UNIVERSITY OF CALIFORNIA

Los Angeles

A thesis submitted in partial satisfaction

of the requirements for the degree

Master of Science in Statistics

by

Steven Lucas Turnbull

2024

ABSTRACT OF THE THESIS

by

Steven Lucas Turnbull
Master of Science in Statistics
University of California, Los Angeles, 2024
Professor Qing Zhou, Chair

Often casual Bayesian networks contain variables that we cannot observe or measure. If we wish to find an optimal graphical representation, typical structure learning cannot account for the existence of latent confounders. This results in possible bias and misleading results. Here we consider structure learning for a class of mixed graphs, bow-free acyclic mixed directed graphs, ADMGs, that offer a representation of directed acyclic graphs, DAGs, with latent confounders. Our approach uses a hybrid of constraint-based MAG learning and score-based optimization to find ADMGs with optimal BIC. We use constraint-based MAG learning to restrict our search space and then find the optimal ADMG representation using a score-based optimization algorithm. We also present a simulation and ADMG comparison framework that shows the empirical effectiveness of our alogrithm.

The thesis of Steven Lucas Turnbull is approved.

Oscar Hernan Madrid Padilla

Yingnian Wu

Qing Zhou, Committee Chair

University of California, Los Angeles

2024

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1

Introduction

Structure learning is a sub-field of causal inference which focuses on learning the causal structure among a set of variables from observational data. Typically this causal structure can be represented by a structural equation model, an SEM, which expresses each observed random variable as function of its direct causes and a noise term. Ultimately, the goal of structure learning is to fully specify an SEM by learning its parameters. Typically, these SEMs are paired with a directed acylic graph, DAG, that model condition independence implied by the SEM. When all variables are observed we represent an SEM in a compact fashion using a Directed Acyclic Graph (DAG). In this case, typical structure learning will learn the support of the DAG by estimating the conditional independence structure using criteria determined by the observed data ([RS02]). However, in many cases of interest, we may suspect that a latent variable, or unobserved variable, may be part of the true SEM. In this case, conditional independence of the variables cannot always be modeled by a DAG. Our goal is to recover a SEM while accounting for possible effects of latent variables.

To do so, we will work within the frameworks of another type of graphical models. In particular, we will work with acylic directed mixed graphs, ADMGs. ADMGs are graphs that contain directed and bidirected edges, while containing no cycles among the directed edges. In particular, an ADMG will contain a directed edge between two variables i and j if one variable is a cause of another and a bidirected edge if i and j share a latent patent.

Our goal is to perform structure learning for ADMGs to find the ADMG that best explains some observational data. In the remainder of this paper we will propose a hybrid

method that uses a combination of learning conditional independence constraints and score-based optimization to uncover possible Verma Constraints. In particular, we will combine conditional independence constraints obtained by first learning an equivalence class of maximal ancestral graphs, a special class of ADMGs, and then using score-optimization to select an ADMG.

CHAPTER 2

Related Works

There are some recent works on structure learning of ADMGs with linear SEMs with Gaussian noises ([NME17], [BNM21], [FNM19]). These methods are all in the category of score-based learning, using a defined score function to search a space of graphical models. There are also some methods that search for a causal ordering under specific assumptions ([SGK20], [LSP23], [ASP23]). Li et al.'s method relies on normality to infer the causal ordering, while Agrawal et al's method considers the non-linear, pervasive confounding setting. In practice these methods are thereby limited to their areas of analysis where these assumptions almost surely hold. Additionally, score-based learning is a tough optimization problem. The possible space of graphs is large and the nature of the optimization problem is combinatorial. Searching this space can be extremely costly and often impossible. Nowzohour et al.'s method proposes a significant improvement by considering connected components individually.

Besides score-based learning, another popular approach for graph learning is constraint-based learning. There are effective methods for learning DAGS, such as the PC algorithm, which work by learning conditional Independence constraints from data to construct the Markov equivalence class of a DAG [SG91]. Unfortunately, there are no such algorithms for learning the structure of ADMGs. The reason these algorithms are hard to develop is the fact that ADMGs encode not only conditional independence constraints but also encode generalized conditional known as Verma constraints ([RER23], [VP22]). As a generalized conditional independence constraint is a function of multiple conditional distributions, it is unknown how to uncover there constraints from data ([ZCP20]). Seeking a subset of ADMGs

where Verma constraints are not present, maximal ancestral graphs were defined ([RS02], [Zha08b]). As a result of a MAG only capturing the conditional independence constraints among the observed variables there are efficient algorithms to learn them. This allows us to learn the conditional independence constraints in a similar manner. Of particular importance to the rest of this paper is the FCI algorithm. The FCI algorithm learns a partial ancestral graph, PAG, which represents the Markov equivalence class of a MAG from observed data ([SMR99], [CMK12], [CDS23]).

CHAPTER 3

Methodology

3.1 Structural Equation Models and the Need for ADMGs

Consider a general framework such that we have a set of observed random variables $X = \{X_1, \ldots, X_p\}$ which have complex conditional independence relations among them. If we are able to characterize relationships within this set of variables, one would define a structural equation model, SEM. A SEM is a system of equations involving the set of variables X, and error terms $\epsilon = \{\epsilon_1, \ldots, \epsilon_p\}$. For our purposes, we will consider a Gaussian linear SEM, which requires all equations to be linear and ϵ to follow a normal distribution:

$$X = BX + \epsilon, \epsilon \sim N(0, \Omega) \tag{1}$$

where $X \in \mathbb{R}^p$ and $\Omega = diag(\omega_1, \ldots, \omega_p)$. The model defines p linear equations of the form $X_j = f(X \backslash \{X_j\}, \epsilon, B_j)$ where f is linear over B_j and ϵ, where B_j is the jth row of the beta matrix B. A SEM of this form defines each ϵ_i to be independent of all other epsilons implying that there is no confounding and all residual variance is random noise. Of particular importance, this equation defines a probabilistic model, P, over X, such that $X \sim N(0, \Theta^{-1})$ where $\Theta = (I_p - B)(\Omega)^{-1}(I_p - B)^T$ [AZ15].

In a causal setting, a SEM is typically associated with some graph $\mathcal{G}(V) = (V, E)$ where $V = \{1, \ldots, p\}$ represent nodes corresponding to each random variable in X, and $E \subset V \times V \backslash \{(i, i) | i \in V\}$ is the set of edges between these nodes. If all variables in X are observed, the graphical model for V can be represented by a directed acylic graph, DAG. In a DAG, each edge, $i \rightarrow j$ in E denotes that X_i is a cause of X_j. Together the pair $\langle G, P \rangle$,

where P is the probabilistic model defined above, form a Causal Bayesian Network.

The DAG representation of these relationships is particularly useful because its structure encodes the conditional independence relationships among the variables X. The markov properties of DAGs imply that we can rewrite the above equation as

$$X_j = \Sigma_{i \in pa_{\mathcal{G}}(i)} \beta_{ij} X_i + \epsilon_j, \quad j \in \{1, \ldots, p\}, \quad (\epsilon_1, \ldots, \epsilon_p) \sim N(0, \sigma^2) \tag{2}$$

where $pa_{\mathcal{G}}(i) = \{i | i \to j \in E\}$ and denotes the parents of node j.

