Improving Constraint-Based Causal Discovery from Moralized Graphs

Ashlynn N Fuccello
Department of Interdisciplinary Studies
Hendrix College
Conway, AR 72032
FuccelloAN@hendrix.edu

Daniel Y Yuan
Department of Computational and Systems Biology
University of Pittsburgh
Pittsburgh, PA 15213
day44@pitt.edu

Panayiotis V Benos*
Department of Computer Science
Department of Computational and Systems Biology
University of Pittsburgh
Pittsburgh, PA 15213
benos@pitt.edu

Vineet K Raghu†
Department of Computer Science
University of Pittsburgh
Pittsburgh, PA 15213
vkr8@pitt.edu

Abstract

There are many causal discovery methods that can learn a directed graph from mixed, observational data, including the popular PC-Stable. However, PC-Stable learns the directed graph by performing conditional independence tests on all variables. New more efficient methods for constraint-based causal learning use intermediate steps (e.g., learning of the moralized graph first) that can reduce the overall execution time. We present new causal discovery algorithms, *Triangle* and *Connected Neighbors* (CN), that improve the efficiency of PC-Stable. These algorithms first calculate the moralized graph from mixed data. Then, Triangle only looks at a subset of edges in the graph for removal before orienting the edges. CN extends the Triangle method via a novel, efficient method to choose which conditional independence tests to perform. The CN approach increases the efficiency of PC-Stable by reducing the number of conditional independence tests performed. The most significant runtime improvements are seen with CN in dense graphs, with minimal losses in adjacency and orientation precision and recall.

1 Introduction

1.1 Background

Causal graphical models (CGMs)[8, 18] are a key tool for modelling systems across many fields [1, 10, 12, 7]. These models provide a straightforward but mathematically robust representation of the system that can be interpreted as probabilistic information and causal inferences about the data.

A typical CGM consists of a directed acyclic graph (DAG) or a collection of DAGs (a Partially Directed Acyclic Graph, PDAG, or Pattern [19]) whose directed edges ($X \rightarrow Y$) represent a cause-effect relationship between the parent X and child Y. This generative model explains how each variable takes on its value, and can predict the result of interventions on individual variables. CGMs can determine the true causal relationships through observational data without experimentation.

*https://www.benoslab.pitt.edu/

†Currently at Massachusetts General Hospital, Boston, MA

34th Conference on Neural Information Processing Systems (NeurIPS 2020), Vancouver, Canada.

Methods for learning DAGs fall into two groups: score-based and constraint-based [20]. The most popular score-based approach is the Greedy Equivalence Search (GES) [4] and its parallelized version, the Fast Greedy Equivalence Search (FGES) [14]. FGES uses a likelihood-based score (e.g., Bayesian Information Criterion) to test whether single edge insertions, deletions, or directionality changes improve the fit of the model, subject to a penalty on the number of parameters. Constraint-based approaches start with a fully connected graph and perform conditional independence tests to remove non-causal edges between variables and orient the rest. Constraint-based algorithms, such as PC[19] and its modifications (e.g., PC-Stable [5], Conservative-PC [13], are well studied and have been shown to produce accurate results [6]. However, they generally do not scale efficiently. Here, we focus on improving runtime efficiency of constraint-based causal discovery.

PC-Stable has exponential time complexity to the number of variables in the graph. In mixed datasets with continuous and discrete variables, this growth pattern can be a critical bottleneck. A promising solution is to efficiently learn an undirected (moralized) graph of conditional dependence relations. Mixed graphical models (MGMs) calculate an undirected graph from mixed data [9, 17]. The moralized graph, in principle, contains a superset of the edges of the true graph (shielded colliders). The extra edges can then be removed by local causal searches. As we will show, PC-Stable is effective at removing these edges but is unnecessarily inefficient when the moralized, undirected graph is known. This becomes increasingly problematic if the underlying causal graph is dense.

This paper addresses this problem by proposing a new constraint-based method for causal discovery, Connected Neighbors (CN), that uses the aforementioned properties of moralized graphs to reduce the number of independence tests needed to learn the directed graph. The efficiency and accuracy of this algorithm is compared to that of PC-Stable as well as FGES with a score suitable for mixed datasets (Degenerate Gaussian Score [2]). We test these methods on datasets of varying sample size and density to characterize the relative performance of these approaches.

