LazyIter: A Fast Algorithm for Counting Markov Equivalent DAGs and Designing Experiments

Ali AhmadiTeshnizi [1] **Saber Salehkaleybar** [1] **Negar Kiyavash** [2]

Abstract

The causal relationships among a set of random variables are commonly represented by a Directed Acyclic Graph (DAG), where there is a directed edge from variable X to variable Y if X is a direct cause of Y. From the purely observational data, the true causal graph can be identified up to a Markov Equivalence Class (MEC), which is a set of DAGs with the same conditional independencies between the variables. The size of an MEC is a measure of complexity for recovering the true causal graph by performing interventions. We propose a method for efficient iteration over possible MECs given intervention results. We utilize the proposed method for computing MEC sizes and experiment design in active and passive learning settings. Compared to previous work for computing the size of MEC, our proposed algorithm reduces the time complexity by a factor of $O(n)$ for sparse graphs where n is the number of variables in the system. Additionally, integrating our approach with dynamic programming, we design an optimal algorithm for passive experiment design. Experimental results show that our proposed algorithms for both computing the size of MEC and experiment design outperform the state of the art.

1. Introduction

Directed Acyclic Graphs (DAGs) are the most commonly used structures to represent causal relations between random variables, where a directed edge $X \to Y$ means that variable X is a direct cause of variable Y. Under the faithfulness assumption, conditional independencies between different variables can be inferred from observational data and consequently, the ground truth graph is identified up to Markov Equivalence Class (MEC) (Pearl, 2009; Spirtes et al., 2000). Unique identification of the ground truth DAG among the graphs in an MEC generally requires interventions on variables (Eberhardt & Scheines, 2007). In some scenarios, interventions could be costly (for instance, in biological experiments), and therefore, selecting the optimal intervention target to learn the causal structure is of great interest (Eberhardt et al., 2005; He & Geng, 2008; Eberhardt, 2012; Hauser & Bühlmann, 2014; Shanmugam et al., 2015; Kocaoglu et al., 2017; Ghassami et al., 2018; Lindgren et al., 2018; Agrawal et al., 2019). Several metrics have been suggested in the literature for target selection (He & Geng, 2008; Hauser & Bühlmann, 2014; Ghassami et al., 2018; Agrawal et al., 2019). A good metric for measuring the effectiveness of an intervention is the number of remaining DAGs in an MEC after the intervention (He & Geng, 2008). To use this metric for target selection, we must be able to efficiently count the number of DAGs in an MEC.

Some previous work used the clique tree representation of chordal graphs to divide the causal graph into smaller subgraphs, and perform counting on each subgraph separately (Ghassami et al., 2019; Talvitie & Koivisto, 2019). The main issue with this approach is the dependence on the maximum clique size which can result in $O(n!)$ operations in some cases where n is the number of variables. Ghassami et al. (2019) and Talvitie & Koivisto (2019) used dynamic programming to count DAGs in an MEC. The best time complexity of these approaches is in the order of $O(2^n n^4)$. However, both approaches do not take advantage of sparsity if the graph is sparse. In some other work, the number of edges oriented after an intervention is proposed as the target selection metric (Hauser & Bühlmann, 2014). Hauser & Bühlmann (2014) used the idea of conditioning on different edge orientations for edges connected to a single node to choose the optimal single-node intervention target. The time complexity of proposed algorithm depends on the size of largest clique in MEC which in the worst case is exponential. In Radhakrishnan et al. (2018) they have taken advantage of generating functions and have discussed several families

*Equal contribution [1]Department of Electrical Engineering, Sharif University of Technology, Tehran, Iran [2]School of Management of Technology, Ecole Polytechnique F´ed´erale de Lausanne, Switzerland. Correspondence to: Ali AhmadiTeshnizi <ali.ahmadi215@student.sharif.edu>, Saber Salehkaleybar <saleh@sharif.edu>, Negar Kiyavash <negar.kiyavash@epfl.ch>.

Proceedings of the 37^{th} International Conference on Machine Learning, Vienna, Austria, PMLR 119, 2020.

of graphs. Recently, several work have been proposed for experiment design in passive and active learning settings which use the aforementioned metrics for target selection (Ghassami et al., 2018; Kocaoglu et al., 2017; Agrawal et al., 2019). As such, it is desirable to efficiently compute the MEC size.

In this paper, we propose "LazyIter", a method for efficiently iterating over possible DAGs that we might get after an intervention on a single node. In this method, we start by setting a node as the root of the DAG and finding the corresponding essential graph resulting from intervening on this node. Subsequently, we take advantage of similarities between different candidate graphs to eliminate the recalculation of edge orientations and find other graphs just by reorienting a small subset of the edges. We utilize this method to design algorithms for computing the size of MECs and solving the budgeted experiment design problem in active and passive settings. The main contributions of this paper are the following:

- We propose an algorithm for computing the MEC size of a graph, which improves the time complexity by a factor of $O(n)$ in sparse graphs with respect to previous work (Talvitie & Koivisto, 2019; Ghassami et al., 2019). Our experiments show that the algorithm outperforms previous work in dense graphs too.

- In the active learning setting, we propose two algorithms for designing experiments for both metrics discussed earlier (number of edges and size of MEC). These algorithms are up to $O(n)$ times faster than the previous approaches (He & Geng, 2008; Hauser & Bühlmann, 2014).

- In the passive learning setting, we propose a dynamic programming algorithm for experiment design. To the best of our knowledge, this is the first efficient exact algorithm capable of finding the optimal solution in the passive learning setting. The most closely related work is an approximation algorithm presented in (Ghassami et al., 2019), which has a considerably higher computational complexity.

