\tau$ for which it is $\tau$-causal, as checking for $p$-colliders is easy. Our algorithm recovers $\mathcal{G}$ with number of interventions that grow as a function of $\tau$ and $n$.

**Outline of our Approach.** Let $\mathcal{G}$ be a $\tau$-causal graph. As in [Kocaoglu et al., 2017b], we break our approach into multiple steps. Firstly, we construct the ancestral graph $\mathrm{Anc}(G)$ using the strongly

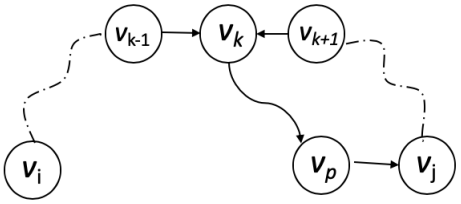


Figure 1: $v_k$ is a $p$-collider for $v_i, v_j$ as it has a path to $v_p$, a parent of $v_j$.

separating set system (Definition 2.2) idea detailed in Section 2. For example, a strongly separating set system can be constructed with $m = 2\log n$ interventions by using the binary encoding of the numbers $1, \cdots, n$ [Kocaoglu et al., 2017b]. After that the algorithm has two steps. In the first step, we recover the observable graph $G$ from $\text{Anc}(G)$. In the next step, after obtaining the observable graph, we identify all the latents $L$ between the variables in $V$ to construct $\mathcal{G}$. In both these steps, an underlying idea is to construct intervention sets with the aim of making sure that all the $p$-colliders between every pair of nodes is included in at least one of the intervention sets. As we do not know the graph $\mathcal{G}$, we devise randomized strategies to hit all the $p$-colliders, whilst ensuring that we do not create a lot of interventions.

A point to note is that, we design the algorithms to achieve an overall success probability of $1-O(1/n^2)$, however, the success probability can be boosted to any $1-O(1/n^c)$ for any constant $c$, by just adjusting the constant factors (see for example the proof of Lemma B.2). Also for simplicity of discussion, we assume that we know $\tau$. However as we discuss in Appendix B this assumption can be easily removed with an additional $O(\log \tau)$ factor.

### 4.1 Recovering the Observable Graph

$\text{Anc}(G)$ encodes all the ancestral relations on observable variables $V$ of the causal graph $G$. To recover $G$ from $\text{Anc}(G)$, we want to differentiate whether $v_i \to v_j$ represents an edge in $G$ or a directed path going through other nodes in $G$. We use the following observation, if $v_i$ is a parent of $v_j$, the path $v_i \to v_j$ is never blocked by any conditioning set $Z \subseteq V \setminus \{v_i\}$. If $v_i \notin \text{Pa}(v_j)$, then we show that we can provide a conditioning set $Z$ in some interventional distribution $S$ such that $v_i \not\perp\!\!\!\perp v_j \mid Z, \text{do}(S)$. For every pair of variables that have an edge in $\text{Anc}(G)$, we design conditioning sets in Algorithm 2 that blocks all the paths between them.

Let $v_i \in \text{Anc}(v_j) \setminus \text{Pa}(v_j)$. We argue that conditioning on $\text{Anc}(v_j) \setminus \{v_i\}$ in $\text{do}(v_i \cup P_{ij})$ blocks all the paths from $v_i$ to $v_j$. The first simple observation, from $d$-separation is that if we take a path that has no $p$-colliders between $v_i$ to $v_j$ (a $p$-collider free path) then it is blocked by conditioning on $\text{Anc}(v_j) \setminus \{v_i\}$ i.e., $v_i \not\perp\!\!\!\perp v_j \mid \text{Anc}(v_j) \setminus \{v_i\}$.

The idea then will be to intervene on colliders $P_{ij}$ to remove these dependencies between $v_i$ and $v_j$ as shown by the following lemma.

**Lemma 4.4.** *Let $v_i \in Anc(v_j)$. $v_i \not\perp\!\!\!\perp v_j \mid \text{do}(v_i \cup P_{ij}), \text{Anc}(v_j) \setminus \{v_i\}$ iff $v_i \notin \text{Pa}(v_j)$.*

From Lemma 4.4, we can recover the edges of the observable graph $G$ provided we know the $p$-colliders between every pair of nodes. However, since the set of $p$-colliders is unknown without the knowledge of $\mathcal{G}$, we construct multiple intervention sets by independently sampling every variable with some probability. This ensures that there exists an intervention set $S$ such that $\{v_i\} \cup P_{ij} \subseteq S$ and $v_j \notin S$ with high probability.

Formally, let $A_t \subseteq V$ for $t \in \{1, 2, \cdots, 72\tau' \log n\}$ be constructed by including every variable $v_i \in V$ with probability $1 - 1/\tau'$ where $\tau' = \max\{\tau, 2\}$. Let $\mathcal{A}_\tau = \{A_1, \cdots, A_{72\tau' \log n}\}$ be the collection of the set $A_t$'s. Algorithm 2 uses the interventions in $\mathcal{A}_\tau$.

**Algorithm 2** RecoverG($\text{Anc}(G), \mathcal{A}_\tau$)

$E = \phi$
**for** $v_i \to v_j$ in $\text{Anc}(G)$ **do**
  Let $\mathcal{A}_{ij} = \{A \mid A \in \mathcal{A}_\tau$ such that $v_i \in A, v_j \notin A\}$
  **if** $\forall\ A \in \mathcal{A}_{ij}, v_i \not\perp\!\!\!\perp v_j \mid \text{Anc}(v_j) \setminus \{v_i\}, \text{do}(A)$ **then**
    $E = E \cup \{(v_i, v_j)\}$
  **end if**
**end for**
**return** $E$

**Proposition 4.5.** *Let $\mathcal{G}(V \cup L, E \cup E_L)$ be a $\tau$-causal graph with observable graph $G(V, E)$. There exists a procedure to recover the observable graph using $O(\tau \log n + \log n)$ many interventions with probability at least $1 - 1/n^2$.*

**Lower Bound.** Complementing the above result, the following proposition gives a lower bound on the number of interventions by providing an instance of a $O(n)$-causal graph such that any non-adaptive algorithm requires $\Omega(n)$ interventions for recovering it. The lower bound comes because of the fact that the algorithm cannot rule out the possibility of latent.

**Proposition 4.6.** *There exists a graph causal $\mathcal{G}(V \cup L, E \cup E_L)$ such that every non-adaptive algorithm requires $\Omega(n)$ many interventions to recover even the observable graph $G(V, E)$ of $\mathcal{G}$.*

## 4.2 Detecting the Latents

We now describe algorithms to identify latents that effect the observable variables $V$ to learn the entire causal graph $\mathcal{G}(V \cup L, E \cup E_L)$. We start from the observable graph $G(V, E)$ constructed in the previous section. Our goal will be to use the fact that $\mathcal{G}$ is a $\tau$-causal graph, which means that $|P_{ij}| \leq \tau$ for every pair $v_i, v_j$. Since we assumed that each latent variable (in $L$) effects at most two observable variables (in $V$), we can split the analysis into two cases: a) pairs of nodes in $G$ without an edge (non-adjacent nodes) and b) pairs of nodes in $G$ with a direct edge (adjacent). In Algorithm LatentsNEdges(Appendix B), we describe the algorithm for identifying the latents effecting pairs of non-adjacent nodes. The idea is to block the paths by conditioning on parents and intervening on $p$-colliders. We use the observation that for any non-adjacent pair $v_i, v_j$ an intervention on the set $P_{ij}$ and conditioning on the parents of $v_i$ and $v_j$ will make $v_i$ and $v_j$ independent, unless there is a latent between them.

**Proposition 4.7.** *Let $\mathcal{G}(V \cup L, E \cup E_L)$ be a $\tau$-causal graph with observable graph $G(V, E)$. Algorithm* LatentsNEdges *with $O(\tau^2 \log n + \log n)$ interventions recovers all latents effecting pairs of non-adjacent nodes in the observable graph $G$ with probability at least $1 - 1/n^2$.*

**Latents Affecting Adjacent Nodes in $G$.** Suppose we have an edge $v_i \to v_j$ in $G(V, E)$ and we want to detect whether there exists a latent $l_{ij}$ that effects both of them. Here, we cannot block the edge path $v_i \to v_j$ by conditioning on any $Z \subseteq V$ in any given interventional distribution $\text{do}(S)$ where $S$ does not contain $v_j$. However, intervening on $v_j$ also disconnects $v_j$ from its latent parent. Therefore, CI-tests are not helpful. Hence, we make use of another test called *do-see* test [Kocaoglu

**Algorithm 3** LatentsWEdges($\mathcal{G}(V \cup L, E \cup E_L), \mathcal{B}_\tau$)

Consider the edge $v_i \to v_j \in E$.
Let $\mathcal{B}_{ij} = \{B \setminus \{v_i\} \mid B \in \mathcal{B}_\tau \text{ s.t. } v_i \in B, v_j \notin B\}$
**if** $\forall B \in \mathcal{B}_{ij}, \Pr[v_j \mid v_i, \text{Pa}(v_j), \text{do}(\text{Pa}(v_i) \cup B)] \neq \Pr[v_j \mid \text{Pa}(v_j), \text{do}(\{v_i\} \cup \text{Pa}(v_i) \cup B)]$ **then**
    $L \leftarrow L \cup l_{ij}, E_L \leftarrow E_L \cup \{(l_{ij}, v_i), (l_{ij}, v_j)\}$
**end if**
**return** $\mathcal{G}(V \cup L, E \cup E_L)$

et al., 2017b], that compares two probability distributions. We assume there exists an oracle that answers whether two distributions are the same or not. This is a well-studied problem with sublinear (in domain size) bound on the sample size needed for implementing this oracle [Chan et al., 2014].

**Assumption 4.8** (Distribution Testing (DT)-Oracle). *Given any $v_i, v_j \in V$ and $Z, S \subseteq V$ tests whether two distributions $\Pr[v_j \mid v_i, Z, \text{do}(S)]$ and $\Pr[v_j \mid Z, \text{do}(S \cup \{v_i\})]$ are identical or not.*

The intuition of the do-see test is as follows: if $v_i$ and $v_j$ are the only two nodes in the graph $G$ with $v_i \to v_j$, then, $\Pr[v_j \mid v_i] = \Pr[v_j \mid \text{do}(v_i)]$ iff there exists no latent that effects both of them. This follows from the *conditional invariance* principle [Bareinboim et al., 2012] (or page 24, property 2 in [Pearl, 2009]). Therefore, the presence or absence of latents can be established by invoking a DT-oracle.

As we seek to minimize the number of interventions, our goal is to create intervention sets that contain $p$-colliders between every pair of variables that share an edge in $G$. However, in Lemmas 4.9, 4.10 we argue that it is not sufficient to consider interventions with only $p$-colliders. We must also intervene on $Pa(v_i)$ to detect a latent between $v_i \to v_j$. The main idea behind LatentsWEdges is captured by the following two lemmas.

**Lemma 4.9** (No Latent Case). *Suppose $v_i \to v_j \in G$ and $v_i, v_j \notin B$, and $P_{ij} \subseteq B$ then, $\Pr[v_j \mid v_i, \text{Pa}(v_j), \text{do}(\text{Pa}(v_i) \cup B)] = \Pr[v_j \mid \text{Pa}(v_j), \text{do}(\{v_i\} \cup \text{Pa}(v_i) \cup B)]$ if there is no latent $l_{ij}$ with $v_i \leftarrow l_{ij} \to v_j$.*

**Lemma 4.10** (Latent Case). *Suppose $v_i \to v_j \in G$ and $v_i, v_j \notin B$, and $P_{ij} \subseteq B$, then, $\Pr[v_j \mid v_i, \text{Pa}(v_j), \text{do}(\text{Pa}(v_i) \cup B)] \neq \Pr[v_j \mid \text{Pa}(v_j), \text{do}(\{v_i\} \cup \text{Pa}(v_i) \cup B)]$ if there is a latent $l_{ij}$ with $v_i \leftarrow l_{ij} \to v_j$.*

From Lemmas 4.9, 4.10, we know that to identify a latent $l_{ij}$ between $v_i \to v_j$, we must intervene on all the $p$-colliders between them with $Pa(v_i) \cup \{v_i\}$. To do this, we again construct random intervention sets. Let $B_t \subseteq V$ for $t \in \{1, 2, \cdots, 72\tau' \log n\}$ be constructed by including every variable $v_i \in V$ with probability $1 - 1/\tau'$ where $\tau' = \max\{\tau, 2\}$. Let $\mathcal{B}_\tau = \{B_1, \cdots, B_{72\tau' \log n}\}$ be the collection of the sets. Consider a pair $v_i \to v_j$. To obtain the interventions given by the above lemmas, we iterate over all sets in $\mathcal{B}_\tau$ and identify all the sets containing $v_i$, but not $v_j$. From these sets, we remove $v_i$ to obtain $\mathcal{B}_{ij}$. These new interventions are then used in LatentsWEdges to perform the required distribution tests using a DT-oracle on the interventions $B \cup Pa(v_i)$ and $B \cup Pa(v_i) \cup \{v_i\}$ for every $B \in \mathcal{B}_{ij}$. We can show:

**Proposition 4.11.** *Let $\mathcal{G}(V \cup L, E \cup E_L)$ be a $\tau$-causal graph with observable graph $G(V, E)$. LatentsWEdges with $O(n\tau \log n + n \log n)$ interventions recovers all latents effecting pairs of adjacent nodes in the observable graph $G$ with probability at least $1 - 1/n^2$.*

**Putting it all Together.** Using Propositions 4.5, 4.7, and 4.11, we get the following result. Note that $\tau \leq n - 2$.