1.2 Preliminaries

Definition 1.2.1 The **adjacency set** for a node X in a graph $G = (\mathbf{V}, \mathbf{E})$ denoted as $Adj(X, G)$ is the set of nodes $\mathbf{Y}$ such that $\forall Y \in \mathbf{Y}$, $Y \neq X$ and the edge $(X - Y) \in E$

Definition 1.2.2 A **first neighbor** or **neighbor** of a node X in a graph $G = (\mathbf{V}, \mathbf{E})$ is any variable Y such that $Y \in Adj(X, G)$

Definition 1.2.3 A **parent** of a node X in a DAG $G = (\mathbf{V}, \mathbf{E})$ is any variable V such that $V \to X \in \mathbf{E}$. The set of all parents of X in G is denoted $Pa(X, G)$

Definition 1.2.4 A **moralized graph** of a DAG $G = (\mathbf{V}, \mathbf{E})$ is an undirected graph $H = (\mathbf{V}, \mathbf{E}^*)$, where
(1) $\forall X, Y \in \mathbf{V}, X \in Adj(Y, G)$ implies $X \in Adj(Y, H)$
(2) $\forall X, Y, Z \in \mathbf{V}$, If $X \to Y \in \mathbf{E}$ and $Z \to Y \in \mathbf{E}$ and $X \notin Adj(Z, G)$ then $X \in Adj(Z, H)$

The following definitions are necessary to understand the proof of correctness of the connected neighbors algorithm. We refer the reader to [19] for more details.

Definition 1.2.5 An **active path** in a DAG $G = (\mathbf{V}, \mathbf{E})$ between variables X and Y given set $\mathbf{S}$ is a set of variables $V_1, \ldots, V_N$ such that

1. $X = V_1$ and $Y = V_N$
2. $\forall i, V_i \in Adj(V_{i+1}, G)$
3. $\forall i > 1$ If $V_{i-1} \to V_i$ and $V_{i+1} \to V_i$, then $V_i \in \mathbf{S}$ or a descendant of $V_i \in \mathbf{S}$ (all colliders on the path are in $\mathbf{S}$)
4. $\forall i > 1$ If $V_i \to V_{i-1}$ or $V_i \to V_{i+1}$, then $V_i \notin \mathbf{S}$ (all non-colliders on the path are not in $\mathbf{S}$)

Definition 1.2.6 Variable X is **d-separated** from variable Y given set $\mathbf{S}$ in a DAG G if there are no active paths between X and Y given $\mathbf{S}$. Variables that are not d-separated are said to be **d-connected**.

PC-Stable The PC algorithm assumes that the data were generated according to a DAG (ground-truth). The original PC algorithm starts with a fully connected undirected graph. First, it iterates through each edge (X-Y) in the graph and runs an unconditional independence test ($X \perp\!\!\!\perp Y \mid \emptyset$) on the two connected nodes. If the nodes are found to be independent, then the edge is immediately removed. After it finishes removing all edges between variables which are independent conditioned on the empty set ($d = 0$), d is incremented until $d >$ the size of the full set of first neighbors. At each d, it checks for conditional independence between all remaining pairs of adjacent variables given all subsets of size d of the first neighbors of X and/or Y. When an edge (X-Y) is removed, the conditioning set used to determine its independence is stored (Separating set or Sepset(X,Y)). Now the graph contains only those edges present in the data generating DAG. The final step of PC is to orient the direction of the remaining edges. First, for each edge that was removed, such as X–Y in Figure 1, any variable Z, such that Z is adjacent to X and Y and is not in Sepset(X,Y), must be a collider. This means that the direction of both edges must point to Z (X -> Z and Y -> Z) as in Figure 1. Then, if orienting an undirected edge would result in a new collider, the edge is oriented in the reverse direction. Finally, if X -> Y, and Y -> Z, and X – Z, then the edge X-Z is oriented as X -> Z to prevent the introduction of a cycle (a path from one variable back to itself)[19].

One of the problems with PC is that since edges are removed immediately after a conditional independence is detected, the output graph depends on the order the algorithm tests the edges [5]. To correct this, PC-Stable removes the edges altogether after all tests of a given conditioning set size (e.g., $d = 1$) are performed. This makes the algorithm independent of the order the edges are tested.

Mixed Graphical Models In this work we use a mixed graphical model (MGM) to efficiently learn the moralized graph [17, 9]. MGM parameterizes the joint distribution of p continuous and q categorical variables (Equation 1). β_{st} represents the linear interaction between continuous variables s and t. ρ_{sj} is a vector of parameters relating categorical variable j to continuous variable s, with one parameter for each category of j. Finally, ϕ_{rj} is a matrix representing the interactions between the categories of categorical variables r and j. This model generalizes two graphical models: multivariate Gaussian for continuous data, and pairwise Markov Random Field for categorical data.

$$p(x, y; \theta) \propto exp\Big(\sum_{s=1}^{p} \sum_{t=1}^{p} -\frac{1}{2} \beta_{st} x_s x_t + \sum_{s=1}^{p} \alpha_s x_s + \sum_{s=1}^{p} \sum_{j=1}^{q} \rho_{sj}(y_j) x_s + \sum_{j=1}^{q} \sum_{r=1}^{q} \phi_{rj}(y_r, y_j) \Big) \quad (1)$$

The parameters are optimized by minimizing the negative log-pseudolikelihood ($\widetilde{l}(\Theta)$)[3], the product of conditional distributions, which are 1) multivariate Gaussian for continuous variables with a mean given by a linear regression on all other variables and 2) Multinomial distribution for discrete variables with probabilities given by a multi-class logistic regression on all other variables. To prevent overfitting, the model includes sparsity penalties (Equation 2).The graph is then formed from edges with non-zero coefficients.