The paper is organized as follows: First, we discuss the terminology and preliminaries in Section 2. Then, in Section 3, we explain our iteration approach and prove its correctness. In Sections 4 and 5, we apply this approach to design algorithms for computing the MEC size and experiment design and also analyze their complexities. Finally, in Section 6, we demonstrate the efficiency of these algorithms by evaluating them on a diverse set of MECs.

2. Preliminaries

2.1. Graph Terminology

A graph $G(V, E)$ is represented with a set of nodes V and a set of edges E, where each edge is a pair (a, b) such that $a, b \in V$. We say there is an undirected edge between nodes a and b if both $(a, b) \in E, (b, a) \in E$, and say there is a directed edge from a to b if $(a, b) \in E$, and $(b, a) \notin E$. A directed (undirected) edge is denoted with $a \to b \in G$ (or $a - b \in G$). We also use $(a, b) \in E$ and $(a, b) \in G$ subsequently. The set of all directed edges of G is denoted by $Dir(G)$, and the number of directed edges in G is denoted by $|Dir(G)|$. A graph is called undirected (directed) if all of its edges are undirected (directed), and is called partially directed if it has both undirected and directed edges. The induced subgraph $G[S]$ is the graph with node set S and with edge set containing all of the edges in E that have both endpoints in S. Union of graphs $G_1(V, E_1), G_2(V, E_2), ..., G_k(V, E_k)$ with the same set of nodes is defined as $\bigcup_{i=1}^{k} G_i = G(V, \bigcup_{i=1}^{k} E_i)$. For convenience, we may use G and V interchangeably. Two graphs are equal if they have the same set of nodes and the same set of edges.

A path is a sequence of nodes $x_1, x_2, x_3, \ldots, x_k$ such that $\forall 1 \leq i < k : (x_i, x_{i+1}) \in E$. A cycle is a sequence of nodes $x_1, x_2, ..., x_k$ such that $\forall 1 \leq i \leq k : (x_i, x_{i+1}) \in E$ where $x_k = x_1$. A path (cycle) is called directed if all of its edges are directed. Node x is called a descendant of node v if there is directed path from v to x, and there are no directed paths from x to v in the graph. A chain graph is a graph with no directed cycles, and a chain component is a connected component of a chain graph after removing all its directed edges. An undirected graph is chordal if for every cycle of length four or more in it, there exists an edge which is not a part of the cycle but connects two nodes of the cycle to each other.

Let $G(V, E)$ be a partially directed graph. The skeleton of G is an undirected graph that we get by replacing all of the directed edges in E by undirected edges. We say node $v \in V$ is separated from node u by set $T \subset V$ if there is no path from v to u in the skeleton of $G[V \backslash T]$, and we call T, a (v, u)-separator in G [1]. We denote parents, children, and neighbors of node $v \in V$ by $pa_G(v)$, $ch_G(v)$, and $ne_G(v)$, respectively. A perfect elimination ordering (PEO) in a graph G is an ordering of its vertices such that for every vertex v, v and its neighbors prior to it in the ordering form a clique. A graph is chordal if and only if it has a perfect elimination ordering (Fulkerson & Gross, 1965).

[1] Please note that the definition of separator here is different from the definition of d-separation in causal Bayesian networks.

2.2. Causal Model

A causal DAG D is a DAG with variables $V_1, \cdots, V_n$ where there is a directed edge from V_i to V_j if V_i is a direct cause of V_j. A joint probability distribution P over these variable satisfies Markov property with respect to D if any variable is independent of its non-descendants given its parents. A Markov Equivalence Class (MEC) is a set of DAGs with the same Markov property. Verma & Pearl (1992) showed that the graphs in an MEC have the same skeleton and the same set of v-structures (induced subgraphs of the form $a \to b \leftarrow c$). The essential graph of D is defined as a partially directed graph $G(V, E)$ where E is the union of all edge sets of the DAGs in the same MEC as D. An essential graph is necessarily a chain graph with chordal chain components (Hauser & Bühlmann, 2012). Verma & Pearl (1992) showed that having observational data, essential graph is obtainable by applying four rules (called "Meek" rules) consecutively on the graph, until no more rules are applicable. A *valid* orientation of edges of a chain component is an orientation in which no cycles and no v-structures are formed. An intervention target $I \subseteq V$ is a set of nodes which we intervene on simultaneously. An intervention family $\mathcal{I}$ is a set of intervention targets. Intervention graph $D^{(I)}$ is the DAG we get from D after removing all edges directed towards nodes in I.

Definition 1. *For a set of intervention targets I, two DAGs D_1 and D_2 are called $\mathcal{I}$-Markov Equivalent (denoted with $D_1 \sim_{\mathcal{I}} D_2$) if they are statistically indistinguishable under intervention targets in $\mathcal{I}$.*

Hauser & Bühlmann (2012) proved that two DAGs D_1 and D_2 are $\mathcal{I}$-Markov equivalent if and only if D_1 and D_2 have the same set of v-structures, and $D_1^{(I)}$ and $D_2^{(I)}$ have the same skeleton for every $I \in \mathcal{I} \cup \{\emptyset\}$.

The $\mathcal{I}$-essential graph $\mathcal{E}_{\mathcal{I}}(D)$ of a DAG $D(V, E)$ is a partially directed graph with the node set V and the edge set equal to the union of all edge sets of the DAGs which are $\mathcal{I}$-Markov equivalent with D. $\mathcal{I}$-MEC is defined as the set of all DAGs that are $\mathcal{I}$-Markov equivalent.

Definition 2. *For undirected chordal chain graph (UCCG) $G(V, E)$ and intervention family $\mathcal{I}$, the intervention result space is defined as:*

$$\mathcal{IR}_{\mathcal{I}}(G) = \{\mathcal{E}_{\mathcal{I}}(D) : D \in \boldsymbol{D}(G)\},$$

where $\boldsymbol{D}(G)$ denotes the set of all DAGs inside MEC corresponding to G.