**Theorem 4.12.** *Given access to a $\tau$-causal graph $\mathcal{G} = \mathcal{G}(V \cup L, E \cup E_L)$ through Conditional Independence (CI) and Distribution Testing (DT) oracles, Algorithms* RecoverG, LatentsNEdges, *and* LatentsWEdges*put together recovers $\mathcal{G}$ with $O(n\tau \log n + n \log n)$ interventions, with probability at least $1 - O(1/n^2)$ (where $|V| = n$).*

# 5 Experiments

In this section, we compare the total number of interventions required to recover causal graph $\mathcal{G}$ parameterized by $p$-colliders (See section 4) vs. maximum degree utilized by [Kocaoglu et al., 2017b].

**Setup**. We demonstrate our results by considering sparse random graphs generated from the families of: (i) Erdös-Rényi random graphs $G(n, c/n)$ for constant $c$, (ii) Random Bipartite Graphs generated using $G(n_1, n_2, c/n)$ model, with partitions $L$, $R$ and edges directed from $L$ to $R$, (iii) Directed Trees with degrees of nodes generated from power law distribution. In each of the graphs we generate, we additionally include latent variables by sampling 5% of $\binom{n}{2}$ pairs and adding a latent between them.

**Finding $p$-colliders.** Let $\mathcal{G}$ contain observable variables and the latents. To find $p$-colliders between every pair of observable nodes of $\mathcal{G}$, we enumerate all paths between them and check if any of the observable nodes on a path can be a possible $p$-collider. As this became practically infeasible for larger values of $n$, we devise an algorithm that runs in polynomial time (in the size of the graph) by constructing an appropriate flow network and finding maximum flow in this network. We will first describe a construction that takes three nodes $(v_i, v_j, v_k)$ as input and checks if $v_k$ is a $p$-collider for the pair of nodes $v_i$ and $v_j$. Iterating over all possible nodes $v_k$ gives us all the $p$-colliders for the pair $v_i, v_j$.

Construction. If $v_k$ is not an ancestor of either $v_i$ or $v_j$, then, output $v_k$ is not a $p$-collider. Else, we describe a modification of $\mathcal{G}$ to obtain the flow network $\widetilde{\mathcal{G}}$. First, initialize $\widetilde{\mathcal{G}}$ with $\mathcal{G}$. Remove all outgoing edges of $v_k$ from $\widetilde{\mathcal{G}}$ and set the capacity of all incoming edges incident on $v_k$ to 1. Add a node $T_{ij}$ along with the edges $T_{ij} \to v_i$ and $T_{ij} \to v_j$ to $\widetilde{\mathcal{G}}$ and set the capacity of these edges to 1. For every node $w \in V \cup L \setminus \{v_k\}$, create two nodes $w_{in}$ and $w_{out}$. Add edge $w_{out} \to w_{in}$ with a capacity 1. Every incoming edge to $w$ i.e., $z \to w$ is replaced by $z \to w_{in}$ and every outgoing edge $w \to z$ is replaced by $w_{out} \to z$ with capacity 1. Find maximum $s, t$ flow in $\widetilde{\mathcal{G}}$ with $T_{ij}, v_k$ as source and sink respectively. If the maximum flow is 2, then output $v_k$ is a $p$-collider, otherwise no.
Now, we outline the idea for the proof of correctness of the above construction.

Sketch of the Proof. After ensuring that $v_k$ has a directed path to either $v_i$ or $v_j$, we want to check whether there is an undirected path from $v_i$ to $v_j$ containing $v_k$ as a collider. In other words, we want to check if there are *two vertex disjoint paths* from $v_i$ and $v_j$ to $v_k$ such that both of these paths have incoming edges to $v_k$. By adding a node $T_{ij}$ connected to $v_i$ and $v_j$, we want to route two units of flow from $T_{ij}$ to $v_k$ where each node has a vertex capacity of 1. Converting vertex capacities into edge capacities by splitting every node into two nodes (one for incoming and the other for outgoing edges) gives us the desired flow network on which we can solve maximum flow.

**Results.** In our plots (Figure 2), we compare the maximum undirected degree ($d$) with the maximum number of $p$-colliders between any pair of nodes (which defines $\tau$). We ran each experiment 10 times and plot the mean value along with one standard deviation error bars.

Recall that in the worst case, the number of interventions used by our approach (Theorem 4.12) is $O(n\tau \log n + n \log n)$ while the algorithm proposed by [Kocaoglu et al., 2017b] uses $O(\min\{d \log^2 n, \ell\} +$

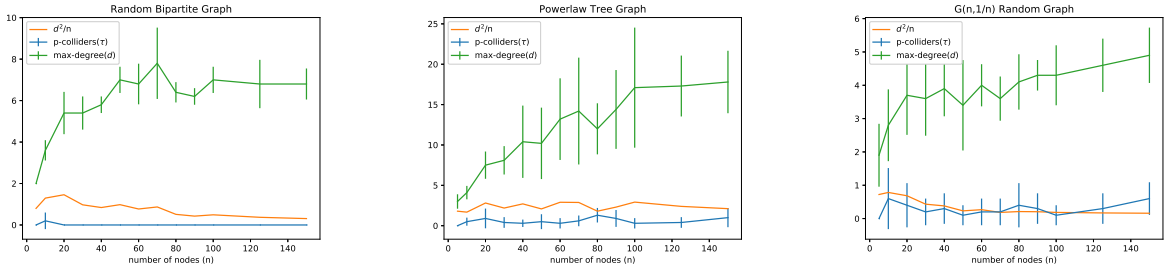


Figure 2: Comparison of $\tau$ vs. maximum degree in various sparse random graph models. On the x-axis is the number of nodes in the graph. Note that our bound on the number of interventions needed to recover $\mathcal{G}$ is better than those provided by [Kocaoglu et al., 2017b] roughly when $\tau < d^2/n$.

$d^2 \log n$) many interventions where $\ell$ is the length of the longest directed path in the graph. So roughly when $\tau < d^2/n$, our bound is better. For this purpose, we also plot the $d^2/n$ line using the mean value of $d$ obtained.

For random bipartite graphs, that can be used to model causal relations over time, we use equal partition sizes $n_1 = n_2 = n/2$ and plot the results for $\mathcal{G}(n/2, n/2, c/n)$ for constant $c = 5$. We observe that the behaviour is uniform for small constant values of $c$. In this case, we observe that the number of $p$-colliders is close to zero for all values of $n$ in the range considered and our bound is better.

For directed random trees where degrees of nodes follow the powerlaw distribution (observed in real world networks [Adamic and Huberman, 2000]), we again observe that for almost all the values of $n$, our bound is better. We run our experiments with small constant values for the exponent $\gamma$ and show the plots for $\gamma = 3$ in Figure 2.

Powerlaw graphs contain only a few nodes concentrated around a very high degree. Therefore, we expect our algorithm to perform better in such cases.

Also for Erdös-Rényi random graphs $\mathcal{G}(n, 1/n)$, we observe that our bound is either better or comparable to that of [Kocaoglu et al., 2017b].

It is interesting to see that in the sparse graphs we considered $\tau$ is considerably smaller compared to $d$. Moreover, if we want to identify only the observable graph $G$ under the presence of latents, our algorithm uses $O(\tau \log n)$ interventions where as the previous known algorithm [Kocaoglu et al., 2017b] uses $O(d \log^2 n)$ interventions. In the random graphs considered above, our algorithms perform significantly better for identifying $G$. Therefore, we believe that minimizing the number of interventions based on the notion of $p$-colliders is a reasonable direction to consider.

## 6 Conclusion

We have studied how to recover a causal graph in presence of latents while minimizing intervention cost. In the linear cost setting, we give a 2-approximation algorithm for ancestral graph recovery. This approximation factor can be improved to $(1+\epsilon)$ under some additional assumptions. Removing these assumptions would be an interesting direction for future work. In the identity cost setting, we give a randomized algorithm to recover the full causal graph, through a novel characterization based on $p$-colliders. In this setting, understanding the optimal intervention cost is open, and an important direction for research.

While we focus on non-adaptive settings, where all the interventions are constructed at once in the beginning, an adaptive (sequential) setting has received recent attention [He and Geng, 2008; Shanmugam et al., 2015], and is an interesting direction in both our cost models.

## Acknowledgements

The first two authors would like to thank Nina Mishra, Yonatan Naamad, MohammadTaghi Hajiaghayi and Dominik Janzing for many helpful discussions.

## References


Lada A Adamic and Bernardo A Huberman. Power-law distribution of the world wide web. *Science*, 287(5461):2115–2115, 2000.

Elias Bareinboim and Judea Pearl. Causal inference and the data-fusion problem. *Proceedings of the National Academy of Sciences*, 113(27):7345–7352, 2016.

Elias Bareinboim, Carlos Brito, and Judea Pearl. Local characterizations of causal bayesian networks. In *Graph Structures for Knowledge Representation and Reasoning*, pages 1–17. Springer, 2012.

Clément L Canonne, Ilias Diakonikolas, Daniel M Kane, and Alistair Stewart. Testing conditional independence of discrete distributions. In *2018 Information Theory and Applications Workshop (ITA)*, pages 1–57. IEEE, 2018.

Siu-On Chan, Ilias Diakonikolas, Paul Valiant, and Gregory Valiant. Optimal algorithms for testing closeness of discrete distributions. In *Proceedings of the twenty-fifth annual ACM-SIAM symposium on Discrete algorithms*, pages 1193–1203. SIAM, 2014.

Frederick Eberhardt. Causation and intervention. *PhD Thesis, Carnegie Mellon University*, 2007.

Frederick Eberhardt and Richard Scheines. Interventions and causal inference. *Philosophy of Science*, 74(5):981–995, 2007.

Alain Hauser and Peter Bühlmann. Characterization and greedy learning of interventional markov equivalence classes of directed acyclic graphs. *Journal of Machine Learning Research*, 13(Aug): 2409–2464, 2012.

Alain Hauser and Peter Bühlmann. Two optimal strategies for active learning of causal models from interventional data. *International Journal of Approximate Reasoning*, 55(4):926–939, 2014.

Yang-Bo He and Zhi Geng. Active learning of causal networks with intervention experiments and optimal designs. *Journal of Machine Learning Research*, 9(Nov):2523–2547, 2008.

Christina Heinze-Deml, Marloes H Maathuis, and Nicolai Meinshausen. Causal structure learning. *Annual Review of Statistics and Its Application*, 5:371–391, 2018.

Patrik O Hoyer, Dominik Janzing, Joris M Mooij, Jonas Peters, and Bernhard Schölkopf. Nonlinear causal discovery with additive noise models. In *Advances in neural information processing systems*, pages 689–696, 2009.

Antti Hyttinen, Frederick Eberhardt, and Patrik O Hoyer. Experiment selection for causal discovery. *The Journal of Machine Learning Research*, 14(1):3041–3071, 2013a.

Antti Hyttinen, Patrik O Hoyer, Frederick Eberhardt, and Matti Jarvisalo. Discovering cyclic causal models with latent variables: A general sat-based procedure. *arXiv preprint arXiv:1309.6836*, 2013b.

Stasys Jukna. *Extremal combinatorics: with applications in computer science*. Springer Science & Business Media, 2011.

Gyula Katona. On separating systems of a finite set. *Journal of Combinatorial Theory*, 1(2): 174–194, 1966.

Ákos Kisvölcsey. Flattening antichains. *Combinatorica*, 1(26):65–82, 2006.

Murat Kocaoglu, Alex Dimakis, and Sriram Vishwanath. Cost-optimal learning of causal graphs. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 1875–1884. JMLR. org, 2017a.

Murat Kocaoglu, Karthikeyan Shanmugam, and Elias Bareinboim. Experimental design for learning causal graphs with latent variables. In *Advances in Neural Information Processing Systems*, pages 7018–7028, 2017b.

Joseph B Kruskal. The number of simplices in a complex. *Mathematical Optimization Techniques*, 10:251–278, 1963.

Erik Lindgren, Murat Kocaoglu, Alexandros G Dimakis, and Sriram Vishwanath. Experimental design for cost-aware learning of causal graphs. In *Advances in Neural Information Processing Systems*, pages 5279–5289, 2018.