$$minimize\, l_\lambda(\Theta) = \widetilde{l}(\Theta) + \lambda_{CC} \sum_{s=1}^{p} \sum_{t=1}^{s-1} |\beta_{st}| + \lambda_{CD} \sum_{s=1}^{p} \sum_{j=1}^{q} ||\rho_{sj}||_2 + \lambda_{DD} \sum_{j=1}^{q} \sum_{r=1}^{j-1} ||\phi_{rj}||_F \quad (2)$$

In recent years, efficient methods have been developed for learning causal graphs, in which an undirected graph is learned first and used as a skeleton on which conditional independence tests are run locally. This reduces the overall running time from a globally exponential problem to a locally exponential problem, compared to the original fully connected starting graph of PC.

2 Methods

2.1 Algorithms

We propose two new algorithms to improve performance of PC-Stable after the moralized graph has been calculated: (1) Triangle, (2) Connected Neighbors.

Triangle The undirected "moralized" graph consists of two types of edges: 1) those in the ground truth DAG and 2) edges between unconnected variables with a common child (colliders). Learning the correct DAG means removing type-2 edges and hence properly orienting the colliders. These edges

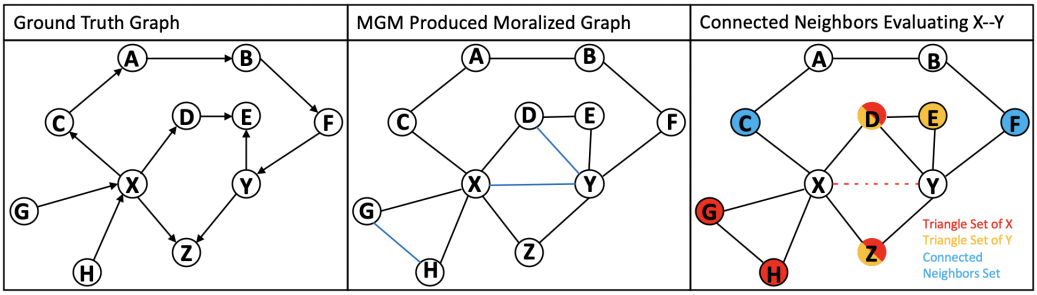

Figure 1: Example of MGM with Connected Neighbors. Ground-truth data generating DAG (left). The data set generated by this ground truth graph is then run through MGM which generates the moralized graph (center) with added edges due to moralization in blue (X–Y, G–H, D–Y). Connected Neighbors and Triangle Sets of X and Y (right).

form "Triangles" in the moralized graph (e.g., X-Z-Y in Figure 1). Thus, our algorithm, *Triangle*, initially determines such triangles by evaluating every edge (X-Y) in the graph and identifying their common first neighbors (Z). Every shared first neighbor indicates the three edges involved in a triangle (X-Y, X-Z, and Y-Z). Then, instead of iterating through all edges in the undirected graph when performing conditional independence tests (as in PC-Stable), only edges that are part of triangles are tested. Once one edge in a triangle is removed the other two are not tested unless they are still involved in other triangles.

Connected Neighbors The second algorithm proposed, *Connected Neighbors* (CN) (Algorithm 1), leverages the fact that conditioning sets that may remove the edge X-Y need only contain neighbors of X and Y that lie on an active path between X and Y. CN builds two sets for each variable of each edge: 1) a triangle set Tri(X), Tri(Y) (Algorithm S1) and 2) a connected neighbors set CN(X,Y), CN(Y,X) (Algorithm S2). Tri(X) includes all nodes involved in a triangle with X. CN(X,Y) is all connected neighbors of X given Y (Definition 1.2). For example, Tri(X) for edge X-Y from Figure 1 would contain G, H, D, and Z while Tri(Y) would contain D, E, and Z. For the same edge, CN(X,Y) contains only C and CN(Y,X) contains only F. Just like Triangle, the method iterates over all edges involved in triangles and performs independence tests conditioned on the union of CN(X,Y) and subsets of Tri(X) of increasing size and the union of CN(Y,X) and subsets of Tri(Y) of increasing size. CN constrains the number of conditional independence tests by limiting the subset search to only those variables involved in Triangles.

Both of these algorithms are "stable" in that the output is independent of the order the edges are encountered and they follow the same procedure as PC-Stable for orientation.

2.2 Connected Neighbors Proof of Correctness

In this section, we show that the *Connected Neighbors* algorithm learns the correct causal DAG when the moralized graph is correctly estimated and a conditional independence oracle is available.