We use $MEC(\mathcal{E}_{\mathcal{I}}(D))$ to show the set of all DAGs in an $\mathcal{I}$-MEC. Throughout the paper, we assume UCCGs are chain components of sufficient and faithful essential graphs.

3. LazyIter

We first propose a method to select the best single-node intervention target in an essential graph. As $\mathcal{I}$-essential graph $\mathcal{E}_I(D)$ on DAG D is a chain graph with undirected chordal chain components, it could be shown that knowing orientations of edges inside a component does not provide any information about the orientation of edges in other components (Hauser & Bühlmann, 2012). Hauser & Bühlmann (2014) showed that each chain component can be treated as an observational essential graph when it comes to intervening on the nodes (i.e., $\mathbf{D}(G)$ is the same set of DAGs, whether G is an observational essential graph or it is a chain component of an $\mathcal{I}$-essential graph). Consequently, we can restrict our attention to UCCGs. He & Geng (2008) presented a method to find $\mathcal{I}$-essential graph from intervention results when the intervention target is the root, which takes $O(n\Delta^2)$ operations. We will use this method in the next sections as a subroutine for computing the size of $\mathcal{I}$-essential graph whenever conditioning on edge orientations results in the intervention target becoming root.

Let $G(V, E)$ be a UCCG and $\{v\}$ be a single-node intervention target on it. After the intervention, we will obtain an $\mathcal{I}$-essential graph $\mathcal{E}_{\{\{v\}\}}(D) \in \mathcal{IR}_{\{\{v\}\}}(G)$ based on the ground truth DAG D. The following theorem allows us to use parent set of v for uniquely representing the resulting $\mathcal{I}$-essential graph:

Proposition 1. *Let $G(V, E)$ be a UCCG, $D \in MEC(G)$ be a DAG, and $v \in V$ be an arbitrary node. Then each $\mathcal{I}$-essential graph $\mathcal{E}_{\{\{v\}\}}(D)$ could be uniquely determined given the parent set of v in D, and there is a one-to-one correspondence between sets $\{P \subseteq ne_G(v) : P$ is a clique$\}$ and $\mathcal{IR}_{\{\{v\}\}}(G)$.*

The proof of this proposition as well as all other proofs are available in the supplementary material. The theorem suggests a way for iterating over $\mathcal{IR}_{\{\{v\}\}}(G)$: Iterate over all cliques in the neighborhood of v and set each clique as the parent set of v and then apply Meek rules to orient as many edges as possible (Hauser & Bühlmann, 2014). According to Proposition 1, the essential graph $\mathcal{E}_{\{\{v\}\}}(D)$ can be determined by $pa_D(v)$. Thus, we use the notation of $\mathcal{P}_v^P(G)$ to point to $\mathcal{E}_{\{\{v\}\}}(D)$ where $D \in \mathbf{D}(G)$ is a DAG such that $pa_D(v) = P$.

Let $R(V, E') = \mathcal{P}_v^P(G)$ be a possible single-node-intervention result on a UCCG $G(V, E)$. Setting aside the nodes in P, we divide the other nodes of R into three distinct groups C_R, A_R, and D_R. C_R is the set of children of v, and $C_R \cup P = ne_G(v)$. A_R is the set of all nodes which are separated from v by P, and D_R is the set of all other nodes. We have:

$$A_R = \{a \in V \backslash ne_G(v) : P \text{ is an } (a, v)\text{-separator in } G\}$$
$$D_R = V \backslash (A_R \cup C_R \cup P).$$

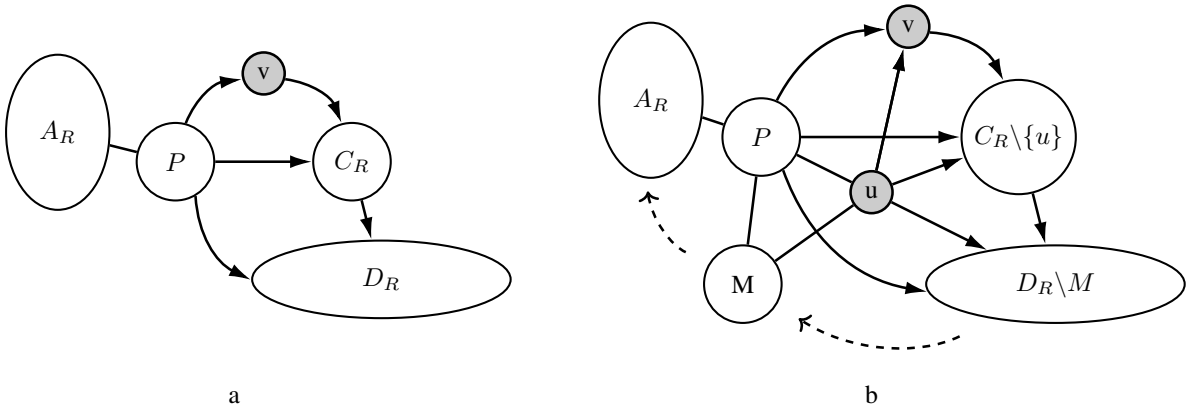

Figure 1. a) A separation of $R = \mathcal{P}_v^P$ into different sets. b) Constructing $R' = \mathcal{P}_v^{P\cup\{u\}}$ from R by moving u from C_R to P and M from D_R to A_R. An arrow between two sets/nodes means that any edge between them is directed in the corresponding direction. A straight line between two sets/nodes, means that any edge between them is undirected. Dashed lines show how nodes are moved from D_R to A_R upon the construction.

See Figure 1a for an illustration. The following theorem states several key properties of the three proposed node groups.