Po-Ling Loh and Peter Bühlmann. High-dimensional learning of linear causal networks via inverse covariance estimation. *The Journal of Machine Learning Research*, 15(1):3065–3105, 2014.

Florence Jessie MacWilliams and Neil James Alexander Sloane. *The theory of error-correcting codes*, volume 16. Elsevier, 1977.

Cai Mao-Cheng. On separating systems of graphs. *Discrete Mathematics*, 49(1):15–20, 1984.

Pekka Parviainen and Mikko Koivisto. Ancestor relations in the presence of unobserved variables. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 581–596. Springer, 2011.

Judea Pearl. *Causality: models, reasoning and inference*, volume 29. Springer, 2000.

Judea Pearl. *Causality: Models, Reasoning, and Inference*. Cambridge university press, 2009.

Thomas Richardson, Peter Spirtes, et al. Ancestral graph markov models. *The Annals of Statistics*, 30(4):962–1030, 2002.

Karthikeyan Shanmugam, Murat Kocaoglu, Alexandros G Dimakis, and Sriram Vishwanath. Learning causal graphs with small interventions. In *Advances in Neural Information Processing Systems*, pages 3195–3203, 2015.

Shohei Shimizu, Patrik O Hoyer, Aapo Hyvärinen, and Antti Kerminen. A linear non-gaussian acyclic model for causal discovery. *Journal of Machine Learning Research*, 7(Oct):2003–2030, 2006.

Ricardo Silva, Richard Scheine, Clark Glymour, and Peter Spirtes. Learning the structure of linear latent variable models. *Journal of Machine Learning Research*, 7(Feb):191–246, 2006.

Peter Spirtes, Clark N Glymour, Richard Scheines, David Heckerman, Christopher Meek, Gregory Cooper, and Thomas Richardson. *Causation, prediction, and search*. MIT press, 2000.

Thomas Verma and Judea Pearl. An algorithm for deciding if a set of observed independencies has a causal explanation. In *Uncertainty in artificial intelligence*, pages 323–330. Elsevier, 1992.

Kun Zhang, Jonas Peters, Dominik Janzing, and Bernhard Schölkopf. Kernel-based conditional independence test and application in causal discovery. In *Proceedings of the Twenty-Seventh Conference on Uncertainty in Artificial Intelligence*, pages 804–813. AUAI Press, 2011.

# Appendix

## A Missing Details from Section 3

**Lemma A.1** (Lemma 3.1 Restated)**.** *Suppose $\mathcal{S} = \{S_1, S_2, \cdots, S_m\}$ is a collection of subsets of $V$. For a given causal graph $G$ if $\text{Anc}(G)$ is recovered using CI-tests by intervening on the sets $S_i \in \mathcal{S}$. Then, $\mathcal{S}$ is a strongly separating set system.*

*Proof.* Suppose $\mathcal{S}$ is not a strongly separating set system. If there exists a pair of nodes $(v_i, v_j)$ such that every set $S_k \in \mathcal{S}$ contains none of them, then, we cannot recover the edge between these two nodes as we are not intervening on either $v_i$ or $v_j$ and the results of an independence test $v_i \perp\!\!\!\perp v_j$ might not be correct due to the presence of a latent variable $l_{ij}$ between them. Now, consider the case when only one of them is present in the set system. Let $(v_i, v_j)$ be such that $\forall S_k : S_k \cap \{v_i, v_j\} = \{v_i\} \Rightarrow v_i \in S_k, v_j \notin S_k$. We choose our graph $G_{ij}$ to have two components $\{v_i, v_j\}$ and $V \setminus \{v_i, v_j\}$; and include the edge $v_j \to v_i$ in it. Our algorithm will conclude from CI-test $v_i \perp\!\!\!\perp v_j \mid \text{do}(S_k)$ that $v_i$ and $v_j$ are independent. However, it is possible that $v_i \not\perp\!\!\!\perp v_j$ because of a latent $l_{ij}$ between $v_i$ and $v_j$, but $v_i \perp\!\!\!\perp v_j \mid \text{do}(S_k)$ as intervening on $v_i$ disconnects the $l_{ij} \to v_i$ edge. Therefore, our algorithm cannot distinguish the two cases $v_j \to v_i$ and $v_i \leftarrow l_{ij} \to v_j$ without intervening on $v_j$. For every $\mathcal{S}$ that is not a strongly separating set system, we can provide a $G_{ij}$ such that by intervening on sets in $\mathcal{S}$, we cannot recover $\text{Anc}(G_{ij})$ correctly. □

### A.1 Missing Details from Section 3.1

We first argue that the matrix returned by Algorithm SSMatrix is indeed a strongly separating matrix.

**Lemma A.2.** *The matrix $U$ returned by Algorithm* SSMatrix *is a strongly separating matrix.*

*Proof.* Consider any two nodes $v_i$, $v_j$ with corresponding row vectors $U(i)$ and $U(j)$. Suppose $\|U(i)\|_1 = \|U(j)\|_1$.

By construction, $U(i) \neq U(j)$ they will differ in at least one coordinate. However, they have equal weights, so, there must exist one more coordinate such that the strongly separating condition holds. If $U(i)$ and $U(j)$ have weights $r^i \neq r^j$, then, $U(i, m' - \log n + r^i) = U(j, m' - \log n + r^j) = 1$ and $U(i, m' - \log n + r^j) = U(j, m' - \log n + r^i) = 0$ by construction outlined in the Algorithm SSMatrix. This proves that the matrix $U$ returned is a strongly separating matrix. □

The following inequalities about Algorithm SSMatrix will be useful in analyzing its performance.

**Lemma A.3.** *. For $m \geq 66 \log n$ and $m'$ as defined in Algorithm* SSMatrix*, we have the following*

*(a)* $\sum_{t=1}^{\log n} \binom{m' - \log n}{t} \geq n$ *(i.e., there are enough vectors of weight $\leq \log n$ only using $m' - \log n$ columns to assign a unique vector to each variable).*

*(b) Let $i^*$ be the smallest integer s.t. $\sum_{t=1}^{i^*} \binom{m'}{t} \geq n$. Then, $\sum_{t=1}^{2i-1} \binom{m' - \log n}{t} \geq \sum_{t=1}^{i} \binom{m'}{t}$ for all $i \in \{2, \ldots, i^*\}$.*

*Proof.* Let $m \geq 66 \log n$. From Algorithm SSMatrix, we have $m' = m - a_1$ for all guesses $1 \leq a_1 \leq \frac{2m}{3}$. By reserving the last "$\log n$" columns in Algorithm SSMatrix, we want to make sure that $m' - \log n$ can fully cover $n$ nodes with weight at most $\log n$. We have :

$$m' = m - a_1 \geq \frac{m}{3} \geq 22 \log n \text{ and}$$
$$\sum_{t=1}^{\log n} \binom{m' - \log n}{t} \geq \binom{m' - \log n}{\log n} \geq \left(\frac{21 \log n}{\log n}\right)^{\log n} > n.$$

Moving onto Part (b). Let $i^*$ be the minimum value of $i$ such that $\sum_{t=1}^{i^*} \binom{m'}{t} \geq n$. Consider $i$ such that $2 \leq i \leq i^*$:

$$\sum_{t=1}^{2i-1} \binom{m' - \log n}{t} \geq \binom{m' - \log n}{2i-1}.$$

Consider now the right hand side:

$$\sum_{t=1}^{i} \binom{m'}{t} \leq \sum_{t=0}^{i} \binom{m'}{t} \leq \sum_{t=0}^{i} \frac{m'^t}{t!} \leq \sum_{t=0}^{i} \frac{i^t}{t!} \left(\frac{m'}{i}\right)^t \leq \mathrm{e}^i \left(\frac{m'}{i}\right)^i.$$

We inductively show that for all $i \geq 2$

$$\frac{\left(\frac{em'}{i}\right)^i}{\binom{m' - \log n}{2i-1}} \leq 1.$$

Let $i = 2$. For $m \geq \frac{m'}{3} \geq 50 \left(\frac{22}{21}\right)^3$ we have

$$\frac{(em'/2)^2}{\binom{m' - \log n}{3}} \leq \frac{(em'/2)^2}{(\frac{m' - \log n}{3})^3} \leq 50 \left(\frac{22}{21}\right)^3 \frac{m'^2}{m'^3} \leq 1.$$

Assume the inequality is correct for some $i > 2$. Now, we show that it must also hold for $i + 1$.

$$\frac{\left(\frac{em'}{i+1}\right)^{i+1}}{\binom{m' - \log n}{2i+1}} = \frac{\left(\frac{em'}{i+1}\right)^i \frac{em'}{i+1} \binom{m' - \log n}{2i-1}}{\binom{m' - \log n}{2i-1}\binom{m' - \log n}{2i+1}} \leq \frac{\left(\frac{em'}{i}\right)^i \frac{em'}{i+1} \binom{m' - \log n}{2i-1}}{\binom{m' - \log n}{2i-1}\binom{m' - \log n}{2i+1}} \leq \frac{\frac{em'}{i+1} \binom{m' - \log n}{2i-1}}{\binom{m' - \log n}{2i+1}}.$$

For ease of notation, denote $a = m' - \log n \geq m'(1 - \frac{1}{22}) \geq 21 \log n$.

Consider the binary entropy function $H(x) = -x \log x - (1 - x) \log(1 - x)$. For $x \in [\frac{2i-1}{a}, \frac{2i+1}{a}]$, $H(x)$ is an increasing function. For some value of $x$ in the range we have :

$$\frac{H(\frac{2i+1}{a}) - H(\frac{2i-1}{a})}{\frac{2i+1}{a} - \frac{2i-1}{a}} = H'(x) = \log\left(\frac{1}{x} - 1\right) \geq \log\left(\frac{a}{2i-1} - 1\right)$$
$$\implies H\left(\frac{2i+1}{a}\right) - H\left(\frac{2i-1}{a}\right) \geq \frac{2}{a} \log\left(\frac{a}{2i-1} - 1\right).$$

Now, consider the fraction

$$\binom{a}{2i-1} / \binom{a}{2i+1}.$$

Using the bound from (MacWilliams and Sloane [1977], Page 309)

$$\sqrt{\frac{a}{8b(a-b)}}2^{mH(b/a)} \leq \binom{a}{b} \leq \sqrt{\frac{a}{2\pi b(a-b)}}2^{mH(b/a)},$$

$$\begin{aligned}
\binom{a}{2i-1}/\binom{a}{2i+1} &\leq \sqrt{8(2i+1)(a-2i-1)/2\pi(2i-1)(a-2i+1)}/2^{aH(\frac{2i+1}{a})-H(\frac{2i-1}{a})} \\
&\leq \sqrt{20/3\pi}/2^{aH(\frac{2i+1}{a})-H(\frac{2i-1}{a})} \\
&\leq \sqrt{20/3\pi}/2^{2\log\left(\frac{a}{2i-1}-1\right)} \\
&= \frac{\sqrt{20/3\pi}}{\left(\frac{a}{2i-1}-1\right)^2}.
\end{aligned}$$

Combining the above, we have :

$$\begin{aligned}
\frac{\left(\frac{em'}{i+1}\right)^{i+1}}{\binom{m'-\log n}{2i+1}} &\leq \frac{\frac{m'}{i+1}\sqrt{20e^2/3\pi}}{\left(\frac{a}{2i-1}-1\right)^2} \\
&\leq \frac{4m'i\sqrt{20e^2/3\pi}}{(a-2i)^2} \\
&\leq \frac{4m'\log n\sqrt{20e^2/3\pi}}{m'^2\left(1-3/22\right)^2} = \frac{4\log n\sqrt{20e^2/3\pi}}{m'\left(1-3/22\right)^2} \leq \frac{21.2\log n}{m'} \leq 1.
\end{aligned}$$

Therefore, we have for all $i \geq 2$

$$\left(\frac{em'}{i}\right)^i/\binom{m'-\log n}{2i-1} \leq 1$$

$$\implies \sum_{t=1}^{i}\binom{m'}{t} \leq \left(\frac{em'}{i}\right)^i \leq \binom{m'-\log n}{2i-1} \leq \sum_{t=1}^{2i-1}\binom{m'-\log n}{t}.$$

□

Let $c_U = \sum_{j=1}^{n} c(v_j)\|U(j)\|_1$ be value of objective for the matrix $U$ returned by Algorithm SSMatrix.