Definition 2.2.1 The triangle set Tri for a variable X in an undirected graph H (denoted $Tri\,(X, H)$) is the set of variables $\mathbf{Z}$ s.t. $\forall Z \in \mathbf{Z}$, $\exists Y \neq X$ such that, $Z \in Adj\,(Y, H)$, $Y \in Adj\,(X, H)$, and $Z \in Adj\,(X, H)$

Definition 2.2.2 The connected neighbors $CN(X, Y)$ of variable X given variable Y in an undirected moralized graph H is the set of variables $\mathbf{Z}$ that satisfy the following conditions

1. $\forall Z \in \mathbf{Z}\ Z \notin Tri\,(X, H)$
2. $\forall Z \in \mathbf{Z}$, $Z \in Adj\,(X, H)$
3. $\forall Z \in \mathbf{Z}$, Z is d-connected to Y given X

Lemma 2.2.1 Let G be a causal DAG and H be the undirected "moralized" version. If $X \in Adj\,(Y, H)$ and $Adj\,(X, H) \cap Adj\,(Y, H) = \emptyset$, then $X \in Adj\,(Y, G)$

Algorithm 1: Connected Neighbors

```
Input  : An undirected moralized graph G = (V, E)
Output : DAG D
buildTriangleSet                          #Algorithm S1;
buildConnectedNeighborsSet                #Algorithm S2;
while depth d < 1000 and the last iteration ran an independence test do
    for edge e in triangleEdges do
        x = node1 of e; y = node2 of e;
        conditioningSet = ∅;
        for each possible combination of d elements in Tri(x), c do
            conditioningSet = c ∪ CN(x, y);
            if x,y is independent conditioned on the conditioningSet then
                Save the conditioningSet as the Separation Set;
                Mark e for removal;
                break;
            end
        end
        if e has not been marked for removal then
            for each possible combination of d elements in Tri(y), c do
                conditioningSet = c ∪ CN(y, x);
                if x,y is independent conditioned on the conditioningSet then
                    Mark e for removal;
                    break;
                end
            end
        end
    end
    Remove all edges marked for removal from G and triangleEdges;
    Update triangleSets, Tri, and CN accordingly;
    d += 1
end
Run PC-Stable orientation steps ;
```

Proof. This is true by the connection between undirected and directed causal graphs. All edges in the undirected graph are present in the directed graph except for those caused by moralization. Since X and Y share no adjacent variables, they cannot be parents of a collider in G, and so the edge cannot be due to moralization. □

Now, we have shown that the only independence tests that must be performed are on edges in triangles. Next, we characterize which conditioning sets must be checked. We prove that our method identifies a separating set for all pairs of variables connected by moralization.

Theorem 2.2.1 If X and Y are d-separated given some set **S** but adjacent in a moralized graph H then they are d-separated given some $\{\mathbf{T} \subseteq Tri\,(X)\} \cup CN\,(X, Y)$ *or* $\{\mathbf{T} \subseteq Tri\,(Y)\} \cup CN\,(Y, X)$

Proof. Let G be the unknown DAG corresponding to the moralized graph H. First, we will show that we will eventually find a choice for $\mathbf{T}$ that blocks all active paths between X and Y, then we will show that adding the remaining connected neighbors does not activate new paths. For the first part of the proof we consider two possibilities: X is or is not a descendant of Y in G.

Case 1: X **is not a descendant of** Y **in G** If X is not a descendant of Y in G, then X is d-separated from Y given $Pa(Y, G)$. If $|Pa(Y, G)| > 1$, then $Pa(Y, G) \subseteq Tri(Y, H)$ (because they all collide at Y). So we can choose $\mathbf{T}$ to be equal to $Pa(Y, G)$. We address the case where $|Pa(Y, G)| = 1$. We show that either this single parent is in $CN(Y, X)$ or X and Y are d-separated given the empty set. We do this via the following lemma.

Lemma 2.2.2 Let P be the single parent of Y in G. If X and Y are adjacent in the moralized graph, X is not a descendant of Y, Y has one parent, and X and Y are d-separated given some set $\mathbf{S}$, then

1. If $P \in CN(Y, X)$, then X and Y are d-separated by P
2. If $P \notin CN(Y, X)$, then X and Y are d-separated by $\emptyset$

Proof of Lemma 2.2.2. Assume X and Y are not d-separated by P and $P \in CN(Y, X)$. Since $P \in CN(Y, X)$, P is d-connected to X given Y. Since X and Y are not d-separated by P, there must be a path between X and Y where all colliders on the path are ancestors of P in G.

Either this path goes through P or a child of Y. If the path goes through a child of Y, then it must contain a collider (since X is not a descendant of Y). Call the first collider on this path $C1 \rightarrow C2 \leftarrow C3$. Since the path is active conditioned on P, $C2$ must be an ancestor of P. But then this creates a cycle $Y \rightarrow \ldots C1 \rightarrow C2 \rightarrow \ldots \rightarrow P \rightarrow Y$. So this path does not go through a child of Y. Then this path must go through P; however, this path must be inactive conditioned on P because P cannot be a collider on this path (as it is a parent of Y).