Theorem 1. *Let $R = \mathcal{P}_v^P(G)$ be an $\mathcal{I}$-essential graph on a UCCG $G(V, E)$. The following statements hold:*

- *There are no edges in G connecting a node in A_R to a node in $C_R \cup D_R \cup \{v\}$.*
- *Every edge (a, b) in R where $a \in P$ and $b \in C_R$ is directed as $a \to b$.*
- *Every edge (a, b) in R where $a \in C_R \cup P$ and $b \in D_R$ is directed as $a \to b$.*
- *All of the edges in $R[A_R \cup P]$ are undirected.*

Now we show that direction of many edges in $\mathcal{P}_v^P(G)$ stay intact when we change the parent set P slightly, and therefore if we already know direction of edges in an $\mathcal{I}$-essential graph, we can find the direction of edges in other $\mathcal{I}$-essential graphs by reorienting just a small fraction of the edges.

Assume we are given R and we want to find $R' = \mathcal{P}_v^{P\cup\{u\}}(G)$ where $u \in C_R$, and $G[P \cup \{u\}]$ is a clique. Note that the skeleton of both R' and R is G, and they only differ in the direction of some edges.

It is easy to see $A_R \subseteq A_{R'}$ as every node which is separated from v by P is also separated from v by $P \cup \{u\}$. Moreover we know that $A_R \cup D_R = A_{R'} \cup D_{R'}$ as both of them represent the set of nodes in $V\backslash(ne_G(v) \cup \{v\})$. Consequently, we have $D_{R'} \subseteq D_R$ and $D_R\backslash D_{R'} = A_{R'}\backslash A_R = M$. We construct R' from R by moving u from children to the parents and M from D_R to A_R, and then reorienting some specific edges as we explain. We have:

$$M = \{a \in D_R : \ P \cup \{u\} \text{ is an } (a, v)\text{-separator in } G\}$$

$$C_{R'} = C_R\backslash\{u\}, A_{R'} = A_R \cup M, D_{R'} = D_R\backslash M.$$

Applying the first statement of Theorem 1 to R', we conclude that there are no edges between M and $D_R\backslash M$ in G (as $M \subseteq A_{R'}$ and $D_R\backslash M = D_{R'}$). The third statement of Theorem 1 implies that in R', any edge between $(P \cup \{u\}) \cup C_{R'} = P \cup C_R$ and $D_{R'} = D_R\backslash M$ is directed towards the node in $D_R\backslash M$. The same thing is true in R, as we have $D_R\backslash M \subseteq D_R$. This means any edge in $R[D_R\backslash M]$ which is directed by applying Meek rules, can be similarly directed in $R'[D_R\backslash M]$, and therefore $R'[D_R\backslash M] = R[D_R\backslash M]$. Moreover, we can say that $R'[A_R \cup P] = R[A_R \cup P]$ because both are undirected graphs on the same skeleton. Using the fourth statement of Theorem 1, we can infer that all of edges in $R'[M\cup\{u\}]$ are undirected, as $M \cup \{u\} \subseteq A_{R'}$. The same is true for edges with one end in $M \cup \{u\}$ and the other end in P. Finally, by the second statement of Theorem 1, all of the edges in $R'[C_R]$ which are connected to u are directed away from u. This means we can find the orientation of edges in R' by executing the following three steps on R:

1. Obtain the set M by finding nodes in $G[V\backslash A_R]$ which are separated from v by $(P \cup \{u\})$. If we execute a breadth first search (BFS) in $G[V\backslash(P \cup \{u\})]$ with v as root, the nodes which are not observed in the BFS constitute $A_{R'}$. By removing nodes of A_R from $A_{R'}$ we will get the set M. This will take $O(n + m) = O(n + n\Delta) = O(n\Delta)$ operations, where n, m, and Δ are the number of variables, the number of the edges, and the maximum degree of the graph respectively.

Algorithm 1 LazyIter

Input: UCCG $G(V, E)$, Node $v \in V$
Output: $\mathcal{IR}_{\{\{v\}\}}(G)$
$\mathcal{L} \leftarrow \emptyset$
Find $\mathcal{P}_v^{\emptyset}(G)$ by setting v as the root of G and orienting as much edges as possible.
$Iter(\mathcal{P}_v^{\emptyset}(G), v)$
return $\mathcal{L}$

---

function $Iter(\mathcal{P}_v^{P}(G), v)$
Add $\mathcal{P}_v^{P}(G)$ to $\mathcal{L}$.
for $u \in C_R$ **do**
  if u is connected to all nodes in P **then**
    $NewGraph \longleftarrow \mathcal{P}_v^{P}(G)$
    $M \longleftarrow$ Set of all nodes separated from v by $P \cup \{u\}$ in $G[V \backslash A_{\mathcal{P}_v^{P}(G)}]$
    Change direction of $v \rightarrow u$ to $u \rightarrow v$ in $NewGraph$
    In $NewGraph$, make all edges with both ends in $M \cup \{u\}$ undirected
    In $NewGraph$, make all edges connecting a node in P to a node in $M \cup \{u\}$ undirected
    In $NewGraph$, direct all edges between u and a node $c \in C_R$ as $u \rightarrow c$
    In $NewGraph[C_{\mathcal{P}_v^{P}(G)}]$, orient edges using Meek rules until no more undirected edges are orientable
    $Iter(NewGraph, v)$
  end if
end for

2. Remove the directions of all edges inside $R[M \cup \{u\}]$ and all edges between $M \cup \{u\}$ and P. This could be done in $O(n\Delta)$ operations.

3. Direct all edges $u - x$ in $R[C_R]$ as $u \rightarrow x$, and apply Meek rules on $R[C_R]$ to find $R'[C_R \backslash \{u\}]$. This could be done in $O(\Delta^3)$ operations (He et al., 2015), as we have $|C_R| \leq \Delta$.