Consider $U_{\text{OPT}}$, and let $V^{(1)}_{\text{OPT}}$ represent all nodes that are assigned weight 1 in it (nodes which have only one 1 in their row). Let $c^{(1)}_{\text{OPT}}$ denote the sum of cost of the nodes in $V^{(1)}_{\text{OPT}}$. In our Algorithm SSMatrix, we maintain a guess for the size of $V^{(1)}_{\text{OPT}}$ as $a_1$. We want to guess the exact value of $|V^{(1)}_{OPT}| \leq m$. However, we only guess $a_1$ until $\frac{2m}{3}$, so that the remaining columns can be used to obtain a valid separating matrix (for each of our guesses) as observed in Lemma A.3. We show that the cost contribution of nodes in $V^{(1)}_{\text{OPT}}$ (by allowing this slack in our guesses) due to Algorithm SSMatrixis not far away from $c^{(1)}_{\text{OPT}}$.

First, we show that for any weight $i \geq 2$ node in $U_{\text{OPT}}$, the output $U$ of Algorithm SSMatrix assigns vectors with weight at most $2i$ and for a weight 1 node, we show that the weight assigned by $U$ is at most 3.

**Lemma A.4.** *Algorithm* SSMatrix *assigns a weight of*

*(a) at most* 3 *for a weight* 1 *node in* $U_{\text{OPT}}$.

*(b) at most* $2i$ *for a node of weight* $i$ *in* $U_{\text{OPT}}$ *for* $i \geq 2$.

*Proof.* (*a*) Let $V$ denote sorted (in the decreasing order of cost) order of nodes. Suppose we assign unique length-$m$ vectors starting from weight 1 to the nodes in the order $V$. Let the assignment of vectors be denoted by $\widetilde{U}$. It is easy to observe that this described assignment $\widetilde{U}$ is not a strongly separating matrix. However, any strongly separating matrix $U$ is such that the vector assigned to any node $v_i$ in $U$ has weight at least that in $\widetilde{U}$ i.e., $\|U(i)\|_1 \geq \|\widetilde{U}(i)\|_1$. As $U$ can be any strongly separating matrix, it also holds for $U_{\text{OPT}}$ giving us $\|U_{\text{OPT}}(i)\|_1 \geq \|\widetilde{U}(i)\|_1$.

The number of weight 1 nodes possible in the assignment $\widetilde{U}$ is $\binom{m}{1}$ and therefore, $|V^{(1)}_{\text{OPT}}| \leq m$. Consider all the nodes of weight $\leq 3$ in $U$ assigned by Algorithm SSMatrix. After discarding the first $m' = m - a_1$ columns assuming our guess $a_1$ in the current iteration, $U$ starts assigning vectors with weight 1 in the remaining $m' - \log n$ while setting a 'row weight indicator bit' in the last $\log n$ columns. In order to obtain nodes of weight $\leq 3$, in $U$, we include vectors of weight $\leq 2$ in the $m' - \log n$ columns. Therefore, total number of such nodes is $a_1 + \binom{m'-\log n}{1} + \binom{m'-\log n}{2}$.

$$
\begin{aligned}
a_1 + \binom{m' - \log n}{1} + \binom{m' - \log n}{2} &\geq \binom{m' - \log n}{1} + \binom{m' - \log n}{2} \\
&\geq m' - \log n + \left(\frac{m' - \log n}{2}\right)^2 \qquad \left(\text{using} \binom{m'}{k} \geq \left(\frac{m'}{k}\right)^k\right) \\
&\geq m \geq |V^{(1)}_{\text{OPT}}|. \qquad \left(\text{using } m' \geq \frac{m}{3} \text{ and } m \geq 66 \log n\right)
\end{aligned}
$$

Therefore, every weight 1 node in $U_{\text{OPT}}$ is covered by a vector in $U$ with weight $\leq 3$.

(*b*) First we argue that using an appropriate $m'$, we can give a construction of $\widetilde{U} \in \{0,1\}^{n \times m'}$ (similar to case (a)) such that weight of node $v_j$ in $\widetilde{U}$ is at most the weight in $U_{\text{OPT}}$ for all nodes of weight more than 2 in $U_{\text{OPT}}$. Let $m' = m - \frac{2m}{3}$. In other words, we are considering the guess $a_1 = \frac{2m}{3}$. As our algorithm $U$ considers all the guesses and returns $U$ with the lowest cost, arguing that our lemma holds for this guess is sufficient. For this value of $m'$, let $\widetilde{U}$ be constructed using vectors from $\{0,1\}^{m'}$ in the increasing order of weight, starting with weight 1.

When $m' = \frac{m}{3}$, it is possible that a node in $U_{\text{OPT}}$ can be assigned a vector of weight 1 from $\{0,1\}^{m'}$ (this can happen when $|V^{(1)}_{\text{OPT}}| \geq \frac{2m}{3}$). As $\widetilde{U}$ assigns weights in the increasing order occupying the entire $m'$ columns, it will not result in a strongly separating matrix. Therefore, any node $v_j$ with weight $i \geq 2$ in $U_{\text{OPT}}$, will be assigned a weight of at most $i$ in $\widetilde{U}$.

We know that the number of vectors of weight at most $i$ in $\widetilde{U}$ is equal to $\sum_{t=1}^{i} \binom{m'}{t}$ and number of vectors with weight at most $2i-1$ using $m' - \log n$ columns of $U$ is equal to $\sum_{t=1}^{2i-1} \binom{m'-\log n}{t}$. As Lemma A.3 holds for all guesses of $a_1$, we have $\sum_{t=1}^{i} \binom{m'}{t} \leq \sum_{t=1}^{2i-1} \binom{m'-\log n}{t}$ for all $i \geq 2$. Using induction, we can observe that $v_j$ is assigned a vector in $\{0,1\}^{m'-\log n}$ with weight at most $2i-1$. As $U$ obtained from Algorithm SSMatrix mimics the construction used in $\widetilde{U}$ over $m' - \log n$ columns, we have that weight of node $v_j$ in $U$ using $m' - \log n$ columns is at most $2i-1$. Combining it with the 'row weight indicator' bit we set to 1 in the last $\log n$ columns gives us the lemma. □

In our next lemma shows that the sum of contribution of the nodes in $V^{(1)}_{\text{OPT}}$ to $c_U$ is at most twice that of $c^{(1)}_{\text{OPT}}$. Combining this with Lemma A.4, we show that $U$ achieves a 2-approximation.

**Lemma A.5.** *Let* $c^{(1)}_U = \sum_{v_i \in V^{(1)}_{\text{OPT}}} c(v_i)\|U(i)\|_1$ *for the matrix* $U$ *returned by Algorithm* SSMatrix, *then* $c^{(1)}_U \leq 2c^{(1)}_{OPT}$.

*Proof.* Suppose $a_1$ represents our guess for the number of weight 1 vectors and $a_1^*$ represent the number of weight 1 vectors in $U_{\text{OPT}}$ i.e, $|V_{\text{OPT}}^{(1)}| = a_1^*$. In Algorithm SSMatrix, we use the following bounds for our guess $0 \leq a_1 \leq 2m/3$. If $a_1^* \leq \frac{2m}{3}$, then it would have been one of our guesses. As we take minimum among all the guesses, we have $c_U^{(1)} = c_{OPT}^{(1)}$ in such a case.

Consider the case when $a_1^* > \frac{2m}{3}$. Let $V_{\text{OPT}}^{(1)} = \{v_1, v_2, \cdots v_{a_1^*}\}$ represent an ordering of nodes in the decreasing ordering of cost that are assigned weight 1 in $U_{\text{OPT}}$. Consider the contribution of only weight 1 nodes to $c_{\text{OPT}}$. We have

$$c_{\text{OPT}}^{(1)} = \sum_{k=1}^{a_1^*} c(v_k) \geq \sum_{k=1}^{2m/3} \frac{2m}{3} c(v_k) \geq \frac{2m}{3} c(v_{2m/3}).$$

We will look at the case when our guess $a_1$ reaches $a_1 = \frac{2m}{3}$ and argue about the cost for this particular value of $a_1$. As we are taking minimum over all the guesses, we are only going to do better and our approximation ratio will only be better. Among the nodes $\{v_1, v_2, \cdots v_{a_1^*}\}$ first $\frac{2m}{3}$ nodes would be assigned weight 1 by $U$. From Lemma A.4, we have that for the remaining $a_1^* - \frac{2m}{3}$ nodes, Algorithm SSMatrix might assign a weight 2 or weight 3 vector in $U$.

$$\begin{aligned}
c_U^{(1)} &\leq \sum_{i=1}^{2m/3} c(v_i) + 3 \sum_{j=2m/3+1}^{a_1^*} c(v_j) = \sum_{i=1}^{a_1^*} c(v_i) + 2 \sum_{j=2m/3+1}^{a_1^*} c(v_j) \\
&\leq \sum_{i=1}^{a_1^*} c(v_i) + 2\left(a_1^* - \frac{2m}{3}\right) c(v_{2m/3+1}) \\
&\leq c_{OPT}^{(1)} + 2\left(a_1^* - \frac{2m}{3}\right) c(v_{2m/3}) && \text{(since } c(v_{2m/3}) \geq c(v_{2m/3+1}) \text{ )} \\
&\leq c_{OPT}^{(1)} + \frac{2m}{3} c(v_{2m/3}) && \text{(since } a_1^* \leq m \text{ )} \\
&\leq 2c_{OPT}^{(1)}.
\end{aligned}$$

This completes the proof of the lemma. □

**Theorem A.6** (Theorem 3.3 Restated)**.** *Let $m \geq 66 \log n$ and $U$ be the strongly separating matrix returned by Algorithm* SSMatrix*. Let $c_U = \sum_{j=1}^{n} c(v_j) \|U(j)\|_1$. Then,*

$$c_U \leq 2 \cdot c_{\text{OPT}},$$

*where $c_{\text{OPT}}$ is the objective value associated with optimum set of interventions corresponding to $U_{\text{OPT}}$.*

*Proof.* From Lemma A.2, we know that matrix returned by Algorithm SSMatrix given by $U$ with cost $c_U$ is a strongly separating matrix. Consider a strongly separating matrix $U_{\text{OPT}}$ that achieves optimum objective value $c_{\text{OPT}}$. Let $V_{\text{OPT}}^{(1)}$ represent all nodes that are assigned weight 1 in $U_{\text{OPT}}$. Let $c_U^{(1)}$ denote the cost of nodes in $V_{\text{OPT}}^{(1)}$ using $U$ returned by Algorithm SSMatrix and $c_{\text{OPT}}^{(1)}$

represents that of $U_{\text{OPT}}$. We have $c_{\text{OPT}} = c^{(1)}_{OPT} + \sum_{j:\|U_{\text{OPT}}(j)\|_1 \geq 2} c(v_j)\, \|U_{\text{OPT}(j)}\|_1$.

$$\begin{aligned}
c_U &= c^{(1)}_U + \sum_{j:\|U_{\text{OPT}}(j)\|_1 \geq 2} c(v_j)\, \|U(j)\|_1 \\
&\leq c^{(1)}_U + \sum_{j:\|U_{\text{OPT}}(j)\|_1 \geq 2} c(v_j)\, 2\|U_{\text{OPT}}(j)\|_1 && \text{(from Lemma A.4)} \\
&\leq 2c^{(1)}_{OPT} + 2 \sum_{j:\|U_{\text{OPT}}(j)\|_1 \geq 2} c(v_j)\, \|U_{\text{OPT}}(j)\|_1 && \text{(from Lemma A.5)} \\
&\leq 2c_{\text{OPT}}.
\end{aligned}$$

This completes the proof of the theorem. □

## A.2 Missing Details from Section 3.2

In this section, we present our algorithm that achieves an improved $1+\epsilon$-approximation in the linear cost model setting under mild assumptions on the cost of the nodes and the number of interventions. The algorithm is adapted from that proposed by Hyttinen et al. [2013a] whose work drew connections between causality and known separating system constructions in combinatorics. In particular, [Hyttinen et al., 2013a] considered a setting where given $n$ variables and $m$, the goal is to construct $k$ sets that are strongly separating with the objective of minimizing the average size of the intervention sets. Stated differently this provides an algorithm for solving 1 when $c(v) = 1$ for all nodes $v \in V$.

In Section A.4, we adapt the algorithm from [Hyttinen et al., 2013a] to deal with the case where each node could have a different cost value. Our main contribution is to show that this adaptation constructs a set of interventions which achieves an objective value in the linear cost model that is within a factor $1+\epsilon$ times of the optimum under some mild restrictions. In Section A.3, we start with some definitions and statements from the combinatorics that will prove useful for stating and analyzing the algorithm.

## A.3 Combinatorics Preliminaries

**Definition A.7.** *(antichain). Consider a collection $\mathcal{S}$ of subsets of $\{v_1, v_2, \cdots v_n\}$ such that for any two sets $S_i, S_j \in \mathcal{S}$, we have $S_i \not\subset S_j$ and $S_j \not\subset S_i$. Then, such a collection $\mathcal{S}$ is called an antichain.*

We provide a lemma that shows that an antichain can also be represented as a strongly separating matrix.