The goal of structure learning is to learn a model of the form $\langle \mathcal{G}, P \rangle$ from observed conditional independencies ([SGS93]). However, what if we don't observe all variables in X. Suppose we partition X into a set of observed variables, $X_O = \{O_1, \ldots, O_p\}$, and a set of latent variables $X_L = \{U_1, \ldots, U_d\}$. If a latent variable U_i is not a confounder, then the distribution of $X \backslash U_i$, after marginalizing out U_i, can still be modeled by a DAG. Thus we assume all $U_i \in L$ are latent confounders. In this case, a typical SEM cannot be learned. Instead, we consider a new form of Gaussian linear SEM which is now parameterized by both B and Ω:

$$X_O = BX_O + \epsilon, \epsilon \sim N(0, \Omega) \tag{3}$$

such that $B, \Omega \in \mathbb{R}^{p \times p}$. Again this SEM would define a probabilistic model, P, over X_O. We will discuss the meaning of these parameters further in the next chapter. Following the latent projection of DAGs, the marginal model can be modeled instead by an acylic directed mixed graph, ADMG ([TP02]). Thus we can form a model, $\langle G, P \rangle$, where G is an ADMG.

3.2 ADMG Notations

An ADMG over a set of observed variables $X_O = \{O_1, \ldots, O_p\}$ with latent confounders $X_L = \{U_1, \ldots, U_d\}$ can be represent as $\mathcal{G}(O \bigcup L) = (O, E_D, E_B)$ where $O = \{1, \ldots, p\}$ is the set of vertices corresponding to the observed random variables with latent confounders $L = \{p+1, \ldots, p+d\}$. Here E_D, E_B are defined such that $E_D, E_B \subset O \times O \backslash \{(i,i) | i \in O\}$.

Since an ADMG is a mixed graph, it contains two types of edges: directed edges, denoted as E_d, and bidirected edges, denoted as E_b. If $(i, j) \in E_d$, the ADMG contains a directed edge from i to j. If $(i, j) \in E_b$, the ADMG contains a bidirected edge from i to j. In this paper, we will consider only bow-free ADMGs, which are ADMGs such that satisfying the condition that if a variable X_i occurs in the structural equation for X_j, then the errors for X_i and X_j are uncorrelated. For this subset of ADMGs, parameter identifiability under linear SEMs has been established ([BP02]).

To help work towards a working understanding of an ADMG, we will define some characteristics. We define parents of a node i as $pa_{\mathcal{G}}(i) = \{j|(j, i) \in E_d\}$ and the siblings as $sb_{\mathcal{G}}(i) = \{j|(j, i) \in E_b\}$. Each definition can easily be extended to a group of nodes, $H \subset O$, as $pa_{\mathcal{G}}(H) = \{j|(j, i) \in E_d\, i \in H\}$ and $sb_{\mathcal{G}}(H) = \{j|(j, i) \in E_b\, i \in H\}$. Additionally, a path of length l, $\pi = (\pi_1, \ldots, \pi_l)$, is a sequence of distinct nodes such that $\pi_i \in O$ and each pair (π_i, π_{i+1}) has some edge between the nodes. We call a path a directed path, or bidirected path if all edges along this path are of the specified type. This notion of directed paths is used to define the ancestors of a node i as $an_{\mathcal{G}}(i) = \{j|i = j$ or there exists a directed path $j \to \cdots \to i\}$ and descendants of a node i as $de_{\mathcal{G}}(i) = \{j|i = j$ or there exists a directed path $i \to \cdots \to j\}$. Additionally, if there is a directed path such that $pi_1 = pi_1$, we call this path a directed cycle. There is a similar definition for bidirected cycles. We can now provide a formal definition of the set of ADMGs we wish to consider:

Definition 3.2.1 *An bow-free ADMG is a directed mixed graph satisfying the following criterion*

1. *Acylic : There are no directed cycles in the mixed graph. In other words for any $i \in O$, $i \notin an_{\mathcal{G}}(i)$.*

2. *Bow-Free : If a variable O_i occurs in the structural equation for O_j, then the errors for O_i and O_j are uncorrelated*

We also will need to define some properties about nodes themselves. We denote a vertex i as a collider if along a path there is structure $k \circ\!\!\rightarrow i \leftarrow\!\!\circ j$ or in other words $(k, i) \in E_b \bigcup E_d$ or $(j, i) \in E_b \bigcup E_d$. If such a structure exists and $(k, j) \notin E_b \bigcup E_d$ this structure is denoted as a v-structure.

We also define a notion of conditionally independence in ADMGs, m-separation. Consider a path between two verticies i and j and a set of conditioning variables, H, such that $i, j \notin H$. A path between i and j is called m-connecting if every non-collider on the path is not in H, and every collider on the path is either in H or an ancestor of H. Then, if there is no such m-connecting path between i and j, we call them m-separated. This specifies that $O_i \perp\!\!\!\perp O_j | S$ by the Global Markov Property ([Zha08a]) where S corresponds to variables associated with the verticies in H. As mentioned above, due to ADMG's mixed nature, they not only encode typical conditional independence constraints but also generalized conditional independence constraints, known as Verma constraints. Verma constraints are equality constraints on functions of multiple conditional distributions.

We will also need to formally define a MAG and a PAG for the remainder of the paper. To start we need to formally define a few more characteristics of an ADMG quickly:

Definition 3.2.2 *1. Inducing Path : A path is inducing if every node, except for the endpoints, on the path is a collider and every collider is an ancestor of a path endpoint*

2. *Ancestral ADMG : An ADMG is ancestral if for all $(i, j) \in E_b$, $j \notin an_{\mathcal{G}}(i)$ (no almost directed cycles) and there are no directed cycles.*

3. *Maximal ADMG : An ancestral ADMG is maximal if there is no inducing path between any two non-adjacent nodes in $\mathcal{G}$*

Definition 3.2.3 *A MAG is an ADMG that is both ancestral and maximal*

A MAG encodes the CI constraints, in accordance with the definition of m-seperation, of the observed variables, X_O, in a DAG with latent variables, X_L. To create a MAG from an

existing ADMG, there are two key steps. First, for any bidirected edges $i \leftrightarrow j$, we change it to a directed edge $i \rightarrow j$ if $i \in an_{\mathcal{G}}(j)$ to remove almost directed cycles. Next, we need to add an edge between all nonadjacent nodes i, j in $\mathcal{G}$ if there is an inducing path between them. If there is an inducing path, if i is an ancestor of j, we orient the edge as a directed edge $i \rightarrow j$ or $j \rightarrow i$ if j is an ancestor of i. Otherwise, we orient it as a bidirected edge. This maintains ancestral relations. We formalize this in algorithm 1.