Assume $P \notin CN(Y, X)$ and X and Y are d-connected given $\emptyset$. If X and Y are d-connected given $\emptyset$, then there is a path between them that contains no colliders. Assume that this path goes through P. Then, there must be a path between P and X with no colliders. Then, $P \in CN(Y, X$. Assume that this path goes through a child C. Then, X must be a descendant of Y in G (which is a contradiction) Or this path must have a collider.

Case 2: X is a descendant of Y in G If X is a descendant of Y, we can apply the same argument (but flip X and Y), since X is d-separated from Y given $Pa(X, G)$.

□

Now, we know that by conditioning on the connected neighbor set and some subset of the Triangle set we will eventually condition on all parents of Y, and d-separate X and Y. But how do we know that we do not condition on a connected neighbor that activates a path to Y?

Lemma 2.2.3 If X and Y are d-separated given $Pa\,(Y, G)$ and X is not a descendant of Y in G, then they are d-separated given $Pa\,(Y, G) \cup CN\,(Y, X)$

Proof of Lemma 2.2.3. Note that $CN\,(Y, X)$ can only consist of children of Y in G and not spouses, because all spouses would be involved in triangles in the moralized graph (and thus be a part of $Tri\,(Y, H)$). First, we note that all paths $Y - Pa\,(Y, G) - \ldots - X$ are blocked because $Pa\,(Y, G)$ are conditioned upon and are non-colliders (otherwise they would be in $Tri(Y, H)$). Thus, we only consider paths through the children of Y in G.

Assume a path $Y \rightarrow C_1 - \ldots - X$ is active conditioned on $Pa\,(Y, G) \cup CN\,(Y, X)$. Assume C_1 is a collider on this path. C_1 is not conditioned on since $C_1 \in Tri(Y, H)$, so this path is inactive unless a descendant of C_1 is conditioned on. We know that any parent of Y cannot be a descendant of C_1 due to acyclicity. So a child of Y (call it C^*) that is in $CN(Y, X)$ must be a descendant of C_1. This is impossible because then there would be a collider at C^* and it would be included in $Tri\,(Y, H)$.

Now, assume C_1 is not a collider on this path. There must be at least one collider on this path, otherwise X is a descendant of Y in G. Consider the first collider on this path from Y to X $A_1 \rightarrow A_2 \leftarrow A_3$. A_2 nor any descendant can be a parent of Y to avoid acyclicity. Thus, A_2 or some descendant must be a connected neighbor child of Y. Again, this would result in a collider at that child, preventing it from being in $CN\,(Y, X)$, as it would be in $Tri(Y, H)$ instead.

So this path must not exist, which implies that there are no active paths between X and Y. □

By the previous lemma, we know that our algorithm eventually conditions on the set $Pa\,(Y, G) \cup CN\,(Y, X)$. Thus, our algorithm removes any edge that has some separating set given a conditional independence oracle and a correct moralized graph. □