**Lemma A.8.** *Let $\mathcal{T} = \{T_1, T_2, \cdots T_n\}$ be an antichain defined on $\{1, 2, \cdots m\}$. Construct a matrix $U \in \{0,1\}^{n \times m}$ where $U(i,j) = 1$ iff $T_i$ contains $j$. Then $U$ is a strongly separating matrix.*

*Proof.* From the definition of antichain, for any two sets $T_i, T_j \in \mathcal{T}$, there exists $k$ and $k'$ such that $k \in T_i \setminus T_j$ and $k' \in T_j \setminus T_i$. So, we have $U(i,k) = U(j,k') = 1$ and $U(i,k') = U(j,k) = 0$. It follows that $U$ is a strongly separating matrix from the definition. □

In the previous lemma, we gave a construction of a strongly separating matrix that corresponds to an antichain. In the next lemma, we show that given a strongly separating system, we can also obtain a corresponding antichain.

**Lemma A.9.** *Let $\mathcal{S} = \{S_1, S_2, \cdots S_m\}$ be a strongly separating set system defined on $\{v_1, v_2, \cdots v_n\}$. Construct a strongly separating matrix $U \in \{0,1\}^{n \times m}$ where $U(i,j) = 1$ iff $S_j$ contains $v_i$. Define a collection of sets $\mathcal{T} = \{T_1, T_2, \cdots T_n\}$ defined over the column indices of $U$ i.e., $\{1, 2, \cdots m\}$ such that $j \in T_i$ iff $U(i,j) = 1$. Then, $\mathcal{T}$ is an antichain.*

*Proof.* From the definition of strongly separating system, we have for every two nodes $v_i, v_j \in \mathcal{S}$, there exists $S_k$ and $S_{k'}$ such that $v_i \in S_k \setminus S_{k'}$ and $v_j \in S_{k'} \setminus S_k$. This implies $k \in T_i \setminus T_j$ and $k' \in T_j \setminus T_i$ as $U(i,k) = U(j,k') = 1$ and $U(i,k') = U(j,k) = 0$. Therefore, for every two sets $T_i$ and $T_j$ in $\mathcal{T}$, we have $T_i \not\subset T_j$ and $T_j \not\subset T_i$. Hence, $\mathcal{T}$ is an antichain. □

**Lemma A.10.** *LYM inequality Jukna [2011]. Suppose $\mathcal{S}$ represent an antichain defined over the elements $\{1, 2, \cdots m\}$. Let $a_k = |\{T \mid T \in \mathcal{S} \text{ where } |T| = k\}|$ defined for all $k \in [m]$, then,*

$$\sum_{k=0}^{m} \frac{a_k}{\binom{m}{k}} \leq 1.$$

**Definition A.11.** *Jukna [2011]. A neighbor of a binary vector $v$ is a vector which can be obtained from $v$ by flipping one of its 1-entries to 0. A shadow of a set $A \subseteq \{0,1\}^m$ of vectors is the set of all its neighbors and denoted by $\partial(A)$.*

Suppose $A \subseteq \{0,1\}^m$ consists of weight $k$ vectors i.e., for all $v \in A, \|v\|_1 = k$ . Then, there is an interesting representation for $|A|$ i.e., size of $A$ called the *k-cascade* form,

$$|A| = \binom{a_k}{k} + \binom{a_{k-1}}{k-1} + \binom{a_{k-2}}{k-2} + \cdots + \binom{a_s}{s} \text{ where } a_k > a_{k-1} > \cdots a_s \geq s \geq 1.$$

Moreover, this representation is unique and for every $|A| \geq 1$, there exists a *k-cascade* form. Given a set $A$ of such vectors, we can make the following observation.

**Observation A.12.** *Let $B \subseteq \{0,1\}^m$ be a collection of vectors with weight exactly $k-1$. If $A \cup B$ is an antichain, then, $B \cap \partial(A) = \phi$.*

The above observation implies that if we want to maximize the number of weight $k-1$ vectors to get a collection of weight $k$ and $k-1$ vectors that form an antichain, then, we have to choose weight $k$ vectors that has a small shadow. Now, we describe the statement of the famous Kruskal-Katona theorem that gives a lower bound on the size of shadow of $A$.

**Theorem A.13** (Kruskal-Katona Theorem Jukna [2011])**.** *Consider a set $A \subseteq \{0,1\}^m$ of vectors such that for all $v \in A, \|v\|_1 = k$ and the k-cascade form is*

$$|A| = \binom{a_k}{k} + \binom{a_{k-1}}{k-1} + \binom{a_{k-2}}{k-2} + \cdots + \binom{a_s}{s}.$$

*Then,*

$$|\partial(A)| \geq \binom{a_k}{k-1} + \binom{a_{k-1}}{k-2} + \binom{a_{k-2}}{k-2} + \cdots + \binom{a_s}{s-1}.$$

**Definition A.14.** *(Colexicographic Ordering) Let $u$ and $v$ be two distinct vectors from $\{0,1\}^m$. In the colexicographic ordering $u$ appears before $v$ if for some $i$, $u(i) = 0, v(i) = 1$ and $u(j) = v(j)$ for all $j > i$.*

We now state a result that the colexicographic ordering (or colex order) of all vectors of $\{0,1\}^m$ achieves the Kruskal-Katona theorem lower bound. Therefore, we can generate a sequence of any number of vectors with weight $k$ that has the smallest possible shadow.

**Lemma A.15.** *Proposition 10.17 from Jukna [2011]. Using the first $T$ of weight $k$ vectors in the colex ordering of $\{0,1\}^m$, we can obtain a collection $A \subseteq \{0,1\}^m$ such that $|\partial(A)| = \binom{a_k}{k-1} + \binom{a_{k-1}}{k-2} + \binom{a_{k-2}}{k-2} + \cdots + \binom{a_s}{s-1}$ where the $k$-cascade form of $T = |A| = \binom{a_k}{k} + \binom{a_{k-1}}{k-1} + \binom{a_{k-2}}{k-2} + \cdots + \binom{a_s}{s}$.*

We state the Flat Antichain theorem, that we will use later.

**Theorem A.16** (Flat Antichain Theorem)**.** *Kisvölcsey [2006] If $\mathcal{A}$ is an antichain, then, there exists another antichain $\mathcal{B}$ defined over same elements, such that $|\mathcal{A}| = |\mathcal{B}|$, $\sum_{A\in\mathcal{A}} |A| = \sum_{B\in\mathcal{B}} |B|$ and for every $B \in \mathcal{B}$, we have $|B| \in \{d-1, d\}$ for some positive integer $d$.*

## A.4 $(1+\epsilon)$-approximation Algorithm

Algorithm $\epsilon$-SSMatrixis an adaptation of Algorithm 4 of [Hyttinen et al., 2013a] for the linear cost model setting. From Lemma A.9 and A.8, it is clear that constructing a strongly separating set system is equivalent to constructing an antichain. A consequence of Flat Antichain theorem [Kisvölcsey, 2006] is that for every antichain $\mathcal{A}$ there is another antichain $\mathcal{B}$ of same size such that $\sum_{A\in\mathcal{A}} |A| = \sum_{B\in\mathcal{B}} |B|$ and $\mathcal{B}$ has sets of cardinality either $d$ or $d-1$ for some positive integer $d$. Therefore, the problem of finding a separating set system reduces to finding an appropriate antichain with weights $d$ and $d-1$ that minimizes the objective (assuming all nodes have cost equal to 1).

**Corollary A.17.** *Hyttinen et al. [2013a] Flat Antichain theorem implies Algorithm 4 achieves optimal cost assuming all nodes have unit costs.*

Algorithm 4 of [Hyttinen et al., 2013a] is a consequence of Kruskal-Katona theorem; using colexicographic ordering we can maximize the $d-1$ weight vectors in an antichain of size $n$ consisting of weight $d$ and $d-1$ vectors. Therefore, choosing $d = k$ where $\binom{m}{k-1} < n \leq \binom{m}{k}$, they consider all possible number of vectors of weight $k$ and find the one with the minimum number of weight $k$ vectors. However, unlike [Hyttinen et al., 2013a], we have to deal with different costs of intervention for each node. We adopt a greedy strategy, where we assign the vectors (obtained using the previous algorithm) in the increasing order of weight to the nodes in the decreasing order of their costs. Observe that our Algorithm $\epsilon$-SSMatrixassigns vectors of weight $k-1$ or $k$ that are relatively high to the nodes with large costs. Surprisingly, we show that when the costs are bounded by $\approx \epsilon n^\epsilon$, and number of interventions $m \leq n^\epsilon$, it achieves a $1+\epsilon$-approximation.

**Lemma A.18.** *Let $U$ represent the output of Algorithm $\epsilon$-SSMatrix. Then, $U$ is a strongly separating matrix.*

*Proof.* From Observation A.12, we have that our set of weight $k$ vectors $A_t$ and set of weight $k-1$ vectors given by $B_t = A_t \setminus \partial(A_t)$ satisfy $B_t \cap \partial(A_t) = \phi$. So, the collection $A_t \cup B_t$ is an antichain. In Algorithm $\epsilon$-SSMatrix, $U$ and $\widetilde{U}$ contain the same collection of vectors, only differing in the ordering $\zeta$. From Lemma A.8, we have $U$ constructed from $A_t \cup B_t$ is a strongly separating matrix. □

The following lemma follows from LYM inequality in Lemma A.10.

**Algorithm 4** $\epsilon$-SSMATRIX$(V, m)$

Let $\widetilde{U} \in \{0,1\}^{n \times m}$ be initialized with all zeros
Find $k$ satisfying $\binom{m}{k-1} < n \leq \binom{m}{k}$
**for** $t = 0$ to $n$ **do**
    Let $A_t$ denote the first $t$ vectors in the colex ordering of $\{0,1\}^m$ with weight $k$. Calculate $|\partial(A_t)|$ using Lemma A.15.
    **if** $t - |\partial(A_t)| + \binom{m}{k-1} \geq n$ **then**
        For the rows $\widetilde{U}(j)$ with $n - t + 1 \leq j \leq n$ assign the vectors of weight $k$ using $A_t$
        For the rows $\widetilde{U}(j)$ with $j \leq n - t$, assign vectors of weight $k-1$ from $\{0,1\}^m$ that are not contained in $\partial(A_t)$
        **break**;
    **end if**
**end for**
Let $\zeta$ denote the ordering of rows of $\widetilde{U}$ in the increasing order of weight.
For every $i \in [n]$ assign $U(i) = \widetilde{U}(\zeta(i))$ where $i^{th}$ row of $U$ corresponds to the node with $i^{th}$ largest cost.
**Return** $U$

**Lemma A.19.** *Let $U_{\text{OPT}}$ represent the optimum solution with $a_q^*$ representing the number of rows of $U$ with weight $q$. Then, for any $t \leq n$ :*

$$\sum_{q=1}^{t} a_q^* \leq \binom{m}{t}.$$

*Proof.* From Lemma A.18 and Corollary A.17, we know that the matrix $U_{\text{OPT}}$ is a strongly separating matrix. Therefore, using Lemma A.9, we can construct a collection $\mathcal{T}$ defined over $\{1, 2, \cdots, m\}$ such that $\mathcal{T}$ is an antichain. $T_i \in \mathcal{T}$ corresponds to a row of $U_{\text{OPT}}$ and $|T_i| = \|U_{\text{OPT}}(i)\|_1$ represents the weight of $i^{th}$ row of $U$. Applying LYM inequality from Lemma A.10 gives us:

$$\sum_{q=1}^{t} \frac{a_q^*}{\binom{m}{t}} \leq \sum_{q=1}^{t} \frac{a_q^*}{\binom{m}{q}} \leq \sum_{q=0}^{m} \frac{a_q^*}{\binom{m}{q}} \leq 1$$

and so $\sum_{q=1}^{t} a_q^* \leq \binom{m}{t}$. □

The next lemma gives an upper bound for $\binom{m}{t}$ that can be used to simplify the statement of the Theorem 3.4.

**Lemma A.20.** *If $6/k \leq \epsilon \leq 1/2$ and $m \geq 2 \log_2 n$:*

$$\binom{m}{t} \leq 2n \cdot 2^{-(\epsilon k/6) \log_2(m/(2k))}.$$

*Proof.* By the definition of $k$,

$$\binom{m}{t} = \binom{m}{k-1}\binom{m}{t} \Big/ \binom{m}{k-1} < n\binom{m}{t} \Big/ \binom{m}{k-1}.$$

Let $H(x)$ denote the binary entropy function. Note that $t = \lfloor k - \epsilon k/3 \rfloor$. Therefore,

$$(k-1) - t \geq k - 1 - k + \epsilon k/3 = \epsilon k/3 - 1 \geq \epsilon k/6,$$

and that for all $x \in [t/m, (k-1)/m]$,

$$H'(x) \geq H'\left(\frac{k-1}{m}\right) = \log_2\left(\frac{m}{k-1} - 1\right) \geq \log_2\left(\frac{m}{2k}\right),$$

where we used the assumption $t/m \le (k-1)/m \le 1/2$. Hence,

$$|H((k-1)/m) - H(t/m)| \ge \frac{\epsilon k/6}{m} \log_2 \left(\frac{m}{2k}\right).$$

Using the bound from (MacWilliams and Sloane [1977], Page 309)

$$\sqrt{\frac{a}{8b(a-b)}} 2^{mH(b/a)} \le \binom{a}{b} \le \sqrt{\frac{a}{2\pi b(a-b)}} 2^{mH(b/a)},$$

we get that

$$\binom{m}{t} / \binom{m}{k-1} \le 2^{m(H(t/m)-H((k-1)/m)} \sqrt{\frac{8(k-1)(m-k+1)}{2\pi t(m-t)}} \le 2 \cdot 2^{-(\epsilon k/6)\log_2(m/(2k))}$$

where the last inequality used $\epsilon \le 1$.