Algorithm 1 ADMGtoMAG

Define empty graph $M = (V, E_d, E_b)$
Add all directed edges from ADMG $\mathcal{G}$ to M.
for all $i \leftrightarrow j$ in $\mathcal{G}$ **do**
    if $i \in an_{\mathcal{G}}(j)$ **then** Update edge set as $E_d = (i, j) \bigcup E_d$
    else if $j \in an_{\mathcal{G}}(i)$ **then** Update edge set as $E_d = (j, i) \bigcup E_d$
    else Update edge set $E_b = (i, j) \bigcup E_b$
    end if
end for
for all non-adjacent (i, j) **do**
    if there is an inducing path **then** Perform steps 4-7
    end if
end for

We note that multiple MAGs may encode the same set of m-seperations. Naturally, this defines an equivalence class, $[\mathcal{M}]$, for a MAG $\mathcal{M}$. This equivalence class can be represented by a partial ancestral graph, PAG. A PAG uses the notation of marks to define the equivalence class. A mark is the figure at the end of each edge, of which there are three options: arrowhead, arrowtail, and $\circ$. For example, a directed edge $\rightarrow$ contains an arrowtail on the left and an arrowhead on the right. Meanwhile, $\leftrightarrow$ has an arrowhead on each end. A $\circ$ mark is a bit different, as it is represent a variant mark, one that varies across the equivalence class. We can now formally define a PAG:

Definition 3.2.4 *For an equivalence class* $[\mathcal{M}]$*, the corresponding PAG,* $\mathcal{P}$

1. *has the same skeleton as* $\mathcal{M}$

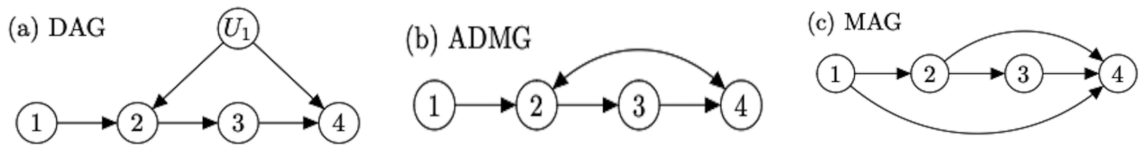


Figure 1: (A) DAG with a latent confounder U_1 (B) ADMG Representation (C) MAG Representation

2. *a mark of an arrowhead or arrowtail iff it is shared by all MAGs in* $[\mathcal{M}]$

3. *a mark* $\circ$ *if marks not shared by all MAGs in* $[\mathcal{M}]$ *are denoted with* $\circ$

([Zha08a]).

For an example of the graphs we have introduced, please see Figure 1.

3.3 ADMG Model Estimation

Previously, we expressed our Gaussian Linear SEM as $X_O = BX_O + \epsilon, \epsilon \sim N(0, \Omega)$. Representing the marginal model of X_O as an ADMG, $\mathcal{G}(O) = (O, E_d, E_B)$, and applying the properties discussed above and an ADMG's Markov Conditions we can rewrite the above equation as:

$$O_j = \Sigma_{i \in pa_{\mathcal{G}}(j)} \beta_{ij} O_i + \epsilon_j, j \in \{1, \ldots, p\},$$
$$(\epsilon_1, \ldots, \epsilon_p) \sim N(0, \Omega)$$

You will notice that our definition here is similar to equation (2) except there is the possibility that the background variables, ϵ_j, have non-zero covariance. This mechanism allows us to model variance exhibited by latent confounders. Here $B = (\beta_{ij})_{p \times p}$ defines the strength of causal relationships among X_O while $\Omega = (w_{ij})_{p \times p}$ models the covariance between the background variables, ϵ_i. Naturally, this expression of the SEM places natural

constraints on these parameters:

$$\begin{aligned} B \in \mathbb{B}(E_d) &= \{(\beta_{ij})_{p\times p} : \beta_{ij} = 0 \text{ if } (i,j) \notin E_d\} \\ \Omega \in \mathbb{P}(E_b) &= \{(\omega_{ij})_{p\times p} : \omega_{ij} = 0 \text{ if } (i,j) \notin E_b\} \end{aligned} \tag{4}$$

Thus, non-zero components of B correspond to directed edges in $\mathcal{G}$ and non-zero components of Ω correspond to bidirected edges in $\mathcal{G}$.

Learning the structure of $\mathcal{G}(O)$ is equivalent to recovering the supports of B and Ω. In doing so, we fully specify the probabilistic model, P, as a multivariate Gaussian distributions such that $X_O \sim N_p(0, \Sigma)$ where $\Sigma = \Sigma_{\mathcal{G}}(B, \Omega) = (I - B)^{-T}\Omega(I - B)^{-1}$. Therefore, by learning the structure of $\mathcal{G}(O)$ we can specify a a model $\langle \hat{G}, \hat{P} \rangle$ ([AZ15]).

Due to structural constraints on B and Ω it is not straightforward to maximize the likelihood over the parameter space. However, it has been show that these estimates can be obtained iteratively. Residual iteration conditional fitting, RICF, is a method that uses an iterative approach that provides accurate approximations of the maximum likelihood estimates for Gaussian linear SEMs of bow-free ADMGs ([Drt08]). We made two small changes to the source code of this algorithm. First, we add a maximum number of iterations to prevent infinite computations. Second, we add hot start capability to limit the number of iterations required by convergence when making small changes to the adjacency matrix. This algorithm allows us to fit proposal ADMGs within our algorithm.

3.4 Constraint-Score Hybrid Structure Learning

Suppose we are a given a dataset $X = \{X_{ij} : i \in \{1, \ldots, n\}, j \in O\}$ where n is the number of samples and $O = \{1, \ldots, p\}$ is the set of observed variables. Our goal is to obtain the most likely ADMG, $\hat{\mathcal{G}}$. To do so, we propose a hybrid method that forms a equivalence class using constraint based methods, followed by a score based optimization algorithm.

The basis of the constraint-score hybrid stucture learning algorithm is to first learn a PAG, $\hat{P}$, from our data. Given the estimated PAG, we can form a candidate set of ADMGs,

$C(\hat{P})$. This is the set of ADMGs such that their corresponding MAG is within the MAG equivalence class represented by the estimated PAG, $\hat{P}$:

$$C(\hat{P}) = \{ADMGtoMAG(\mathcal{G}) \in \hat{P}\}$$

Once we have obtained the equivalence class, we use it to restrict the search-space of our score-based search.

3.4.1 Constraint-Based MAG Learning

We will now describe the details of the constraint-based mag learning in greater details. Constraint-based algorithms typically learn conditional independence constraints from the data. Recall that ADMGs not only encode conditional independence constraints among X_O but also encode Verma constraints ([RER23], [VP22]). Currently, it is unknown how to find such constraints from data or how to use them to perform constraint-based learning ([ZCP20]). However, there are efficient algorithms to uncover MAGs from observational data, since MAGs only encode conditional independence constraints among observed variables ([RS02], [Zha08b]).

Within our algorithm, we will employ the fast causal inference algorithm, FCI, to fit an estimated PAG, $\hat{P}$, from observational data. The algorithm consists of a skeleton learning via CI tests followed by a set of edge orientation rules ([Zha08b], [SMR99], [CMK12]). For our Gaussian linear setting, we use a typical Gaussian conditional independence test. In other words, using our observed data, we test the hypothesis that $X \perp\!\!\!\perp Y|S$ for each $(X, Y) \in X_O \times X_O$ and consider all subsets $S \subset X_O$ as conditioning sets. There are two hyperparameters of note that our algorithm takes as a result of the FCI algorithm. The first is the significance level, α, of the hypothesis tests and the second is the maximum size of a conditioning set S we wish to consider. We recommend a maximum size of conditioning set S between 4 and 7, and an α between 0.01 and 0.1.

After obtaining the PAG, we wish to establish the equivalence class of ADMGs implied

by $\hat{P}$, $C(\hat{P})$. We know that each M represented by $\hat{P}$ is a member of the equivalence class, $M \in C(\hat{P})$. This gives us a natural subset of $C(\hat{P})$ to use as a starting point for score-based learning. We call this subset $C_M(\hat{P})$ where $C_M(\hat{P}) = \{M : M \in \hat{P}\}$ and $C_M(\hat{P}) \subseteq C(\hat{P})$. We then populate $C_M(\hat{P})$ following the Arrowhead Augmentation rules set out in [Zha06]. Starting with $\hat{P}$, we orient any edge $\circ\!\!\rightarrow$ into $\rightarrow$. Following this step, we choose a fixed node, i, and orient each chordal component of the graph into a valid DAG without orienting any additional edges into i. To build $C_M(\hat{P})$, we perform this algorithm for each $i \in \{1, \ldots, p\}$. In the best case scenario we would obtain one MAG per node; however, usually many of the resulting MAGs are identical.