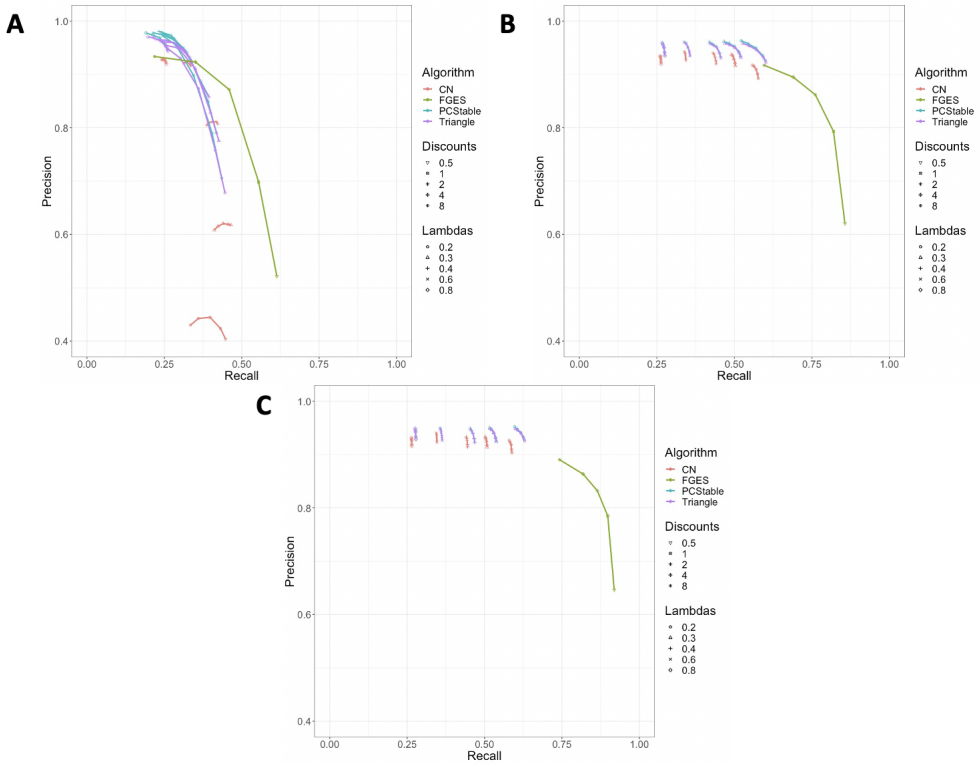

Figure 2: Adjacency Precision-Recall curves where each point represents the average precision/recall values across all 50 graphs at each λ (0.2, 0.3, 0.4, 0.6, 0.8) alpha (0.0001, 0.001, 0.01, 0.05, 0.1) pair for PC-Stable, CN, and Triangle vs each penalty discount (0.5, 1, 2, 4, 8) for FGES. All results from 100 variable ground truth DAGs with node degree fixed at 4 and sample size varied (a) 100 samples, (b) 500 samples, (c) 5000 samples.

2.3 Experiments

For simulation studies, 50 DAGs were randomly generated for various combinations of experimental parameters using the TETRAD software [15]. These parameters included number of nodes in the graph: $(50, 100, 500)$ and average degree: $(2, 4, 6)$. Each graph consisted of 50% continuous and 50% discrete variables (with 2-5 categories, chosen uniformly at random for each variable). Each node had a max in and out degree of 10. 5000 samples were generated for each graph using the Lee and Hastie simulator [9] and then subsampled to create 500 and 100 sample datasets.

MGM[16] ran on each simulated dataset to estimate the undirected, moralized graph. Penalty parameters (λ) varied: $(0.2, 0.3, 0.4, 0.6, 0.8)$ with equal values for each edge type. PC-Stable, Triangle, and CN started with the estimated moralized graph (instead of the full graph) and ran across multiple independence test thresholds (α) $(1 * 10^{-4}, 1 * 10^{-3}, 0.01, 0.05, 0.1)$. FGES started with the empty graph and ran over multiple penalty discounts $(0.5, 1, 2, 4, 8)$. A likelihood ratio independence test for mixed data was used for all experimental runs [16]. Runtime (in ms) for each algorithm was recorded, and the Markov Equivalence Class of the output graph was compared to Markov Equivalence Class of the ground truth graphs to determine accuracy. While the analysis could have been done using the known moralized graph instead of MGM, the use of MGM provides a more practical evaluation as the moralized graph must be learned from data. All simulations were run on Intel Xeon Gold processors with 16 cores and 32 GB of RAM.

3 Results

3.1 Accuracy

Various metrics were used to evaluate the accuracy of the graphs produced by each algorithm compared to the Markov Equivalence Class of the ground truth DAG. These include: adjacency and orientation precision and recall, and the structural hamming distance (SHD) [21]. SHD identifies the number of changes (edge flips, insertions, or deletions) that are needed to transform the learned DAG to the data-generating DAG.

In terms of SHD, all constraint-based algorithms performed similarly at higher sample sizes across different numbers of variables and node degrees. CN performed worse in small sample sizes and low λ (Figures S1-S14). FGES consistently performed the same or worse than the constraint-based algorithms in terms of SHD (Figures S1-S14). CN showed a loss in orientation precision at smaller λ (denser graphs) compared to Triangle and PC-Stable, which perform similarly to each other (Figures S15-S28). Overall, the orientation precision and recall of CN across α and λ values is more consistent than that of PC-Stable and Triangle. All of the constraint-based algorithms perform worse than FGES in orientation recall as sample size increases (Figures S15-S28).

However, there were larger differences between algorithms for adjacency precision and recall. When comparing the constraint-based methods to FGES, we found that the constraint-based methods generally have a higher precision and lower recall (Figure 2, S29-S39), as seen by others [11]. As sample size increases, the constraint-based algorithms perform worse in recall than FGES (Figure 2). While the methods use different sets of parameters (α independence test threshold for constraint-based methods; penalty discount for FGES), generally the precision/recall curve across multiple parameters are much tighter for constraint-based methods. Therefore, these constraint based methods are insensitive to parameter selection at higher sample sizes (Figure 2 B,C). Additionally, CN is even less sensitive to changes in α especially at lower λ values. When comparing the constraint-based methods, PC-Stable and Triangle have practically identical recall and precision that follow the same trends across different parameter sets. CN follows similar trends with no change in recall but a loss of precision that decreases as sample size increases (Figure 2). These trends in adjacency precision and recall hold across varying graph sizes and densities (Figures S29-S39).

3.2 Efficiency

Runtime (wall clock time in ms) was measured for each algorithm, excluding the time for the estimation of the moralized undirected graph, in order to focus on the difference between the efficiency of the algorithms. FGES was excluded from runtime experiments as it is parallelized and the sparsity parameters are not directly comparable. PC-Stable and Triangle performed similarly and consistently across all experimental parameters (number of variables, sample size, density) (S40-S48). CN is slower than other constraint-based methods at low sample sizes but faster at high sample sizes (Figure 3 A,C,B). Additionally, the improvement of CN over PC-Stable and Triangle is more pronounced as graph density increases (i.e. lower λ for MGM, higher degree) (Figure 3 D,C,E).