The procedure for finding $IR_{\{\{v\}\}}(G)$ is given in Algorithm 1. In order to find $IR_{\{\{v\}\}}(G)$, first we obtain $\mathcal{P}_v^{\emptyset}(G)$ by setting v as the root of the graph and directing edges based on Meek rules in $O(n\Delta^2)$ operations (He et al., 2015). Then we initiate $\mathcal{L}$ as an empty set and call $LazyIter(\mathcal{P}_v^{\emptyset}(G), v)$ which will add all desired $\mathcal{I}$-essential graphs to set $\mathcal{L}$ (for finding $\mathcal{P}_v^{\emptyset}(G)$ we can use the algorithm presented in (He et al., 2015) which needs $O(n\Delta^2)$ operations). The algorithm will call itself recursively $O(2^{\Delta})$ times, and the three mentioned operations are executed in each call in order to find the new $\mathcal{I}-$essential graph corresponding to the new parent set. When the execution is completed, $\mathcal{L}$ will contain the list of all obtainable $\mathcal{I}$-essential graphs. The complexity of the algorithm is $O(n\Delta^2 + 2^{\Delta}(n\Delta + \Delta^3)) = O(2^{\Delta}(n\Delta + \Delta^3))$. The first step is executed in line 12 of Algorithm 1, the second step is executed in lines 14 and 15, and the last step is executed in lines 16 and 17.

4. Computing size of MEC

We count the number of DAGs inside an MEC by partitioning them into $\mathcal{I}$-Markov equivalence classes.

Lemma 1. *Let $G(V, E)$ be a UCCG and $\mathcal{I}$ be an arbitrary intervention family. Then we have:*

$$|MEC(G)| = \sum_{R \in \mathcal{IR}_{\mathcal{I}}(G)} \Big[\prod_{C \in \mathcal{C}(R)} |MEC(C)| \Big],$$

where $\mathcal{C}(R)$ denotes the set of all chain components of R.

Assume we are given a UCCG $G(V, E)$ and want to calculate $|MEC(G)|$. We first choose an arbitrary node $v \in V$, set $\mathcal{I} = \{\{v\}\}$, and use $LazyIter$ to find all of the $\mathcal{I}$-essential graphs. Then for each of them, we calculate the number of DAGs inside its corresponding $\mathcal{I}$-MEC by multiplying size of its chain components. As each chain component of an $\mathcal{I}$-essential graph is a UCCG (Hauser & Bühlmann, 2014), Lemma 1 is applicable on it and we could do the calculation recursively. Finally, we sum up all these values to get $|MEC(G)|$.

We take advantage of dynamic programming to eliminate repetitive calculations. Ghassami et al. (2019); Talvitie & Koivisto (2019) used a similar idea for observational essential graphs, which we extended to interventional cases.

The algorithm is presented in Algorithm 2. Every time $Count(S)$ is called, it will take $O(1)$ operations if $DP[S]$ is already calculated. Otherwise, it calls $LazyIter$ once which takes $O(2^{\Delta}(n\Delta + \Delta^3))$ operations, and executes the

Algorithm 2 LazyCount

Input: UCCG $G(V, E)$
Output: $|MEC(G)|$
$CountDP[] \leftarrow$ A storage indexed on $S \subseteq V$ and initiated by 1 if $|S| = 1$ and NULL otherwise.
return $Count(V)$

---

function $Count(S)$
if $CountDP[S]$ is not $NULL$ **then**
  return $DP[S]$
end if
$CountDP[S] \leftarrow 0$
$v \leftarrow$ an arbitrary node in S
$\mathcal{L} \leftarrow LazyIter(G[S], v)$
for $R \in \mathcal{L}$ **do**
  $num \leftarrow 1$
  for $C(S', E') \in \mathcal{C}(R)$ **do**
    $num \leftarrow num \times Count(S')$
  end for
  $CountDP[S] \leftarrow CountDP[S] + num$
end for
return $CountDP[S]$

Algorithm 3 Active Learning by Minimizing $\mathcal{I}$-MEC size

Input: UCCG $G(V, E)$
Output: A single-node intervention target $\{v_{opt}\}$
$CountDP[] \leftarrow$ A storage indexed on $S \subseteq V$ and initiated by 1 if $|S| = 1$ and NULL otherwise.
$s_{opt} \leftarrow 0$
$v_{opt} \leftarrow NULL$
for $v \in V$ **do**
  $\mathcal{L} \leftarrow LazyIter(G, v)$
  $s_v \leftarrow 0$
  for $R \in \mathcal{L}$ **do**
    $mecsize \leftarrow 1$
    for $C(V', E') \in \mathcal{C}(R)$ **do**
      $mecsize \leftarrow mecsize \times Count(V')$
    end for
    $s_v \leftarrow max(s_v, mecsize)$
  end for
  if $s_v < s_{opt}$ **then**
    $s_{opt} \leftarrow s_v$
    $v_{opt} \leftarrow v$
  end if
end for
return v_{opt}

two for-loops. The outer for-loop is executed at most 2^{Δ} times, and the inner for-loop is executed at most n times. Calculation of $\mathcal{C}(\mathcal{E})$ could also be done in $O(n\Delta)$ steps. After these calculations, $DP[S]$ will be saved and there is no need to calculate it in later calls. On the other hand, there are at most 2^n values for index of DP, and therefore the time complexity of Algorithm 2 is:

$$O\Big(2^n\big(2^{\Delta}(n\Delta+\Delta^3)+2^{\Delta}(n+n\Delta)\big)\Big) = O(2^n 2^{\Delta}(n\Delta+\Delta^3)).$$

5. Experiment Design

Assume we want to find the best intervention target $I \subseteq V$ in UCCG $G(V, E)$. For experiment design, given an objective function, we need to compare the efficiency of different intervention targets based on it. A common objective function is the size of $\mathcal{I}$-essential graph obtained after intervention (Ghassami et al., 2019). The smaller the class is, the more information we have gained from the intervention. If we consider the worst-case setting, we have:

$$I_{opt} = \arg\min_{I \subseteq V} \Big(\max_{R \in \mathcal{IR}_{\{I\}}(G)} |MEC(R)| \Big). \qquad (1)$$

Another objective function used in previous work is the number of directed edges after an intervention (Ghassami et al., 2018; Hauser & Bühlmann, 2014):

$$I_{opt} = \arg\max_{I \subseteq V} \Big(\min_{R \in \mathcal{IR}_{\{I\}}(G)} |Dir(R)| \Big), \qquad (2)$$

We solve the experiment design problem for both of these objective functions, in both active and passive learning settings.