For a summary of constraint-based MAG Learning see algorithm 2.

Algorithm 2 Constraint-Based MAG Learning

Input: $\{X_{ij}\}_{n\times p}$, α_f, $m.max$

Output: $C_M(\hat{P})$

1: Perform FCI using gaussain CI tests $\hat{P} = FCI(data = X,$ significance level $= \alpha_f,$ max conditioning set size $= m.max)$
2: **for** $i \in \{1, \ldots, p\}$ **do**
3: $M_i = \hat{P}$
4: Perform Arrowhead Augmentation from [Zha06]
5: Orient all $\circ\!\!\rightarrow$ in M_i to $\rightarrow$
6: Orient each chordal component in M_i of our graph into a valid DAG without orienting any additional edges into i
7: Store M_i
8: **end for**

3.4.2 Score-based ADMG Learning

To perform score-based learning, at a conceptual level, we define a score $S(\mathcal{G}|X)$ for any candidate ADMG $\mathcal{G}$ and observed data X. The goal of the search algorithm is to find $\hat{\mathcal{G}} \in C(\hat{P})$ to minimize the score $S(\hat{\mathcal{G}}|X)$. This process will uncover generalized CI constaints which cannot be detected by ordinary MAG learning. In other words, we start with a MAG and perform a set of steps to uncover generalized CI constraints that may decrease our score.

For each $M_i \in C_M(\hat{P}); i \in \{1, \ldots, |C_M(\hat{P})|\}$ we repeatedly perform four steps to find the

optimal ADMG. These four steps are preceded and followed by an additional greedy step. Due to the parameterization of the gaussian linear SEM, we choose BIC as our score, which we express as:

$$\begin{aligned} S(\mathcal{G}|X) = BIC(\mathcal{G}) &= -2\ell(\Sigma = (I-B)^{-T}\Omega(I-B)^{-1}) + log(n) * n_p \\ &= -2(\frac{-n}{2}log(|\Sigma|) + tr(\Sigma^{-1}S)) - \text{ constant } + log(n) * n_p \end{aligned}$$

where S is the sample covariance matrix and n_p is the number of parameters implied by the current model

Performing these six steps, we hope to uncover the optimal ADMG from the starting MAG, M_i. We call the result of this process A_i. After we have performed this process for all M_i, we select the optimal ADMG, A^*, using a simple argmin operation: $A^* = argmin_{A' \in \{A_1,\ldots,A_{|C_M(\hat{P})|}\}} S(A'|X)$

The six steps fit into two categories, greedy steps, more specifically a greedy add and greedy remove step, and ADMG search steps. First, we perform a greedy add step to increase the density of the starting graph. Then, we perform the ADMG search steps. These steps are informed by the steps taken to convert an ADMG to a MAG. Finally, we perform a greedy remove step to remove unnecessary edges. In the rest of this chapter, we will discuss the details of each step.

This process is formalized in algorithm 3.

Greedy Steps Given an initial MAG M_i, we perform a greedy add step to obtain, G_0, the initial graph for the ADMG search steps. The main goal of this step is to combat potential sparsity in the PAG $\hat{P}$.

The greedy add step considers all possible additions of bidirected and directed edges to $G_{add}(O) = (O, E_b, E_d)$. Following the definition of a bow-free ADMG, we can only add edges

Algorithm 3 Score-Based ADMG Learning

Input: $C_M(\hat{P})$, $max.iter$

Output: A^*

for $M_i \in C_M(\hat{P})$ **do**
  $G_0 = greedyBICAdd(M_i)$
  Find $pid(G_0)$
  Find $pbi(G_0)$
  while $i < max.iter$ **do**
    $G_{t+1} = argmin_{G \in \{G' = DirectedToBidirected(G_t, G_t\}} BIC(G)$
    $G_{t+1} = argmin_{G \in \{G' = RemoveInducingEdge(G_{t+1}, G_{t+1}\}} BIC(G)$
    $G_{t+1} = argmin_{G \in \{G' = ReverseStep3(G_{t+1}, G_{t+1}\}} BIC(G)$
    $G_t = argmin_{G \in \{G' = ReverseStep4(G_{t+1}, G_{t+1}\}} BIC(G)$
    Break if no changes were made during this step
  end while
  $A_i = greedyBICRemove(G_{res})$
  Store A_i
end for
$A^* = argmin_{A' \in \{A_1, \ldots, A_{|C_M(\hat{P})|}\}} BIC(A')$

that do not create a bow nor create a cycle. Suppose we have the following set of edges

$$R = \{(i,j) | (i,j) \in O \times O \text{ and } (i,j) \notin E_d \bigcup E_b \\ \text{and Adding (i,j) does not make a cycle or bow in } G_{add}\}$$

At each step we attempt we form a set of proposal graphs, $G(R) = \{G_{E_1}, \ldots, G_{E_{|R|}}\}$ where each graph has one more edge than G_{add}, $E_i \in R$. We score each of the graphs in $G(R)$ using our score function $S(G|X)$. Then, we select the graph with the minimal score, G' from amongst $G(R)$. If $S(G'|X) < S(G_{add})$, we accept G', i.e. $G_{add} = G'$ and this process continues with $R = R \backslash E$. We do this until there are no more edges that decrease the score or we have hit a max number of edge additions.

There may be concerns that such an approach would lead to over-parametrization, however, step two of the ADMG search steps in combination with the greedy remove step address this well. Greedy remove considers removing all edges in the final graph $G_{res}(O) = (O, E_d, E_b)$ for removal. Again consider the edge set $R = E_b \bigcup E_d$. For each

$E_i \in R$, we will form a proposal graph, G_{E_i} with one less edge than G_{res}. Again we call this set of proposal graphs $G(R) = \{G_{E_1}, \ldots, G_{E_{|R|}}\}$. From amongst $G(R)$, we select the graph with the minimal score, G', and its associated edge removal, $E \in R$. If $S(G'|X) < S(G_{res}|X)$, we remove this edge from G_{res}, i.e. $G_{res} = G'$, and continue the process with the edge set $R = R \backslash E$. We continue this process until there are no more edges that decrease the score.

ADMG search steps Now we will discuss the four ADMG search steps. Each step is characterized by the same process. Given the current graph G_t, we make a local change to propose a new G' and then follow an acceptance-rejection rule to obtain G_{t+1}.