4 Discussion

We introduce two new methods for constraint-based causal discovery, Triangle and CN, that are theoretically correct when the moralized graph is known. We find a minor runtime improvement by limiting independence tests to edges involved in triangles. Triangle and CN have similar accuracy to PC-Stable, as expected, since additional tests by PC-Stable should not remove other edges.

The CN algorithm has initial runtime overhead as it runs independence tests to build the connected neighbor conditioning sets. For small/sparse graphs, the added preparation time upfront is significant in comparison to the relatively small number of independence tests that need to be performed to remove edges. In denser graphs, this initial cost becomes insignificant compared to the time saved during the later phase of the algorithm.

We notice a consistent, small loss of precision in CN compared to PC-Stable or Triangle. This may be due to the fact that CN needs to perform order 2 conditional independence tests to identify the connected neighbor sets, and higher order conditional independence tests in the second phase since it always includes the Connected Neighbors in the conditioning set. These high-order tests may

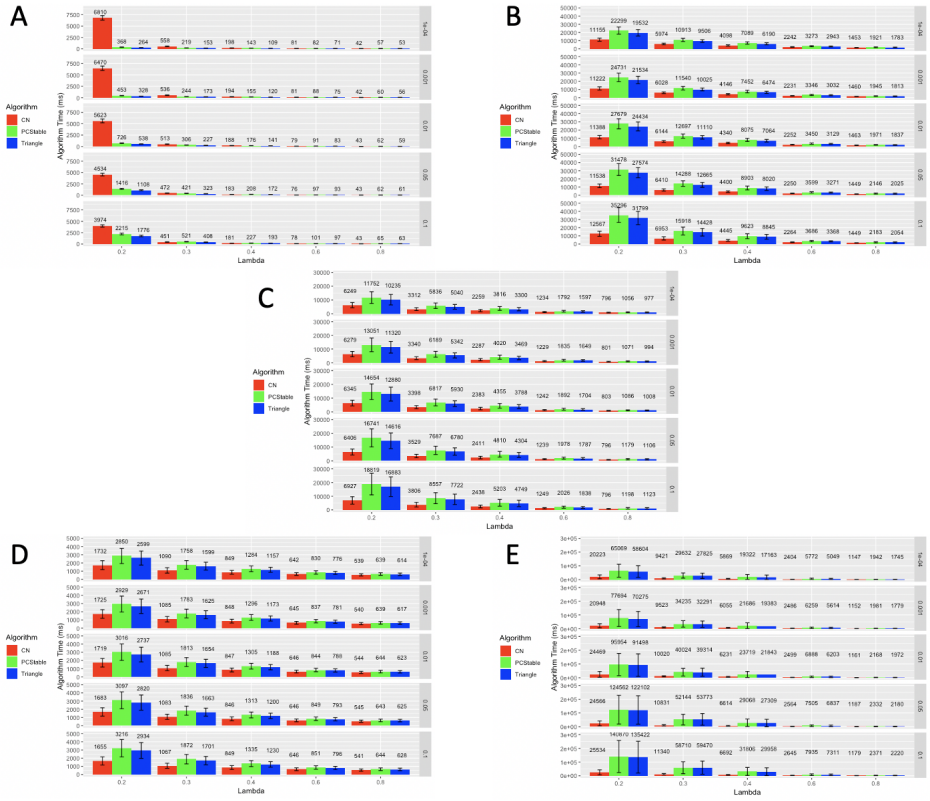

Figure 3: Runtime results with 100 variable ground truth DAGs. Node degree was fixed at 4 and sample size varied from (a) 100 samples, (c) 500 samples, (b) 5000 samples. In addition, sample size was held fixed at 500 and node degree varied from (d) 2, (c) 4, (e) 6.

be underpowered in small sample sizes, resulting in smaller connected neighbor sets than would be expected theoretically. Since recall appears to be unaffected, the main issue is most likely in the construction of the connected neighbor sets and not the search procedure for a separating set. Nevertheless, the loss of precision is small and for large datasets (high sample size or graph degree), CN can act as an efficient alternative to PC-Stable.

Future directions include the application and evaluation of this approach to other constraint-based methods such as other variants of PC, CPC, FCI, etc. Also, parallelizing CN could show even larger improvement in algorithm efficiency. Lastly, further reductions in conditioning sets or edges may be possible when starting with a moralized graph to improve runtime without losing precision.

Broader Impact

This paper presents a theoretical approach to causal discovery. The approach proposed improves on an existing algorithm, and therefore does not generally have major benefits, disadvantages, or biases not already assumed by the original method. Generally, causal discovery algorithms may benefit researchers in all fields by prioritizing promising hypotheses. However, failure of these algorithms may result in loss of efficiency, and waste of resources. Biases in the data may be reflected in learnt causal graphs and must be dealt with accordingly. Subsequent prediction models based on causal graphs may result in biased decision making if not carefully considered.