□

**Corollary A.21.** *(Corollary 3.5 Restated). Algorithm* $\epsilon$-SSMatrix *is a* $(1+\epsilon)$*-approximation if the maximum cost satisfies*

$$c_{max} \le \epsilon/6 \cdot 2^{(\epsilon k/6)\log_2(m/(2k))}$$

*assuming* $n^{\epsilon/6} \ge m \ge 2\log_2 n$*. If a)* $m \ge (2\log_2 n)^{c_1}$ *for some constant* $c_1 > 1$ *or b)* $4\log_2 n \le m \le c_2 \log_2 n$ *for some constant* $c_2$ *then the RHS bound is at least* $\epsilon/6 \cdot n^{\Omega(\epsilon)}$*.*

*Proof.* First note that $k \ge \log_m n$ since

$$m^k \ge \binom{m}{k} \ge n.$$

When $2\log n \le m \le n^\epsilon$, we have $k \ge \log_m n \ge \frac{6}{\epsilon}$. From the previous lemma A.20,

$$c_{max} \le \epsilon/6 \cdot 2^{(\epsilon k/6)\log_2(m/(2k))} \le \epsilon n/3 \binom{m}{t}$$

Using Theorem 3.4, we have that Algorithm $\epsilon$-SSMatrix is a $(1+\epsilon)$-approximation. We next consider the simplification in Part (a). If $m \ge (2\log_2 n)^{c_1}$ for some $c_1 > 1$ then $k \le \log_2 n$ as $\binom{m}{\log n} \ge n$. So $2k \le 2\log_2 n \le m^{1/c_1}$. Hence,

$$\log_2(m/(2k)) \ge (1 - 1/c_1)\log_2 m$$

and so

$$2^{(\epsilon k/6)\log_2(m/(2k))} \ge 2^{(\epsilon k(1-1/c_1)(\log_2 m)/6)} \ge n^{\frac{\epsilon(1-1/c_1)}{6}}.$$

where the last inequality follows since $k \ge \log_m n$.
We next consider the simplification in Part (b). Now suppose $m \le c_2 \log_2 n$ for some constant $c_2 \ge 2$ then, $k \ge \log_{ec_2} n$ since

$$(c_2 e)^k \ge (me/k)^k \ge \binom{m}{k} \ge n\ .$$

Note that for $m \ge 4\log n$,

$$\log_2(m/(2k)) \ge \log_2(4\log n/(2\log_2 n)) \ge 1$$

and so

$$2^{(\epsilon k/6)\log_2(m/(2k))} \ge 2^{(\epsilon k/6)} \ge \quad n^{\frac{\epsilon}{6\log_{ec_2} 2}}\ .$$

□

## B Missing Details from Section 4

**Removing Dependence on $\tau$ in Algorithms** RecoverG, LatentsNEdges **and** LatentsWEdges**.** Let $\mathcal{G}$ be a $\tau$-causal graph. Algorithms RecoverG, LatentsNEdges, and LatentsWEdges assume that we know $\tau$, however this assumption can be easily removed. For a fixed $\tau$, let $\mathcal{G}_\tau$ be graph returned after going through all these above algorithms. Given $\mathcal{G}_\tau$, checking whether $v_k$ is a $p$-collider for some pair $v_i, v_j$ is simple, iterate over all paths between $v_i$ and $v_j$ that include $v_k$. Let $\Pi = \{\pi_1, \ldots, \pi_r\}$ be these paths. For each $\pi_w \in \Pi$, remove the edges in $\pi_w$ from $\mathcal{G}_\tau$ see if $v_k$ has a descendant in $\mathrm{Pa}(v_i) \cup \mathrm{Pa}(v_j)$ in this modified graph. If this holds for any path $\pi_w \in \Pi$, then $v_k$ is a $p$-collider for the pair $v_i, v_j$. We describe an efficient algorithm for finding $p$-colliders in section 5.

The idea is as follows, we invoke Algorithms RecoverG, LatentsNEdges and LatentsWEdges for $\tau = 1, 2, 4, ..,$ until we find the first $\widehat{\tau}$ and $2\widehat{\tau}$ such that $\mathcal{G}_{\widehat{\tau}} = \mathcal{G}_{2\widehat{\tau}}$. We now check whether the observable nodes in $\mathcal{G}_{\widehat{\tau}}$ has at most $\widehat{\tau}$ $p$-colliders, if so we are output $\mathcal{G}_{\widehat{\tau}}$ (and $\widehat{\tau}$). Otherwise, we continue by doubling $\tau$, i.e., by considering $2\widehat{\tau}$ and $4\widehat{\tau}$. By increasing $\tau$ by a constant factor, it is easy to see that process will stop in at most $\log(2\tau)$ steps and when it stops it produces the correct observable graph $\mathcal{G}$ and also that $\widehat{\tau} \leq 2\tau$. Overall, this will increase the number of interventions in Theorem 4.12 by a factor of $O(\log \tau)$ (to $O(\tau^2 \log n \log \tau + n\tau \log n \log \tau)$ interventions). Through a union bound, the same success probability of $1 - O(1/n^2)$ can be ensured by adjusting the constants.

**Lemma B.1** (Lemma 4.4 Restated)**.** *Let $v_i \in Anc(v_j)$. $v_i \perp\!\!\!\perp v_j \mid \mathrm{do}(v_i \cup P_{ij}), \mathrm{Anc}(v_j) \setminus \{v_i\}$ iff $v_i \notin \mathrm{Pa}(v_j)$.*

*Proof.* Suppose $v_i \in \mathrm{Anc}(v_j) \setminus \mathrm{Pa}(v_j)$. Consider the interventional distribution $\mathrm{do}(v_i \cup P_{ij})$ where $P_{ij}$ is the set of $p$-colliders between $v_i$ and $v_j$. We intervene on $v_i$ to block the path (if present) given by $v_i \leftarrow \widetilde{l} \rightarrow v_j$ where $\widetilde{l} \in L$. Consider all the remaining *undirected* paths between $v_i$ and $v_j$ denoted by $\Pi_{ij}$. We divide $\Pi_{ij}$ into three cases. Let $\pi \in \Pi_{ij}$ be a path from $v_i$ to $v_j$.

1. $\pi$ contains no *colliders*, then, $\pi$ is blocked by $\mathrm{Anc}(v_j) \setminus \{v_i\}$. As $\pi$ contains no colliders, we can write $\pi = v_i \cdots v_k \rightarrow v_j$ where $v_k \in \mathrm{Anc}(v_j)$. As we are conditioning on $\mathrm{Anc}(v_j) \setminus \{v_i\} \supseteq \{v_k\}$, $\pi$ is blocked by $v_k$.

2. $\pi$ contains colliders but not a $p$-collider. We argue that there are no collider nodes in $\pi$ that are also in $\mathrm{Anc}(v_j) \setminus \{v_i\}$. As there are no $p$-colliders, it means that all the colliders have no descendants in the conditioning set $\mathrm{Anc}(v_j) \setminus \{v_i\}$. Because if a collider $v_c$ have a descendant in $\mathrm{Anc}(v_j) \setminus \{v_i\}$, then there is a path from $v_c$ to $\mathrm{Pa}(v_j)$ through $\mathrm{Anc}(v_j) \setminus \{v_i\}$. This means that $v_c$ is a $p$-collider, contradicting our assumption. Therefore, from Rule-2 of $d$-separation, $\pi$ is blocked.

3. $\pi$ contains at least one $p$-collider. We are intervening on $P_{ij}$ containing all the $p$-colliders. In the intervened mutilated graph, all the $p$-colliders no longer have an incoming arrow and therefore are not colliders. So $\pi$ is blocked.

If $v_i \notin \mathrm{Pa}(v_j)$, we can conclude that $v_i \perp\!\!\!\perp v_j \mid \mathrm{do}(\{v_i\} \cup P_{ij}), \mathrm{Anc}(v_j) \setminus \{v_i\}$. Suppose $v_i \in \mathrm{Pa}(v_j)$. In the interventional distribution $\mathrm{do}(\{v_i\} \cup P_{ij})$, we still have $v_i \in \mathrm{Pa}(v_j)$ and any conditioning will not block the path $\pi = v_i \rightarrow v_j$. Therefore, $v_i \not\perp\!\!\!\perp v_j \mid (\mathrm{do}(\{v_i\} \cup P_{ij}), \mathrm{Anc}(v_j) \setminus \{v_i\}$ if $v_i \in \mathrm{Pa}(v_j)$. □

**Lemma B.2.** *Let $\mathcal{G}(V \cup L, E \cup E_L)$ be a $\tau$-causal graph with observable graph $G(V, E)$. Given an ancestral graph $\mathrm{Anc}(G)$, Algorithm RecoverG correctly recovers all edges in the observable graph with probability at least $1 - 1/n^2$.*

*Proof.* Let $\tau' = \max\{\tau, 2\}$. From Lemma 4.4, we can recover the edges of $G$ provided we know the $p$-colliders between every pair of nodes. As we do not know the graph $G$, we devise a randomized strategy to hit all the $p$-colliders, whilst ensuring that we don't create a lot of interventions. Suppose $\max_{(v_i,v_j)\in V\times V}|P_{ij}| \leq \tau$. We show that with high probability, $\forall v_i \in Anc(v_j)$, $\exists A_t$ such that $\{v_i\} \cup P_{ij} \subseteq A_t$ and $v_j \notin A_t$. We can then use the CI-test described in Lemma 4.4 to verify whether $v_i$ is a parent of $v_j$. In Algorithm RecoverG, we repeat this procedure on every edge of $Anc(G)$ and output $G$.

Suppose $v_i \in Anc(v_j)$. Let $\Gamma_t$ denote the event that $A_t \in \mathcal{A}_\tau$ such that $\{v_i\} \cup P_{ij} \subseteq A_t$ and $v_j \notin A_t$ for a fixed $t \in \{1, \ldots, 72\tau' \log n\}$. Let $T = 72\tau' \log n$. As we include a vertex $v_i \in A_t$ with probability $1 - 1/\tau'$, we obtain

$$\Pr[\Gamma_t] = \left(1 - \frac{1}{\tau'}\right)^{|P_{ij}|+1} \frac{1}{\tau'} \geq \left(1 - \frac{1}{\tau'}\right)^{\tau'+1} \frac{1}{\tau'}.$$

Using the inequality $(1 + \frac{x}{n})^n \geq e^x(1 - \frac{x^2}{n})$ for $|x| \leq n$, and since $\tau' \geq 2$ we have:

$$\Pr[\Gamma_t] \geq \frac{1}{e^{\tau'+1/\tau'}}\left(1 - \frac{(\tau'+1)^2}{\tau'^2(\tau'+1)}\right)\frac{1}{\tau'} \geq \frac{1}{18\tau'}$$
$$\Rightarrow \Pr[\bar{\Gamma}_t] \leq 1 - \frac{1}{18\tau'} \text{ and } \Pr[\exists t \in [T] : \Gamma_t] \geq 1 - \left(1 - \frac{1}{18\tau'}\right)^{72\tau' \log n}.$$

Using the inequality $(1 + \frac{x}{n})^n \leq e^x$ for $|x| \leq n$ we have:

$$\Pr[\exists t \in [T] : \Gamma_t] \geq 1 - \frac{1}{n^4}.$$

So the probability that there exists at least one set $A_t$ for the given pair $v_i, v_j$ for which $v_i \cup P_{ij} \subseteq A_t$ and $v_j \notin A_t$ is at least $1 - \frac{1}{n^4}$.[4] To ensure this probability of success for every pair of variables, we use a union bound over the $n^2$ node pairs. □

**Proposition B.3** (Proposition 4.5 Restated)**.** *Let $\mathcal{G}(V \cup L, E \cup E_L)$ be a $\tau$-causal graph with observable graph $G(V, E)$. There exists a procedure to recover the observable graph using $O(\tau \log n + \log n)$ many interventions with probability at least $1 - 1/n^2$.*

*Proof.* As is well-known, e.g. [Kocaoglu et al., 2017b], a strongly separating set system can be constructed with $m = 2\log n$ interventions by using the binary encoding of the numbers $1, \ldots, n$. Two intervention sets are constructed for every bit location $k \in [\log n]$, one with any node $v_i$ if the number $i$ has $k$th bit set to 1, and other with any node $v_i$ if the number $i$ has $k$th bit set to 0. Therefore, we require $2\log n$ interventions to obtain ancestral graph Anc$(G)$ of the observable graph. From Lemma B.2, we require $O(\tau \log n)$ interventions to recover all the edges of observable graph of $G$ from Anc$(G)$ with probability $1 - \frac{1}{n^2}$. Therefore, using $O(\tau \log n)$ interventions, Algorithm RecoverGcan recover the observable graph $G(V, E)$ with high probability. □

It is well established that $\log(\chi(G))$ interventions are necessary and sufficient in the causally sufficient systems (where there are no latents) where $\chi(G)$ is the chromatic number of $G$. Generalized over all graphs this becomes $\log(n)$. Our following lower bound shows that, even if there are no latent variables in the underlying system, if the algorithm cannot rule latents out, and needs to consider latents as a possibility to compute the graph skeleton, then $\Omega(n)$ interventions are necessary. Shanmugam et al. [2015] provide a lower bound in a different setting, when the intervention sets are required to have only limited number of variables.