To propose a local change we define two subsets of edges from the initial graph, G_0. G_0 is the initial MAG, M_i, after the greedy add step. These two subsets are related to the process of converting an ADMG to a MAG. Lines 3-8 in algorithm 1 imply that a directed edge in our MAG G_0 could be a bidirected edge in an ADMG if there is an ancestral relation between the two nodes. Accordingly we define the set of possible bidirected edges, $pbi(G_0)$:

$$pbi(G_0) = \{(i,j)|(i,j) \in E_d \text{ and } i \in an_{G_{rem}}(j)\}$$

where G_{rem} is the graph G_0 where the edge (i,j) has been removed

Similarly, due to the operations performed in lines 9-12 in the same algorithm, any edge in G_0 may have been added to an ADMG due to there being an inducing path between the nodes. We define the set of possible inducing edges as $pid(G_0)$ or more formally:

$$pid(G_0) = \{(i,j)|(i,j) \in E_d \bigcup E_b \text{ and there is an inducing path between i and j}\}$$

Each admg search step iterates over some subset of the edges of $G_t = (O, E_d, E_b)$, $R \subset E_d \bigcup E_b$. A proposal graph, G_E, is created and scored for each edge $E \in R$. We call the set of these proposal graphs $G(R) = \{G_{E_1}, \ldots, G_{E_{|D|}}\}$. We consider this set of graphs and may

accept a change based off one of two acceptance rules. The two currently implemented rules are a greedy rule and a simulated annealing rule. The greedy rule implements an exhaustive search algorithm in which each graph $G_E \in G(R)$ is considered the graph with the minimal score $S(G|X)$ is denoted as G'. We then accept the change if $S(G'|X) < S(G_t|X)$. On the other hand, when using the simulated annealing rule, while iterating through the graphs in $G(R)$ we may accept a graph that increases our score based on a bernoulli draw. This rule requires two parameters, a temperature, T, that denotes the probability of accepting a change regardless of its effect on the score and a cooldown degree, C, that is the shrinkage factor of T after each iteration. Whenever a proposal graph $G_E \in G(R)$ is scored, we perform one bernoulli draw, $B \sim Bern(T)$, to determine whether or not we should accept the edge change. If we accept an edge in this manner it is added directly to to the current graph G_t and we skip all other edges in R. Otherwise, if an edge change is not accepted by this draw, it will be accepted or rejected based on the greedy rule.

Following this framework, we define the four steps in terms of the edges we iterate over. Each of these define a different edge set R in which we perform the above process. In the first step, we propose to convert a directed edge in $(i, j) \in E_d \bigcap pbi(G_0)$ to a bidirected edge. This is the set of all edges in $pbi(G_0)$ that are still directed edges in G_t. In the second step, we consider all edges in $(i, j) \in (E_b \bigcup E_d) \bigcap pid(G_0)$ for removal. These are all the edges in $pid(G_0)$ that are still in G_t. To ensure reversibility, we also consider the reverse of these steps. We consider converting an edge $(i, j) \in E_b \bigcap pbi(G_0)$ back to a directed edge and add an edge $(i, j) \in pid(G_0) \backslash E$ to G_t. These reverse steps consider the sets of edges that we have changed in a previous step of our algorithm.

3.5 Implementation

To clarify the implementation of *pbi* and *pdi*, we will dicuss the process of building both here

3.5.1 Building pbi

Consider an ADMG $G(O) = (O, E_d, E_b)$. We define edges that are possible bidirected edges as directed edges in the current ADMG that may have been converted from directed edges in the process of forming a MAG. To maintain ancestral properties this set is defined as $pbi(\mathcal{G}) = \{(i, j) : i- > j \ \& \ i- > ...- > j\}$.

To build pbi, we need to identify pairs of verticies, (i, j) such that $(i, j) \in E_d$ and there is a directed path between i and j of length 2 or longer. We first form a new graph $G_D = (O, E_d)$, where E_d is equivalent to the set of directed edges in G. To identify paths of length l, we consider A^l where A is the adjacency matrix of G_D. In a directed graph, if $A^l_{ij} = 1$ there is a path of length l ($l \geq 2$) from i to j. Therefore an edge is added to pbi, if for any $l \in \{1, \ldots, p\}$, $A^l_{ij} = 1$ and $(i, j) \in E_d$. We do this via an iterative approach in which we iterate through $l \in \{2, \ldots, p\}$ adding pairs of vertices (i, j) such that $A^l_{ij} = 1$ and $(i, j) \in E_d$.

3.5.2 Building pid

Consider an ADMG $G(O) = (O, E_d, E_b)$. We define edges as possible inducing edges, $pid(\mathcal{G})$, if there is an inducing path (of length ≥ 2) between i and j. To decrease the complexity of this problem, we form a superset of $pid(G)$, which we denote as $pid_S(G)$. $pid_S(G)$ drops the assumption that each intermediate node must be an ancestor of a path endpoint. In this form, we only need to find pairs of nodes $(z, w) \in E_b \bigcup E_d$ such that there is a path of the form $w \circ\!\!\rightarrow i \leftrightarrow \ldots \leftrightarrow j \leftarrow\!\!\circ z$ where i and j are any two nodes $i, j \in O$. Thus we break this problem down into two parts: find the endpoints of bidirected paths of a length $l (l \geq 2)$ and finding nodes with an edge into these endpoints.

We consider this problem in the same way we considered building pbi. First we form an undirected graph, $G_U(0) = (O, E)$, where each bidirected edge $(i, j) \in E_b$ is added as an undirected edge to E. We will use the adjacency matrix of G_U, A, to find paths of length l in G_U and in turn find bidirected paths of length l in G. To find paths of length l, we consider

A^l where A is the adjacency matrix of G. As we do this, we ignore the diagonal of the power matrix as we don't care about paths back to a start node on the undirected graph. Thus, if $A^l_{ij} \geq 1$ where $i \neq j$ there is a bidrected path of length l between i and j. However, we actually care about all the nodes that are adjacent to i and j rather than the endpoints of this path. Thus for any pair of nodes $(z, w) \in E_b \bigcup E_d$ such that $(z, x), (w, y) \in E_b \bigcup E_d$ and $A^l_{xy} \geq 1; x \neq y$ for any length $l \in \{2, \ldots, p\}$, we add (z, w) to pbi_S. Again, we can employ an iterative approach in which we iterate through $l \in \{2, \ldots, p\}$ adding pairs of vertices (i, j) to pbi_S that satisfy these conditions.

CHAPTER 4

Results

In this chapter, we summarise empirical results showing our methods performance across various dimensional and sample size settings. For each setting, we run $n_s = 50$ simulations. For comparison purposes, we compare our method to the greedySearch method proposed by Nowzohour et al. and LRpS (Low-rank plus Sparse) estimator proposed by Frot et al([NME17], [FNM19]). The greedySearch method follows a score-based approach in which they score connected components individually to get around computation limitations. The result of greedySearch is the optimal ADMG based on their connected component criterion. The LRpS + GES algorithm considers a convex optimization problem that accounts for latent variables. The results of LRpS + GES is a CPDAG and is used as a non-ADMG proxy for performance on data with latent variables.

4.1 Generated Data Simulations

This set of simulations heavily derives from the simulation framework formulated by Nowzohour et al., who borrow from Kuipers et al. ([NME17], [KM13]). Here we randomly generated $N = 50$ ADMGs, sample a specified number of data points from the implied distribution, and compare the estimators. We will begin by discussing the simulation framework and then proceed to discuss the results.

4.1.1 ADMG Generation Framework

We wish to sample bow-free ADMGs uniformly at random from the space the possible ADMGs. However, this is not a straightforward problem and simple generation procedures exhibit highly bias results. Following Kuipers et al., and Nowzohour et al. we can use a random process with graphs as states and a uniform limiting distribution. To do so Nowzohour et al. developed the following MCMC algorithm for ADMGs and proved that the following algorithm results in a transition matrix that is irreducible and symmetric and therefore exhibits a unique uniform stationary distribution.