Acknowledgements

This work was supported by NSF grant 1659611 (Pitt TECBio REU to ANF) and NIH grants T32CA082084 (to DYY, VKR), U01HL137159, and R01LM012087 (to PVB).

References

[1] Irina Abecassis, Andrew J Sedgewick, Marjorie Romkes, Shama Buch, Tomoko Nukui, Maria G Kapetanaki, Andreas Vogt, John M Kirkwood, Panayiotis V Benos, and Hussein Tawbi. Parp1 rs1805407 increases sensitivity to parp1 inhibitors in cancer cells suggesting an improved therapeutic strategy. *Scientific reports*, 9(1):1–9, 2019.

[2] Cooper GF. Andrews B, Ramsey J. Learning high-dimensional directed acyclic graphs with mixed data-types. *Proceedings of machine learning research*, 104:4–21, 2019.

[3] Julian Besag. Statistical analysis of non-lattice data. *Journal of the Royal Statistical Society, Series D*, pages 179–195, 1975.

[4] David Maxwell Chickering. Optimal structure identification with greedy search. *Journal of machine learning research*, 3(Nov):507–554, 2002.

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

[6] Clark Glymour, Kun Zhang, and Peter Spirtes. Review of causal discovery methods based on graphical models. *Frontiers in genetics*, 10:524, 2019.

[7] Georgios D Kitsios, Adam Fitch, Dimitris V Manatakis, Sarah Rapport, Kelvin Li, Shulin Qin, Joseph Huwe, Yingze Zhang, John Evankovich, William Bain, et al. Respiratory microbiome profiling for etiologic diagnosis of pneumonia in mechanically ventilated patients. *Frontiers in microbiology*, 9:1413, 2018.

[8] Daphne Koller and Nir Friedman. *Probabilistic graphical models: principles and techniques*. MIT press, 2009.

[9] Jason D Lee and Trevor J Hastie. Learning the Structure of Mixed Graphical Models. *Journal of Computational and Graphical Statistics*, 24(1):230–253, 2015.

[10] Vineet K Raghu, Colin H Beckwitt, Katsuhiko Warita, Alan Wells, Panayiotis V Benos, and Zoltán N Oltvai. Biomarker identification for statin sensitivity of cancer cell lines. *Biochemical and biophysical research communications*, 495(1):659–665, 2018.

[11] Vineet K Raghu, Allen Poon, and Panayiotis V Benos. Evaluation of causal structure learning methods on mixed data types. *Proceedings of machine learning research*, 92:48, 2018.

[12] Vineet K Raghu, Wei Zhao, Jiantao Pu, Joseph K Leader, Renwei Wang, James Herman, Jian-Min Yuan, Panayiotis V Benos, and David O Wilson. Feasibility of lung cancer prediction from low-dose ct scan and smoking factors using causal models. *Thorax*, 74(7):643–649, 2019.

[13] Joseph Ramsey, Jiji Zhang, and Peter L. Spirtes. Adjacency-Faithfulness and Conservative Causal Inference. *Proceedings of the Twenty-Second Conference on Uncertainty in Artificial Intelligence*, pages 401–408, 2006.

[14] Joseph D. Ramsey. Scaling up greedy equivalence search for continuous variables. *arXiv*, 2015.

[15] Joseph D Ramsey, Kun Zhang, Madelyn Glymour, Ruben Sanchez Romero, Biwei Huang, Imme Ebert-Uphoff, Savini Samarasinghe, Elizabeth A Barnes, and Clark Glymour. Tetrad—a toolbox for causal discovery. In *Proceedings of the 8th International Workshop in Climate Informatics*.

[16] Andrew J Sedgewick, Kristina Buschur, Ivy Shi, Joseph D Ramsey, Vineet K Raghu, Dimitris V Manatakis, Yingze Zhang, Jessica Bon, Divay Chandra, Chad Karoleski, et al. Mixed graphical models for integrative causal analysis with application to chronic lung disease diagnosis and prognosis. *Bioinformatics*, 35(7):1204–1212, 2019.

[17] Andrew J. Sedgewick, Ivy Shi, Rory M. Donovan, and Panayiotis V. Benos. Learning mixed graphical models with separate sparsity parameters and stability-based model selection. *BMC Bioinformatics*, 17(S5):175, 2016.

[18] Peter Spirtes. Introduction to Causal Inference. *Journal of Machine Learning Research*, pages 1–3, 2011.

[19] Peter Spirtes, Clark N Glymour, and Richard Scheines. *Causation, prediction, and search*. MIT press, 2000.

[20] Peter Spirtes and Kun Zhang. Causal discovery and inference: concepts and recent methodological advances. In *Applied informatics*, volume 3, page 3. SpringerOpen, 2016.

[21] Ioannis Tsamardinos, Laura E Brown, and Constantin F Aliferis. The max-min hill-climbing bayesian network structure learning algorithm. *Machine learning*, 65(1):31–78, 2006.