5.1. Active Learning

In the active learning, the information obtained from the former interventions can be used to choose the next targets. Similar to the approach taken in Hauser & Bühlmann (2014), we aim to find the best single-node intervention target in each learning step. We take advantage of $LazyIter$ and $LazyCount$ for this purpose.

Let $G(V, E)$ be a UCCG. Considering objective function (1), we want to find a node v such that intervening on it, minimizes the size of the resulting $\mathcal{I}$-MEC. We first use $LazyIter$ to find the set of all $\mathcal{I}$-essential graphs for different single-node intervention targets. Then, for each $\mathcal{I}$-essential graph $\mathcal{P}_v^P(G)$, we obtain the size of its corresponding $\mathcal{I}$-MEC by multiplying sizes of its chain components. Finally, we use these values to find the optimal intervention target. The description of this algorithm is presented in Algorithm 3. The procedure is almost the same for objective function (2). We just need to calculate number of directed edges for each $\mathcal{I}$-essential graph, instead of calculating its $\mathcal{I}$-MEC size.

All of the operations in Algorithm 3 could be divided to two parts:

- Calculating the values of $CountDP[]$ using function $Count()$, which takes at most $O(2^n 2^\Delta(n\Delta + \Delta^3))$ operations.
- Iterating over the three for-loops (taking n, 2^Δ, and n steps respectively), calling $LazyIter$ (taking $O(2^\Delta(n\Delta + \Delta^3))$ operations), and calculating $\mathcal{C}(R)$ (taking $O(n\Delta)$ operations). All of these steps together need $O(n2^\Delta(n\Delta + \Delta^3))$ operations.

Therefore Algorithm 3 calculates the MEC size in at most $O(2^n 2^\Delta(n\Delta + \Delta^3)) + O(n2^\Delta(n\Delta + \Delta^3)) = O(2^n 2^\Delta(n\Delta + \Delta^3))$ operations. If we want to find the best target with respect to objective function (2), there is no need to calculate $CountDP[]$, but all other operations should be executed similarly. Consequently, the time complexity in this case would be $O(n2^\Delta(n\Delta + \Delta^3))$.

5.2. Passive Learning

Let $G(V, E)$ be a UCCG, where each node $v \in V$ is assigned a cost c_v. We aim to find a set of k single-node interventions, and therefore our intervention family is of the form $\mathcal{I} = \{\{v_1\}, \{v_2\}, ..., \{v_k\}\}$, similar to the model considered in Ghassami et al. (2018). Using the following lemma, we break the problem down to smaller subproblems and take advantage of dynamic programming:

Lemma 2. *Let $G(V, E)$ be a UCCG, $\mathcal{I} = \{\{v_1\}, \{v_2\}, ..., \{v_k\}\}$ an intervention family, D the ground truth DAG of G, and for each chain component $C \in \mathcal{C}(\mathcal{E}_{\{\{v_1\}\}}(G))$, $\{v_1^C, v_2^C, ..., v_{m_C}^C\} \subseteq V$ be the subset of intervention targets which are inside C. Then we have:*

$$Dir(\mathcal{E}_{\{\{v_1\},\{v_2\},...,\{v_k\}\}}(D)) = Dir(\mathcal{E}_{\{\{v_1\}\}}(D)) \cup \mathcal{Z},$$

where

$$\mathcal{Z} = \bigcup_{C \in \mathcal{C}(\mathcal{E}_{\{\{v_1\}\}}(D))} Dir(\mathcal{E}_{\{\{v_1^C\},\{v_2^C\},...,\{v_{m_C}^C\}\}}(D[C])).$$

Assume we want to find the optimum intervention target with respect to objective function (2). For any $T, S \subseteq V$ where $T = \{v_1, v_2, ..., v_t\}$ and $T \subseteq S$, we define $DP[S][T]$ as follows:

$$DP[S][T] = \min_{D \in \mathbf{D}(G)} |Dir(\mathcal{E}_{\{\{v_1\},\{v_2\},...,\{v_t\}\}}(D[S])|. \tag{3}$$

Proposition 2. *The following equation holds for DP function (3):*

$$DP[S][T] = \min_{R \in \mathcal{IR}_{\{\{v_1\}\}}(G[S])} \Big\{ |Dir\Big(R\Big)| + \sum_{C \in \mathcal{C}(R)} DP\Big[C\Big]\Big[T \cap C\Big] \Big\}. \tag{4}$$