Let $G_k(O) = (O, E_d, E_B)$ be the ADMG at the current step of the MCMC iteration. At each step of the iteration, we sample a position $(i, j) \in V \times V \backslash \{(i, j) | i \in V\}$ uniformly at random and perform a single bernoulli draw $\sigma \sim Bernoulli(0.5)$. Using these two sampled values, we will formulate proposal edge sets E'_D and E'_B. If there is an edge present between i and j and $\sigma = 0$, we remove the edge $(E'_D = E_D \backslash \{(i, j), (j, i))\}, E'_B = E_B \backslash \{(i, j), (j, i))\})$. Otherwise if $\sigma = 1$, we do nothing. On the other hand, if there is not an edge present between i and j and $\sigma = 0$ we add a directed edge $i \to j (E'_D = E_D \bigcup \{(i, j)\}, E'_B = E_B)$. If $\sigma = 1$, we add a bidirected edge $i \leftrightarrow j (E'_D = E_D, E'_B = E_B \bigcup \{(i, j), (j, i)\})$. Finally, we update the current graph $G_{k+1}(O) = (O, E'_D, E'_B)$ ([NME17], [KM13]).

Due to its properties, after an initial burn-in period, the distribution of across all possible ADMGs has a unique uniform stationary distribution. Starting with an empty graph, Nowzohour et al. suggests sampling ADMGs after a burn in period of $O(d^4)$.

In the following simulations, we follow this process and sample 50 ADMGs, $n_s = 50$, for each setting. Additionally, following Nowzohour et al. we only take ADMGs such that the maximum in degree of a node is $0.3 * p$. However, varying the in degree does not effect the performance of any algorithm. After sampling an ADMG, we generate the parameters, B and Ω, following the same process as Nowzohour et al.. For all $(i, j) \in E_d$, we generated $B_{ij} \sim N(0, 1)$. Ω is a bit more tricky due to the constraint that it must be positive definite.

Nowzohour et al. follows Gershgorin's circle theorem, to satisfy this constraint. For all $(i,j) \in E_b$ we generate $\Omega_{ij} \sim N(0,1)$ and set the diagonal as the row sums of the absolute values plus an independently sampled $\chi^2(1)$. This provides us with a positive definite Ω [NME17].

Now that we have the parameters, we simply sample N data points from the implied normal distribution. To be clear for each setting, we take $X = \{x_1, \ldots, x_N\} \overset{\text{iid}}{\sim} N(0, (I - B)^{-T}\Omega(I - B)^{-1})$ where $x_i \in \mathbb{R}^p$.

To compare the three methods discussed above, we will use the following metrics:

1. Expected Bayesian Information Criterion(BIC) Difference : We define the Expected BIC Difference as the mean difference between the BIC of an estimated ADMG and the true ADMG. This can be estimated as:

   $$E\hat{B}IC((G_{True_i})_{i\in\{1,\ldots,n_s\}}, (G_{method_i})_{i\in\{1,\ldots,n_s\}}) = \frac{1}{n_s}\Sigma_{i=1}^{n_s} BIC(G_{method_i}) - BIC(G_{True_i})$$

   We interpret a value of 0 as, on average, the method perfectly predicting the true ADMG.

2. Test Likelihood : We sample an additional 10,000 datapoints $X_{test} = \{x_1, \ldots, x_{10,000}\} \overset{\text{iid}}{\sim} N(0, (I - B)^{-T}\Omega(I - B)^{-1})$ and use this data to calculate the test likelihood of each method. This characterizes how well the models do on new data.

3. F1 Score: Following Nowzohour et al., we estimate the equivalence classes of the True ADMG G and each estimated bow-free ADMG. The equivalence class $EC(G)$ is defined as all bow-free ADMGs G' such that their score is within ϵ, a small number, of the score of G. Nowzohour et al. proposes an algorithm to emprically estimate an equivalence class, $\hat{EC}(G)$, that initially populates $\hat{EC}(G)$ with bow-free ADMGs that have the same skeleton and collider triples as G. This first step follows Nowzohour et al.'s derived sufficient conditions for equivalence of two ADMGs. Then starting from each of these ADMGs, a recursive search is performed by checking all possible edge-changes.

Each ADMG found during this search that is within ϵ of the score of G is added to $\hat{EC}(G)$ [NME17]. For each iteration in our simulation, we generate $\hat{EC}(G)$ for the true ADMG and $\hat{EC}(\hat{G})$ with an $\epsilon = 1 \times 10^{-9}$.

Edge	$E_{possible}(\hat{EC}(G), i, j)$	$E_{possible}(\hat{EC}(\hat{G}), i, j)$
(1,3)	$\leftrightarrow$	$\leftrightarrow$
(1,4)	$\{\leftrightarrow, \leftarrow\}$	$\{\leftrightarrow, \leftarrow\}$
(1,4)	$\{\leftrightarrow, \leftarrow\}$	$\{\leftrightarrow, \leftarrow\}$
(2,3)	$\leftarrow$	$\leftarrow$
(3,5)	$\leftrightarrow$	$\leftrightarrow$
(4,5)	$\{\leftrightarrow, \rightarrow\}$	$\{\leftrightarrow, \rightarrow\}$

Table 4.1: Edge Sets resulting from equivalence class comparison of a toy $p = 5$ example

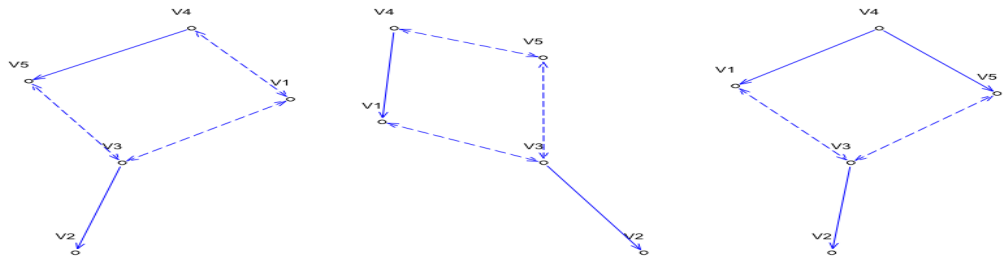
Suppose we have the true ADMG $G(O) = (O, E_d, E_b)$ and its corresponding equivalence class $\hat{EC}(G)$ along with the estimated ADMG $\hat{G}(O) = (O, E_d, E_b)$ and its corresponding equivalence class $\hat{EC}(\hat{G})$. For all $(i, j) \in O \times O$ where $i \neq j$, we will consider the set of edges that are between these nodes throughout the equivalence class. For clarity, we denote this set of edges $E_{possible}(\hat{EC}(G), i, j) \subseteq \{\rightarrow, \leftarrow, \leftrightarrow\}$. There are three types of possible errors: incorrectly oriented edges(R), false positive(FP) and false negatives (FN). In our case these notions need to be extended across $E_{possible}(\hat{EC}(G), i, j)$:

$$TP : E_{possible}(\hat{EC}(G), i, j) = E_{possible}(\hat{EC}(\hat{G}), i, j)$$

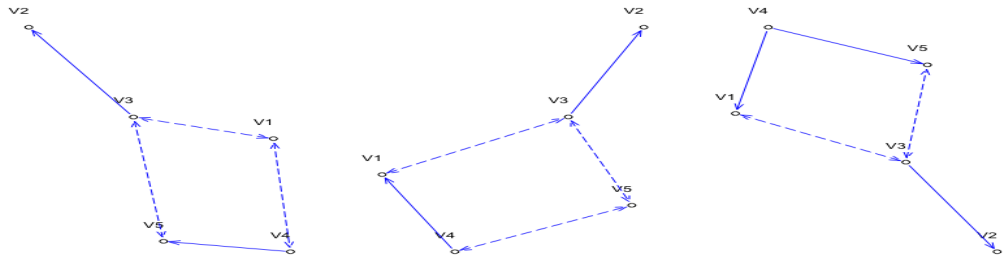
$$FP : E_{possible}(\hat{EC}(\hat{G}), i, j) \neq \emptyset \text{ but } E_{possible}(\hat{EC}(G), i, j) = \emptyset$$

$$FN : E_{possible}(\hat{EC}(\hat{G}), i, j) = \emptyset \text{ but } E_{possible}(\hat{EC}(G), i, j) \neq \emptyset$$

$$R : E_{possible}(\hat{EC}(\hat{G}), i, j) \neq E_{possible}(\hat{EC}(G), i, j) \text{ and}$$
$$E_{possible}(\hat{EC}(G), i, j) \neq \emptyset, E_{possible}(\hat{EC}(\hat{G}), i, j) \neq \emptyset$$



(a)



(b)

Figure 1: (a) : Equivalence Class for true ADMG (b) : Equivalence class for estimated ADMG

To define precision and recall we know that the number of true edges can be expressed as $|E| = TP + FN + R$ and the number of predicted edges $|\hat{E}| = TP + FP + R$. Therefore we get the following definitions:

$$Precision = \frac{TP}{TP + FP + R}$$
$$Recall = \frac{TP}{TP + FN + R}$$
$$F1 = \frac{2 * TP}{2TP + FP + FN + 2R}$$

We will use the F1 score to evaluate each estimation method in this fashion.

As an example, we will display one simulation with $p = 5$ in Figure 1. The resulting edge sets are summarized in table 4.1. This would result in an F1 Score of 1, as for each edge the possible edges are equivalent.

4.1.2 Simulation over Data Dimension

Following the above framework, we perform this simulation for $p \in \{5, 10, 15, 20, 30\}$. The result of these simulations are summarised in Figure 2. We fix the training sample size as $N = 10,000$ to make sample size effects negligible. We see that our model produces, on average, an ADMG with a higher F1 score and an expected BIC difference closer to 0 while producing comparable test log-likleihood and times.

4.1.3 Simulation over Training Sample Size

Following the above framework, we perform this simulation for $N \in \{50, 100, 1000, 5000, 10000\}$. Here we fix $p = 10$, and vary the value of N. The results of these simulations are summarized in Figure 3. As N increases, we see that Hybrid Learning provides an estimated ADMG with a higher F1 Score and an expected BIC difference closer to 0 when compared to the

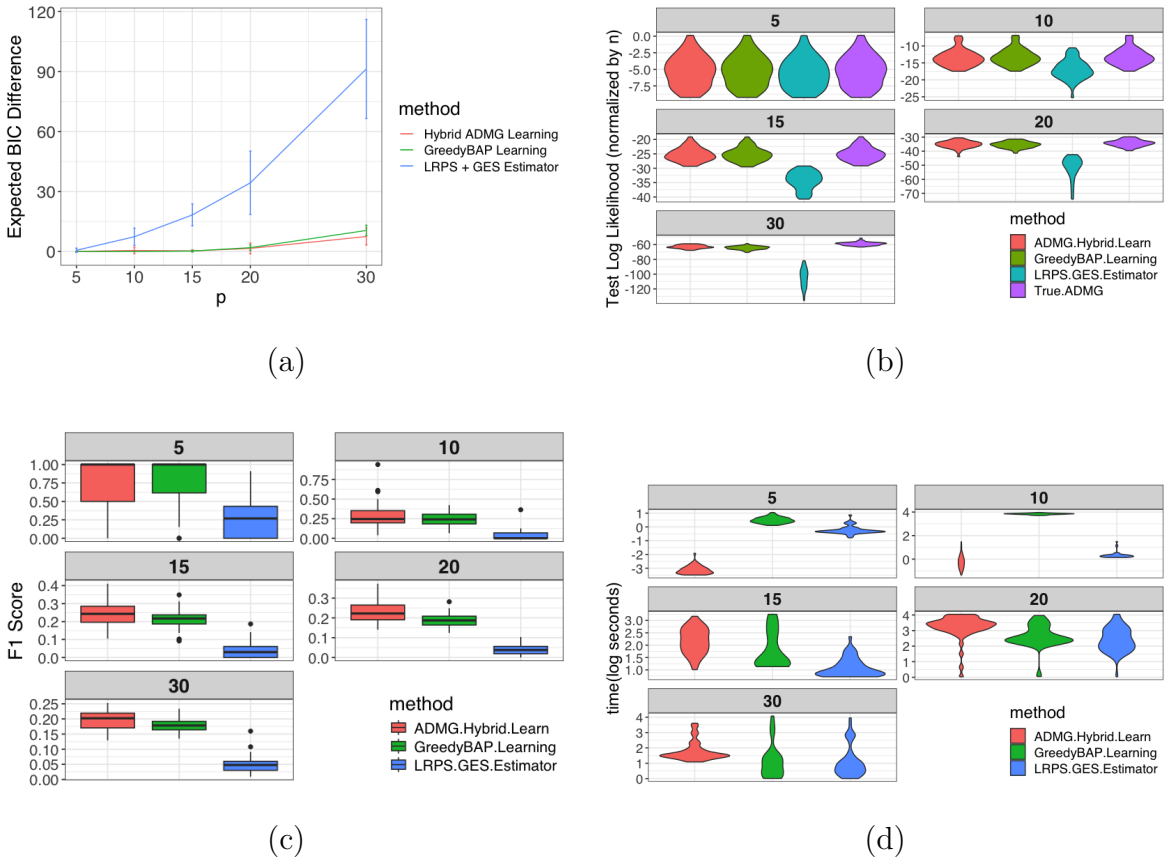


Figure 2: (a) : Estimated Expected BIC Difference across p (b) : Test Log Likelihood for $10,000$ iid data points across p (c): F1 Score for each method's estimated ADMG across p(d) : Time (in log seconds) required to estimate an ADMG across p