[4] Note by adjusting the constant 72, we could have pushed this probability to any $1/n^c$ for constant $c$.

**Algorithm 5** LatentsNEdges$(G(V,E), \mathcal{D}_\tau)$

$L \leftarrow \phi, E_L \leftarrow \phi$
**for** $(v_i, v_j) \notin E$ **do**
  Let $\mathcal{D}_{ij} = \{D \mid D \in \mathcal{D}_\tau \text{ and } v_i, v_j \notin D\}$
  **if** $v_i \not\perp\!\!\!\perp v_j \mid do(D) \cup \text{Pa}(v_i) \cup \text{Pa}(v_j)$ for every $D \in \mathcal{D}_{ij}$ **then**
    $L \leftarrow L \cup l_{ij}, E_L \leftarrow E_L \cup \{(l_{ij}, v_i), (l_{ij}, v_j)\}$
  **end if**
**end for**
**return** $\mathcal{G}(V \cup L, E \cup E_L)$

**Proposition B.4** (Proposition 4.6 Restated)**.** *There exists a graph causal $\mathcal{G}(V \cup L, E \cup E_L)$ such that every non-adaptive algorithm requires $\Omega(n)$ many interventions to recover even the observable graph $G(V,E)$ of $\mathcal{G}$.*

*Proof.* Consider an ordering of observable variables given by $v_1, v_2, \cdots, v_n$. Let $G$ be a graph with all directed edges $(v_a, v_b)$ for all $b > a$. Suppose the set of interventions generated by the non-adaptive algorithm is given by $\mathcal{H}$. Now consider $v_i$ for some fixed $i \geq \frac{n}{4}$.

We claim that if every intervention $H \in \mathcal{H}$ is such that for some $j \in \{3, \cdots, i-1\}$, $v_j \notin H$, then there exists a graph $G_i$ such that $G$ and $G_i$ are both indistinguishable under all the interventions in $\mathcal{H}$ irrespective of other conditioning. Now consider any set $H_j \subseteq (\{v_1, v_2, \cdots, v_{i-1}\} \setminus \{v_j\}) \bigcup \{v_{i+1}, \cdots, v_n\}$. Let $G_i$ be such that it contains all the directed edges $(v_a, v_b)$ for all $b > a$ but does not contain the directed edge $(v_1, v_i)$. To distinguish between $G$ and $G_i$ one needs to determine whether $v_1 \to v_i$. Note that any intervention we use to determine the edge should contain $v_1$ to rule out the possibility of the influence of latent $v_1 \leftarrow l_{1i} \to v_i$ on the CI-tests we perform. Now, under $\text{do}(H_j)$, there are only two CI-tests possible to determine whether $v_1 \to v_i : v_1 \perp\!\!\!\perp v_i \mid v_j, \text{do}(H_j)$ and $v_1 \perp\!\!\!\perp v_i \mid \text{do}(H_j)$. However, for both graphs $G$ and $G_i$, both these independence tests will always turn out negative. In the former case, it is because $v_j$ will be a collider on the path $v_i, v_j, v_{j-1}, v_i$, and in the latter case there is a path $v_1, v_j, v_i$ that is not blocked. In other words, the CI-tests will provide no information to distinguish between $G$ and $G_i$, unless $\mathcal{H}$ contains the set $\{v_1, v_3, \ldots, v_{i-1}\}$.

One can similarly construct these $G_i$'s for all $i \geq \frac{n}{4}$, thereby $\mathcal{H}$ needs to contain the intervention sets $\{v_1, v_3, \ldots, v_{i-1}\}$ for all $n/4 \leq i \leq n$ to separate $G$ from all the $G_i$'s. This proves the claim. □

## B.1 Latents Affecting Non-adjacent Nodes in $G$

Let $\bar{E} = \{(v_i, v_j) \mid (v_i, v_j) \notin E\}$ be the set of non-edges in $G$. The entire procedure for finding latents between non-adjacent nodes in $G$ is described in Algorithm LatentsNEdges. Similar to Algorithm RecoverG, we block the paths by conditioning on parents and intervening on $p$-colliders. The idea is based on the observation that for any non-adjacent pair $v_i, v_j$ an intervention on the set $P_{ij}$ and conditioning on the parents of $v_i$ and $v_j$ will make $v_i$ and $v_j$ independent, unless there is a latent between then. The following lemma formalizes this idea.

**Lemma B.5.** *Suppose $(v_i, v_j) \in \bar{E}$. Then, $v_i \perp\!\!\!\perp v_j \mid \text{do}(P_{ij}), \text{Pa}(v_i) \cup \text{Pa}(v_j)$ iff $v_i$ and $v_j$ has no latent between them.*

*Proof.* Suppose there is no latent between $v_i$ and $v_j$. We follow the proof similar to the Lemma 4.4. Consider the pair of variables $v_i$ and $v_j$ and all the paths between them $\Pi_{ij}$. Let $\pi \in \Pi_{ij}$.

1. Let $\pi$ be a path not containing any colliders. Using Rule-1 of $d$-separation, we can block $\pi$ by conditioning on either $\text{Pa}(v_i)$ or $\text{Pa}(v_j)$.

2. If $\pi$ contains colliders and no $p$-colliders, then, using Rule-2 of $d$-separation, $\pi$ is blocked as the colliders have no descendants in $\text{Pa}(v_i) \cup \text{Pa}(v_j)$.

3. We block the paths $\pi$ containing $p$-colliders by intervening on $P_{ij}$

As all the paths in $\Pi_{ij}$ are blocked, we have $v_i \perp\!\!\!\perp v_j \mid \text{do}(P_{ij}), \text{Pa}(v_i) \cup \text{Pa}(v_j)$. If there is a latent $l_{ij}$ then the path $v_i \leftarrow l_{ij} \rightarrow v_j$ is not blocked and therefore $v_i \not\perp\!\!\!\perp v_j \mid (\text{do}(P_{ij}), \text{Pa}(v_i) \cup \text{Pa}(v_j))$. □

Formally, let $D_t \subseteq V$ for $t \in \{1, 2, \cdots, 24\tau'^2 \log n\}$ be constructed by including every variable $v_i \in V$ with probability $1 - \frac{1}{\tau'}$ where $\tau' = \max\{\tau, 2\}$. Let $\mathcal{D}_\tau = \{D_1, \cdots, D_{24\tau'^2 \log n}\}$ be the collection of the set $D_t$'s. Using these interventions $\mathcal{D}_\tau$, we argue that we can recover all the latents between non-edges of $G$ correctly with high probability.

**Proposition B.6** (Proposition 4.7 Restated)**.** *Let $\mathcal{G}(V \cup L, E \cup E_L)$ be a $\tau$-causal graph with observable graph $G(V, E)$. Algorithm* LatentsNEdges *with $O(\tau^2 \log n + \log n)$ many interventions recovers all latents effecting pairs of non-adjacent nodes in the observable graph $G$ with probability at least $1 - 1/n^2$.*

*Proof.* We follow a proof similar to Lemma B.2. Consider a pair of variables $v_i$ and $v_j$ such that there is no edge between them in $G$. From Lemma B.5, we know that by intervening on all the colliders between $v_i$ and $v_j$, we can identify the presence of a latent. In Algorithm LatentsNEdges, we iterate over sets in $\mathcal{D}_{ij}$. As $\mathcal{D}_{ij} \subseteq \mathcal{D}_\tau$, we have $|\mathcal{D}_{ij}| \leq 24\tau'^2 \log n$. Let $\Gamma_t$ denote the event that $D_t \in \mathcal{D}_{ij}$ is such that $v_i, v_j \notin D_t$ and $P_{ij} \subseteq D_t$ for a fixed $t \in \{1, \cdots, 24\tau'^2 \log n\}$. Let $T = 24\tau'^2 \log n$.

$$\Pr[\Gamma_t] = \left(1 - \frac{1}{\tau'}\right)^{|P_{ij}|} \frac{1}{\tau'^2} \geq \left(1 - \frac{1}{\tau'}\right)^{\tau'} \frac{1}{\tau'^2}.$$

Using the inequality $(1 + \frac{x}{n})^n \geq e^x(1 - \frac{x^2}{n})$ for $|x| \leq n$, and since $\tau' \geq 2$ we have:

$$\Pr[\Gamma_t] \geq \tfrac{1}{\mathrm{e}}(1 - \tfrac{1}{\tau'})\tfrac{1}{\tau'^2} \geq \tfrac{1}{2\mathrm{e}\tau'^2}$$
$$\Rightarrow \Pr[\bar{\Gamma}_t] \leq 1 - \tfrac{1}{6\tau'^2} \text{ and } \Pr[\exists t \in [T] : \Gamma_t] \geq 1 - \left(1 - \tfrac{1}{6\tau'^2}\right)^{24\tau'^2 \log n}.$$

Using the inequality $(1 + \frac{x}{n})^n \leq e^x$ for $|x| \leq n$ we have:

$$\Pr[\exists t \in [T] : \Gamma_t] \geq 1 - \frac{1}{n^4}.$$

So the probability that there exists a set $D_t$ for which $v_i, v_j \notin D_t$ and $P_{ij} \subseteq D_t$ is at least $1 - \frac{1}{n^4}$. A union bound over at most $n^2$ pair of variables completes the proof. □

## B.2 Latent Affecting Adjacent Nodes in $G$

We follow an approach similar to the one presented in section B.1 for detecting the presence of latent between an edge $v_i \rightarrow v_j$ in $G$. In Algorithm 3, we block all the paths (excluding the edge) between the variables $v_i$ and $v_j$ using a conditioning set $\text{Pa}(v_j)$ in the intervention distribution $\text{do}(\text{Pa}(v_i) \cup P_{ij})$ in the *do-see* tests we perform. This idea is formalized using the following lemma.

**Lemma B.7.** *Suppose $v_i \rightarrow v_j \in G$. Let $l_{tj}$ be a latent between $v_t$ and $v_j$ where $v_t \neq v_i$ and $v_i, v_j \notin B$, $P_{ij} \subseteq B$. Then, $l_{tj} \perp\!\!\!\perp v_i \mid \text{Pa}(v_j), \text{do}(B \cup \{v_i\} \cup \text{Pa}(v_i))$ and $l_{tj} \perp\!\!\!\perp v_i \mid \text{Pa}(v_j), \text{do}(\text{Pa}(v_i) \cup B)$.*

*Proof.* The proof goes through by analyzing various cases. We give a detailed outline of the proof.

Claim 1: $l_{tj} \perp\!\!\!\perp v_i \mid \text{Pa}(v_j), \text{do}(B \cup \{v_i\} \cup \text{Pa}(v_i))$. Suppose $v_t \in \text{Pa}(v_i) \cup B$. Consider all the paths between $v_i$ and $l_{tj}$ in the interventional distribution $\text{do}(B \cup \{v_i\} \cup \text{Pa}(v_i))$. The only paths

that are not separated because of the intervention are $l_{tj} \to v_j \leftarrow v_i$, $l_{tj} \to v_j \leftarrow v_k \cdots \leftarrow v_i$ where $v_k \in \text{Pa}(v_j)$, and $l_{tj} \to v_j \to \cdots \leftarrow v_i$. As we are not conditioning on $v_j$, $l_{tj} \to v_j \leftarrow v_i$ is blocked (Rule-2 in $d$-separation); conditioning on $\text{Pa}(v_j) \ni v_k$ block the paths $l_{tj} \to v_j \leftarrow v_k \cdots \leftarrow v_i$ (Rule-1 in $d$-separation); and $l_{tj} \to v_j \to \cdots \leftarrow v_i$ paths have a collider that is not $\text{Pa}(v_j)$ hence blocked by Rule-2 in $d$-separation.