Algorithm 4 Passive Learning by Maximizing Number of Oriented Edges

Input: UCCG $G(V, E)$, budget b, intervention cost for each node $v \in V$ as $cost_v$
Output: A single-node intervention target $\{v_{opt}\}$
$DP[S][T] \leftarrow$ A storage indexed on $S \subseteq V$ and $T \subseteq S$, and initiated by 0 if $|S| = 1$ and NULL otherwise.
$best \leftarrow \emptyset$
for $T \subseteq V$ **do**
  if $\sum_{x \in T} cost_x \leq budget$ **then**
    if $Calculate(V, best) \leq Calculate(V, T)$ **then**
      $best \leftarrow T$
    end if
  end if
end for
return $best$

function $Calculate(S, T)$
if $DP[S][T]$ is not NULL **then**
  return $DP[S][T]$
end if
$DP[S][T] \leftarrow \infty$
$v \leftarrow$ an arbitrary member of T
$\mathcal{L} \leftarrow LazyIter(G, v)$
for $R \in \mathcal{L}$ **do**
  $num \leftarrow |Dir(R)|$
  for $C(S', E') \in \mathcal{C}(R)$ **do**
    $num \leftarrow num + Calculate(S', T \cap S')$
  end for
  $DP[S][T] \leftarrow min(DP[S][T], num)$
end for
return $DP[S][T]$

This proposition suggests that we could select an arbitrary intervention target, iterate over all $\mathcal{I}$-essential graphs in its intervention result space, and find number of directed edge in each case using already-calculated DP values. After finding all $DP[V][T]$ values, we can choose the one which has a cost less than our budget and maximizes number of directed edges. For optimization with respect to objective function (1), we can define $DP[S][T]$ as the maximum size of $\mathcal{I}$-MEC obtained from $G[S]$ after intervening on nodes in T. With the similar arguments, we can show that if we substitute $|Dir(R)|$ with $|MEC(R)|$ in equation (4), the resulting equation holds for this new DP array.

The number of DP elements is 3^n, as each node is either in T, or in $S \backslash T$, or in $V \backslash S$. For calculation of each DP value, $LazyIter$ is called once and then two for-loops are executed, iterating for 2^Δ and n steps respectively. Hence, Algorithm 5.2 finds the best passive intervention target with respect to objective function (2) in $O(3^n 2^\Delta(n\Delta + \Delta^3))$ operations.

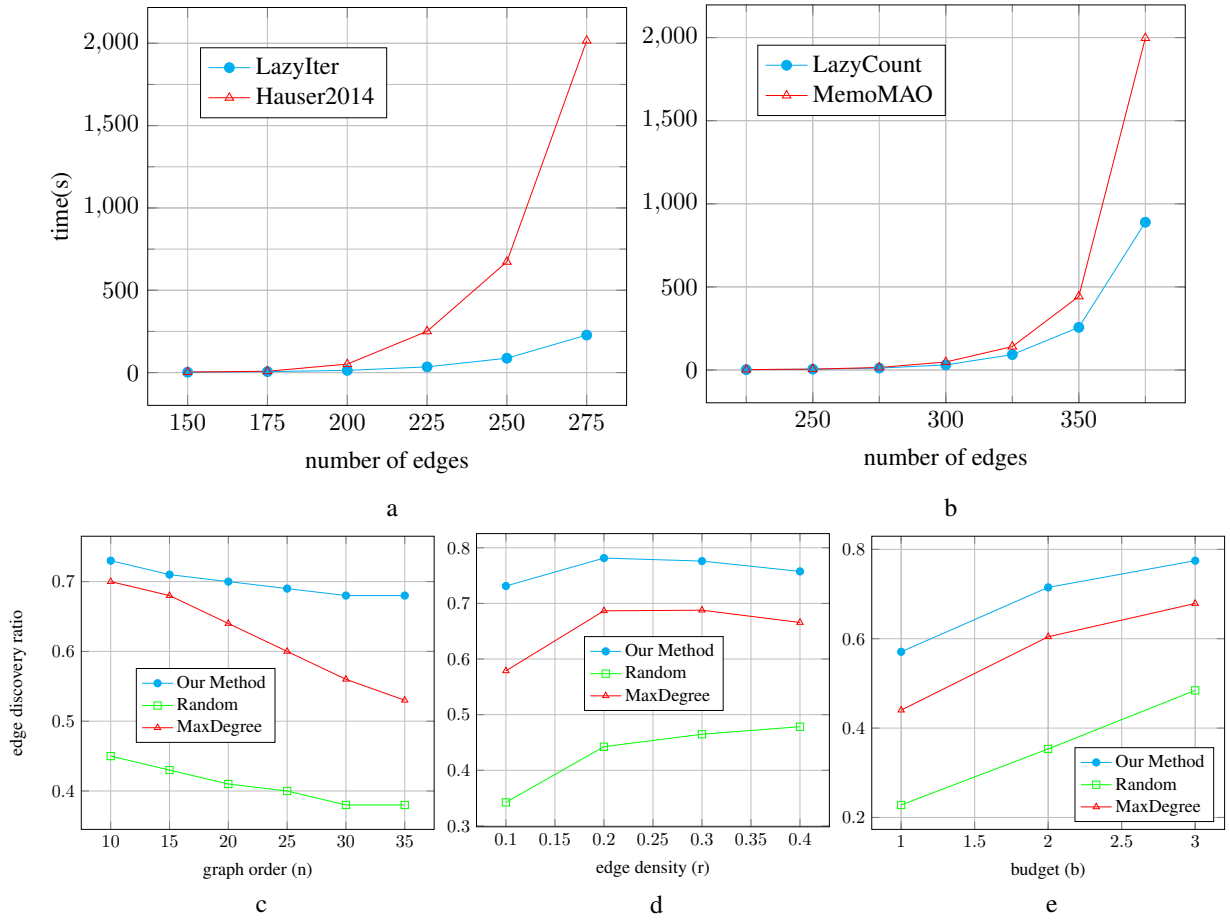

Figure 2. (a) Comparison between execution times of LazyIter and algorithm in Hauser & Bühlmann (2014) versus number of edges for graphs with 30 nodes. (b) Comparison between execution times of LazyCount and MemoMAO (Talvitie & Koivisto, 2019) versus number of edges for graphs with 30 nodes. Comparison between edge discovery ratio versus (c) graph order for $b = 2$ and $r = 0.4$, (d) edge density for $n = 35$ and $b = 3$, (e) budget for $n = 40$ and $r = 0.3$