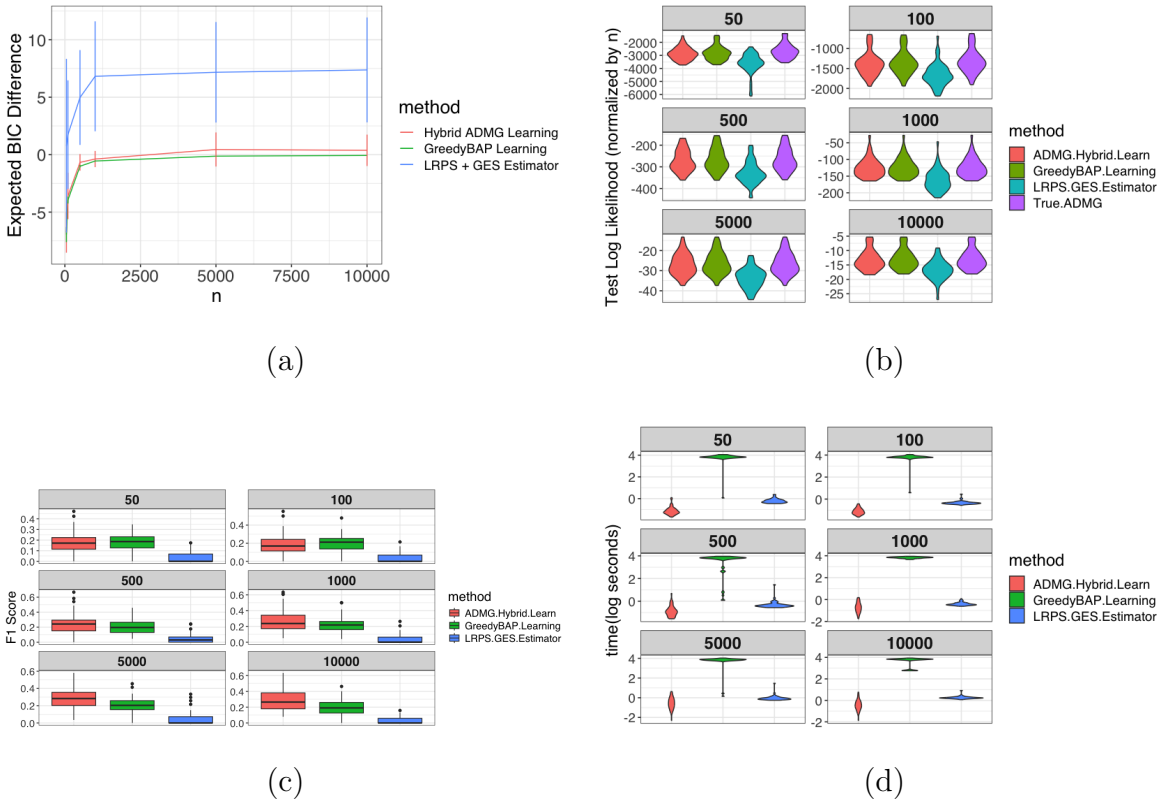


Figure 3: (a) : Estimated Expected BIC Difference across log(n) (b) : Test Log Likelihood for 10,000 iid data points (c): F1 Score for each method's estimated ADMG across n (d) : Time (in log seconds) required to estimate an ADMG

other two methods. Additionally, time is not effected by the number of sample in any of the three algorithms.

CHAPTER 5

Conclusions

Here we have developed and presented structure learning of bow-free Gaussian Linear ADMGs. Our method utilizes constraint based learning in to initialize our search and provide possible local changes that we consider throughout our algorithm. Through the use of these methods, we limit the search space and are able to decrease the computational complexity of the search. Following the simulation framework set out by Nowzohour et al. we have seen that in practice our algorithm optimizes BIC well without overfitting and shows promise in estimating true causal effects. As noted by Nowzohour et al., uncovering causal effects is a tough problem in general and we believe these results are promising.

REFERENCES

[ASP23] Raj Agrawal, Chandler Squires, Neha Prasad, and Caroline Uhler. "The decamfounder: Nonlinear causal discovery in the presence of hidden variables." *Journal of the Royal Statistical Society Series B: Statistical Methodology*, **85**(5):1639–1658, Jul 2023.

[AZ15] Bryon Aragam and Qing Zhou. "Concave Penalized Estimation of Sparse Gaussian Bayesian Networks." *Journal of Machine Learning Research*, **16**, 2015.

[BNM21] R. Bhattacharya, T. Nagarajan, D. Malinsky, and I. Shpitser. "Dierentiable causal discovery under unmeasured confounding." *Proceedings of the 24th International Conference on AI and STATS*, **130**, 2021.

[BP02] Carlos Brito and Judea Pearl. "A new identification condition for recursive models with correlated errors." *Structural Equation Modeling: A Multidisciplinary Journal*, **9**(4):459–474, Oct 2002.

[CDS23] Wenyu Chen, Mathias Drton, and Ali Shojaie. "Causal structural learning via local graphs." *SIAM Journal on Mathematics of Data Science*, **5**(2):280–305, May 2023.

[CMK12] Diego Colombo, Marloes H. Maathuis, Markus Kalisch, and Thomas S. Richardson. "Learning high-dimensional directed acyclic graphs with latent and selection variables." *The Annals of Statistics*, **40**(1), Feb 2012.

[Drt08] Mathias Drton. "Iterative conditional fitting for discrete chain graph models." *COMPSTAT*, p. 93–104, 2008.

[FNM19] Benjamin Frot, Preetam Nandy, and Marloes H. Maathuis. "Robust causal structure learning with some hidden variables." *Journal of the Royal Statistical Society Series B: Statistical Methodology*, **81**(3):459–487, Mar 2019.

[KM13] Jack Kuipers and Giusi Moffa. "Uniform random generation of large acyclic digraphs." *Statistics and Computing*, **25**(2):227–242, Nov 2013.

[LSP23] Chunlin Li, Xiaotong Shen, and Wei Pan. "Nonlinear causal discovery with confounders." *Journal of the American Statistical Association*, p. 1–10, Mar 2023.

[NME17] Christopher Nowzohour, Marloes H. Maathuis, Robin J. Evans, and Peter Bühlmann. "Distributional equivalence and structure learning for bow-free acyclic path diagrams." *Electronic Journal of Statistics*, **11**(2), Jan 2017.

[RER23] Thomas S. Richardson, Robin J. Evans, James M. Robins, and Ilya Shpitser. "Nested Markov properties for acyclic directed mixed graphs." *The Annals of Statistics*, **51**(1), Feb 2023.

[RS02] Thomas Richardson and Peter Spirtes. "Ancestral graph Markov models." *The Annals of Statistics*, **30**(4), Aug 2002.

[SG91] Peter Spirtes and Clark Glymour. "An algorithm for fast recovery of sparse causal graphs." *Social Science Computer Review*, **9**(1):62–72, Apr 1991.

[SGK20] Saber Salehkaleybar, AmirEmad Ghassami, Negar Kiyavash, and Kun Zhang. "Learning Linear Non-Gaussian Causal Models in the Presence of Latent Variables." *Journal of Machine Learning Research*, **21**:1–24, 2020.

[SGS93] Peter Spirtes, Clark Glymour, and Richard Scheines. "Regression, causation and prediction." *Causation, Prediction, and Search*, p. 238–258, 1993.

[SMR99] Peter Sprites, Christopher Meek, and Thomas S Richardson. "An algorithm for causal inference in the presence of latent variables and selection bias." *Computation, Causation, and Discovery*, 1999.

[TP02] Jin Tian and Judea Pearl. "On the testable implications of causal models with hidden variables." *Uncertainty in Artificial Intelligence*, **18**:519–527, 2002.

[VP22] TS Verma and Judea Pearl. "Equivalence and synthesis of causal models." *Probabilistic and Causal Inference*, p. 221–236, Feb 2022.

[ZCP20] Chi Zhang, Bryant Chen, and Judea Pearl. "A simultaneous discover-identify approach to causal inference in linear models." *Proceedings of the AAAI Conference on Artificial Intelligence*, **34**(06):10318–10325, Apr 2020.

[Zha06] Jiji Zhang. "Causal Inference and Reasoning in Causally Insufficient Systems." Jul 2006.

[Zha08a] Jiji Zhang. "Causal Reasoning with Ancestral Graphs." *Journal of Machine Learning Research*, **9**:1437–1474, 2008.

[Zha08b] Jiji Zhang. "On the completeness of orientation rules for causal discovery in the presence of latent confounders and selection bias." *Artificial Intelligence*, **172**(16–17):1873–1896, Nov 2008.