Suppose $v_t \notin \text{Pa}(v_i) \cup B$. As before it follows that all paths between $l_{tj}$ and $v_i$ going through $v_j$ are blocked. All other paths between $l_{tj}$ and $v_i$ should have a collider. This is because in any such path $\pi$ the only edge from $l_{tj}$ is $l_{tj} \to v_t$ and the edges that remain at $v_i$ are outgoing. It is easy to see that the collider on this path $\pi$ can't be in $\text{Pa}(v_j)$ because otherwise it will also be a $p$-collider between $v_i$ and $v_j$ which are intervened on through $B$. When there is a collider on the path that is not in the conditioning set, then the path is blocked (Rule-2 in $d$-separation). The same holds for all paths between $l_{tj}$ and $v_i$.

Claim 2: $l_{tj} \perp\!\!\!\perp v_i \mid \text{Pa}(v_j), \text{do}(B \cup \text{Pa}(v_i))$. Consider all the paths between $l_{tj}$ and $v_i$. Using the above arguments, we have that all paths containing $v_j$ are blocked. All other paths between $l_{tj}$ and $v_i$ should have a collider. This is because in any such path $\pi$ the only edge from $l_{tj}$ is $l_{tj} \to v_t$ and $\pi$ will end at $v_i$ either as $l_{tj} \cdots \leftarrow v_i$ or $l_{tj} \to \cdots \leftarrow v_k \to v_i$ where $v_k \in \text{Pa}(v_i)$. It is again easy to see that the collider on this path $\pi$ can't be in $\text{Pa}(v_j)$ because otherwise it will also be a $p$-collider between $v_i$ and $v_j$ which are intervened on through $B$. As before, when there is a collider on the path that is not in the conditioning set, then the path is blocked (Rule-2 in $d$-separation). The same holds for all paths between $l_{tj}$ and $v_i$. □

**Lemma B.8** (Lemma 4.9 Restated)**.** *Suppose $v_i \to v_j \in G$ and $v_i, v_j \notin B$, and $P_{ij} \subseteq B$ then, $\Pr[v_j \mid v_i, \text{Pa}(v_j), \text{do}(\text{Pa}(v_i) \cup B)] = \Pr[v_j \mid \text{Pa}(v_j), \text{do}(\{v_i\} \cup \text{Pa}(v_i) \cup B)]$ if there is no latent $l_{ij}$ with $v_i \leftarrow l_{ij} \to v_j$.*

*Proof.* Suppose $v_i \to v_j$ in $G$ and there is no latent between $(v_i, v_j)$. Then, we claim that $\Pr[v_j \mid v_i, \text{Pa}(v_j), \text{do}(\text{Pa}(v_i) \cup B)] = \Pr[v_j \mid \text{Pa}(v_j), \text{do}(\{v_i\} \cup \text{Pa}(v_i) \cup B)]$. Let $L_j$ represents all the latent parents of $v_j$. By including $v_i$ in the intervention,

$$\begin{aligned}
&\Pr[v_j \mid \text{Pa}(v_j), \text{do}(\{v_i\} \cup \text{Pa}(v_i) \cup B)] \\
&= \sum_{L_j} \Pr[v_j \mid L_j, \text{Pa}(v_j), \text{do}(\{v_i\} \cup \text{Pa}(v_i) \cup B)] \Pr[L_j \mid \text{Pa}(v_j), \text{do}(\{v_i\} \cup \text{Pa}(v_i) \cup B)]. \\
&= \sum_{L_j} \Pr[v_j \mid L_j, \text{Pa}(v_j), \text{do}(\{v_i\} \cup \text{Pa}(v_i) \cup B)] \Pr[L_j \mid \text{Pa}(v_j), \text{do}(\text{Pa}(v_i) \cup B)]. \qquad (2)
\end{aligned}$$

As the value of $L_j$ is only affected by conditioning on its descendants, and in the interventional distribution $\text{do}(\text{Pa}(v_i))$, $v_i$ is not a descendant of $L_j$, the last statement is true.

Under conditioning on $v_i$

$$\begin{aligned}
&\Pr[v_j \mid v_i, \text{Pa}(v_j), \text{do}(\text{Pa}(v_i) \cup B)] \\
&= \sum_{L_j} \Pr[v_j \mid L_j, v_i, \text{Pa}(v_j), \text{do}(\text{Pa}(v_i) \cup B)] \Pr[L_j \mid v_i, \text{Pa}(v_j), \text{do}(\text{Pa}(v_i) \cup B)] \\
&= \sum_{L_j} \Pr[v_j \mid L_j, v_i, \text{Pa}(v_j), \text{do}(\text{Pa}(v_i) \cup B)] \Pr[L_j \mid \text{Pa}(v_j), \text{do}(\text{Pa}(v_i) \cup B)]. \qquad (3)
\end{aligned}$$

The last statement is true because $L_j \perp\!\!\!\perp v_i \mid \text{Pa}(v_j)$ in the distribution $\text{do}(B \cup \text{Pa}(v_i))$ from Lemma B.7. From the invariance principle (page 24 in [Pearl, 2009], Kocaoglu et al. [2017b]), we have for any variable $v_i$

$$\Pr[v_i \mid \text{Pa}(v_i)] = \Pr[v_i \mid Z, \text{do}(\text{Pa}(v_i) \setminus Z)] \text{ for any } Z \subseteq \text{Pa}(v_i)$$

Applying it to our case we get

$$\Pr[v_j \mid L_j, v_i, \text{Pa}(v_j), \text{do}(\text{Pa}(v_i) \cup B)] = \Pr[v_j \mid L_j, \text{Pa}(v_j), \text{do}(\{v_i\} \cup \text{Pa}(v_i) \cup B)].$$

Putting this together with (2) and (3), we get $\Pr[v_j \mid v_i, \text{Pa}(v_j), \text{do}(\text{Pa}(v_i) \cup B)] = \Pr[v_j \mid \text{Pa}(v_j), \text{do}(\{v_i\} \cup \text{Pa}(v_i) \cup B)]$, if there is no latent $l_{ij}$ with $v_i \leftarrow l_{ij} \rightarrow v_j$. □

**Lemma B.9** (Lemma 4.10 Restated)**.** *Suppose $v_i \rightarrow v_j \in G$ and $v_i, v_j \notin B$, and $P_{ij} \subseteq B$, then, $\Pr[v_j \mid v_i, \text{Pa}(v_j), \text{do}(\text{Pa}(v_i) \cup B)] \neq \Pr[v_j \mid \text{Pa}(v_j), \text{do}(\{v_i\} \cup \text{Pa}(v_i) \cup B)]$ if there is a latent $l_{ij}$ with $v_i \leftarrow l_{ij} \rightarrow v_j$.*

*Proof.* Suppose $v_i \rightarrow v_j$ in $G$ and there is a latent $l_{ij}$ between $(v_i, v_j)$. Then, we claim that $\Pr[v_j \mid v_i, \text{Pa}(v_j), \text{do}(\text{Pa}(v_i) \cup B)] \neq \Pr[v_j \mid \text{Pa}(v_j), \text{do}(\{v_i\} \cup \text{Pa}(v_i) \cup B)]$. Let $L_j$ represents all the latent parents of $v_j$, where $l_{ij} \in L_j$. Therefore, $v_i$ is a descendant of $L_j$. By including $v_i$ in the intervention,

$$\begin{aligned}
&\Pr[v_j \mid v_i, \text{Pa}(v_j), \text{do}(\text{Pa}(v_i) \cup B)] \\
&= \sum_{L_j} \Pr[v_j \mid L_j, \text{Pa}(v_j), \text{do}(\{v_i\} \cup \text{Pa}(v_i) \cup B)] \Pr[L_j \mid \text{Pa}(v_j), \text{do}(\{v_i\} \cup \text{Pa}(v_i) \cup B)] \\
&= \sum_{L_j} \Pr[v_j \mid L_j, \text{Pa}(v_j), \text{do}(\{v_i\} \cup \text{Pa}(v_i) \cup B)] \Pr[L_j \mid \text{Pa}(v_j), \text{do}(\text{Pa}(v_i) \cup B)].
\end{aligned}$$

As the value of $L_j$ is only affected by conditioning on its descendants, and in the interventional distribution $\text{do}(\text{Pa}(v_i))$, $v_i$ is not a descendant of $L_j$, the last statement is true. Under conditioning on $v_i$, we have :

$$\begin{aligned}
&\Pr[v_j \mid v_i, \text{Pa}(v_j), \text{do}(\text{Pa}(v_i) \cup B)] \\
&= \sum_{L_j} \Pr[v_j \mid L_j, v_i, \text{Pa}(v_j), \text{do}(\text{Pa}(v_i) \cup B)] \Pr[L_j \mid v_i, \text{Pa}(v_j), \text{do}(\text{Pa}(v_i) \cup B)] \\
&= \sum_{L_j} \Pr[v_j \mid L_j, v_i, \text{Pa}(v_j), \text{do}(\text{Pa}(v_i) \cup B)] \frac{\Pr[v_i \mid L_j, \text{Pa}(v_j), \text{do}(\text{Pa}(v_i) \cup B)]}{Pr[v_i \mid \text{Pa}(v_j), \text{do}(\text{Pa}(v_i) \cup B)]} \Pr[L_j \mid \text{Pa}(v_j), \text{do}(\text{Pa}(v_i) \cup B)] \\
&= \sum_{L_j} \Pr[v_j \mid L_j, v_i, \text{Pa}(v_j), \text{do}(\text{Pa}(v_i) \cup B)] \frac{\Pr[v_i \mid L_j, \text{Pa}(v_j), \text{do}(\text{Pa}(v_i) \cup B)]}{Pr[v_i \mid \text{Pa}(v_j), \text{do}(\text{Pa}(v_i) \cup B)]} \Pr[L_j \mid \text{Pa}(v_j), \text{do}(\text{Pa}(v_i) \cup B)].
\end{aligned}$$

From the invariance principle (page 24 in [Pearl, 2009], Kocaoglu et al. [2017b]), we have for any variable $v_i$

$$\Pr[v_i \mid \text{Pa}(v_i)] = \Pr[v_i \mid Z, \text{do}(\text{Pa}(v_i) \setminus Z)] \text{ for any } Z \subseteq \text{Pa}(v_i)$$

Applying it to our case we get

$$\Pr[v_j \mid L_j, v_i, \text{Pa}(v_j), \text{do}(\text{Pa}(v_i) \cup B)] = \Pr[v_j \mid L_j, \text{Pa}(v_j), \text{do}(\{v_i\} \cup \text{Pa}(v_i) \cup B)].$$

However, since the numerator of

$$\frac{\Pr[v_i \mid L_j, \text{Pa}(v_j), \text{do}(\text{Pa}(v_i) \cup B)]}{\Pr[v_i \mid \text{Pa}(v_j), \text{do}(\text{Pa}(v_i) \cup B)]}$$

depends on $L_j$ as $v_i$ is a descendant of $l_{ij} \in L_j$, whereas the denominator is not dependent on $L_j$, the ratio is not equal to 1 unless in pathological cases. A similar situation arises in the do-see test analysis for [Kocaoglu et al., 2017b]. Hence, we have $\Pr[v_j \mid v_i, \text{Pa}(v_j), \text{do}(\text{Pa}(v_i) \cup B)] \neq \Pr[v_j \mid \text{Pa}(v_j), \text{do}(\{v_i\} \cup \text{Pa}(v_i) \cup B)]$. □

**Proposition B.10** (Proposition 4.11 Restated)**.** *Let $\mathcal{G}(V \cup L, E \cup E_L)$ be a $\tau$-causal graph with observable graph $G(V, E)$. Algorithm* LatentsWEdges *with $O(n\tau \log n + n \log n)$ many interventions recovers all latents effecting pairs of adjacent nodes in the observable graph $G$ with probability at least $1 - \frac{1}{n^2}$.*

*Proof.* From Lemma B.2, we know that with probability $1 - \frac{1}{n^2}$, for every pair $v_i$ and $v_j$, there exists, with high probability, an intervention $B \in \mathcal{B}_\tau$ such that $v_i \in B, v_j \notin B$ and $P_{ij} \subseteq B$. On this $B$, using Lemmas 4.9 and 4.10, we can identify the latent by using a distribution test on $B \cup Pa(v_i)$ and $B \cup Pa(v_i) \cup \{v_i\}$.

For every variable $v_i \in V$, our algorithm constructs at most $2|\mathcal{B}_\tau|$ many interventions, given by $\text{do}(\{v_i\} \cup \text{Pa}(v_i) \cup B)$ and $\text{do}(\text{Pa}(v_i) \cup B)$ for every $B \in \mathcal{B}_\tau$. Therefore, the total number of interventions used by Algorithm LatentsWEdges is $O(n\tau \log n + n \log n)$. □