6. Experimental Results

We compared $LazyIter$ and $LazyCount$ against previous work. The performance of our active learning algorithms depend on these two routines. Our DP-based passive learning algorithm is the first exact algorithm for worst-case experiment design, so we compared it with Random and MaxDegree heuristics. The only related previous work Ghassami et al. (2019) is an approximation designed for the average-case passive learning. Their algorithm has a time complexity of $O(kNn^{(\Delta+1)})$ (where N is the number of sampled DAGs and k is the budget), and is considerably more computationally expensive than our algorithm. However, the results are not comparable as their algorithm does not solve the problem in the worst-case setting. For each test, we generated 100 graphs using the method presented in He et al. (2015) and calculated the average test results on them. As we can see in Figure 2 (a), $LazyIter$ outperforms (Hauser & Bühlmann, 2014) in all cases, especially when the graph is dense. We also tested $LazyCount$ against $MemoMAO$, which is the state-of-the-art MEC size calculation algorithm (Talvitie & Koivisto, 2019). Even though the difference in execution times is not considerable for sparse graphs, our algorithm performs much better for dense graphs, as seen in Figure 2 (b). The main reason for this is that $LazyCount$ requires fewer DP values in its execution. Figures 2 (c), (d), and (e) present the discovered edge ratio (the number of edges whose orientations are inferred from experiments to the number of edges in the graph) of the passive learning algorithm versus different graph orders, edge densities (ratio of the number of edges to the maximum possible number of edges), and budgets (number of interevetions), respectively. As the graph order increases, finding the optimal target becomes harder, and therefore the difference between our algorithm and the heuristics becomes more considerable.

7. Conclusion

We proposed a new method to iterate efficiently over possible $\mathcal{I}$-essential graphs and utilized it to design algorithms for computing MEC size and experiment design for active and passive learning settings. Experimental results showed that the proposed algorithms outperform other related works in terms of time complexity. As a direction of future research, it would be interesting to extend to the proposed algorithms for other objective functions in designing experiments, such as average number of oriented edges. Moreover, one can work on designing algorithms in the passive learning setting where we can intervene on multiple variables in each experiment.

References

Agrawal, R., Squires, C., Yang, K., Shanmugam, K., and Uhler, C. Abcd-strategy: Budgeted experimental design for targeted causal structure discovery. *arXiv preprint arXiv:1902.10347*, 2019.

Eberhardt, F. Almost optimal intervention sets for causal discovery. *arXiv preprint arXiv:1206.3250*, 2012.

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

Eberhardt, F., Glymour, C., and Scheines, R. On the number of experiments sufficient and in the worst case necessary to identify all causal relations among n variables. pp. 178–184, 2005.

Fulkerson, D. R. and Gross, O. A. Incidence matrices and interval graphs. *Pacific J. Math.*, 15(3):835–855, 1965.

Ghassami, A., Salehkaleybar, S., Kiyavash, N., and Bareinboim, E. Budgeted experiment design for causal structure learning. In *International Conference on Machine Learning*, pp. 1724–1733, 2018.

Ghassami, A., Salehkaleybar, S., Kiyavash, N., and Zhang, K. Counting and sampling from markov equivalent dags using clique trees. *Proceedings of the AAAI Conference on Artificial Intelligence*, 33:3664–3671, Jul 2019.

Hauser, A. and Bühlmann, P. Characterization and greedy learning of interventional markov equivalence classes of directed acyclic graphs. *J. Mach. Learn. Res.*, 13(1): 2409–2464, August 2012. ISSN 1532-4435.

Hauser, A. and Bühlmann, P. Two optimal strategies for active learning of causal models from interventional data. *International Journal of Approximate Reasoning*, 55(4): 926–939, Jun 2014. ISSN 0888-613X.

He, Y., Jia, J., and Yu, B. Counting and exploring sizes of markov equivalence classes of directed acyclic graphs. *The Journal of Machine Learning Research*, 16(1):2589–2609, 2015.

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

Kocaoglu, M., Shanmugam, K., and Bareinboim, E. Experimental design for learning causal graphs with latent variables. In *Advances in Neural Information Processing Systems*, pp. 7021–7031, 2017.

Lindgren, E., Kocaoglu, M., Dimakis, A. G., and Vishwanath, S. Experimental design for cost-aware learning of causal graphs. In *Advances in Neural Information Processing Systems*, pp. 5279–5289, 2018.

Pearl, J. *Causality: Models, Reasoning and Inference*. Cambridge University Press, USA, 2nd edition, 2009. ISBN 052189560X.

Radhakrishnan, A., Solus, L., and Uhler, C. Counting markov equivalence classes for dag models on trees. *Discrete Applied Mathematics*, 244:170–185, Jul 2018. ISSN 0166-218X. doi: 10.1016/j.dam.2018.03.015. URL http://dx.doi.org/10.1016/j.dam.2018.03.015.

Shanmugam, K., Kocaoglu, M., Dimakis, A. G., and Vishwanath, S. Learning causal graphs with small interventions. In *Advances in Neural Information Processing Systems*, pp. 3195–3203, 2015.

Spirtes, P., Glymour, C., and Scheines, R. *Causation, Prediction, and Search*. Springer, 2000.

Talvitie, T. and Koivisto, M. Counting and sampling markov equivalent directed acyclic graphs. In *The Thirty-Third AAAI Conference on Artificial Intelligence*, pp. 7984–7991. AAAI Press, 2019.

Verma, T. and Pearl, J. An algorithm for deciding if a set of observed independencies has a causal explanation. *Uncertainty in Artificial Intelligence*, pp. 323–330, 1992.