LEARNING CAUSAL BOUNDS USING MARGINAL INDEPENDENCE INFORMATION WITH APPLICATIONS TO GENE EXPRESSION ANALYSIS

by

BORZOU ALIPOURFARD

Presented to the Faculty of the Graduate School of
The University of Texas at Arlington in Partial Fulfillment
of the Requirements
for the Degree of

DOCTOR OF PHILOSOPHY

THE UNIVERSITY OF TEXAS AT ARLINGTON

May 2022

To my mother Ladan and my brother Omid

who set the example and who made me who I am.

ACKNOWLEDGEMENTS

I want to thank Dr. Gao for trusting me and giving me the freedom to pursue my research interests. If it wasnt for her trust I wouldnt have had the confidence to overcome and push through so many difficulties. I wish to thank my academic advisors Dr. Gautam Das, Dr. Manfred Huber, Dr. Dajiang Zhu for their interest in my research and for taking time to serve in my dissertation committee.

2/7/2022

ABSTRACT

LEARNING CAUSAL BOUNDS USING MARGINAL INDEPENDENCE INFORMATION WITH APPLICATIONS TO GENE EXPRESSION ANALYSIS

BORZOU ALIPOURFARD, Ph.D.

The University of Texas at Arlington, 2022

Supervising Professor: Jean Gao

Discovering causal relations is a fundamental goal of science. Randomized controlled experiments were often considered to be the only reliable method for tackling this task. However, in recent years, various causal discovery methods have been proposed that are capable of identifying causal relations from purely observational data. While these causal discovery methods provide a theoretical framework for bridging the gap from statistical relations to causal conclusions, causal discovery remains a challenging task in practice; this challenge arises because many of the assumptions made in obtaining these theoretical results are often not met in practice. This is especially true when one considers causal discovery in the landscape of genomic data. The critical challenge in learning causal relations in genomic data concerns a marked contrast between the sample size requirements of the aforementioned causal discovery algorithms and the size of the samples obtained through genomic experiments. These causal discovery tools require at least thousands of samples for identifying small causal networks in domains with five to ten variables; in genomic data, we often face networks with hundreds of nodes while the available sample is limited to

thousands at best. Besides this factor, in genomic studies, we have to deal with measurement errors, averaging effects, and feedback loops, all of which undermine the theoretical assumptions that come at the core of any typical causal discovery algorithms. In this work, we propose a series of improvements to the available causal discovery algorithms and propose new causal discovery tools that overcome all the aforementioned challenges allowing one to learn and uncover relations of causal nature in gene expression measurements.

TABLE OF CONTENTS

LIST OF ILLUSTRATIONS

LIST OF TABLES

CHAPTER 1

INTORDUCTION

Discovering causal relations is a fundamental goal of science. Randomized controlled experiments were often considered to be the only reliable method for tackling this task. However, in recent years, various causal discovery methods have been proposed that are capable of identifying causal relations from purely observational data. While these causal discovery methods provide a theoretical framework for bridging the gap from statistical relations to causal conclusions, causal discovery remains a challenging task in practice; this challenge arises because many of the assumptions made in obtaining these theoretical results are often not met in practice. This is especially true when one considers causal discovery in the landscape of genomic data. The critical challenge in learning causal relations in genomic data concerns a marked contrast between the sample size requirements of the aforementioned causal discovery algorithms and the size of the samples obtained through genomic experiments. These causal discovery tools require at least thousands of samples for identifying small causal networks in domains with five to ten variables; in genomic data, we often face networks with hundreds of nodes while the available sample is limited to thousands at best. Besides this factor, in genomic studies, we have to deal with measurement errors, averaging effects, and feedback loops, all of which undermine the theoretical assumptions that come at the core of any typical causal discovery algorithms. In this work, we propose a series of improvements to the available causal discovery algorithms and propose new causal discovery tools that overcome all the

aforementioned challenges allowing one to learn and uncover relations of causal nature in gene expression measurements.

In chapter two we study a possible improvement to the Score based learning (SBL) algorithms for causal discovery. The initial step in the majority of the SBL algorithms consists of computing the scores of all possible child and parent-set combinations for the variables. For networks with continuous variables, a particular score is usually calculated as a function of the regression of the child over the variables in the parent-set. The sheer number of regressions models to be solved necessitates the design of efficient numerical algorithms. In chapter two, we propose an algorithm for an efficient and exact calculation of regressions for all child and parent-set combinations. In the proposed algorithm, we use QR decompositions (QRDs) to capture the dependencies between the regressions for different families and Givens rotations to efficiently traverse through the space of QRDs such that all the regression models are accounted for in the shortest path possible. We compare the complexity of the suggested method with different algorithms, mainly those arising in all subset regression problems, and show that our algorithm has the smallest algorithmic complexity. We also explain how to parallelize the proposed method so as to decrease the runtime by a factor proportional to the number of processors utilized.

While, Score based learning has proved a promising approach for learning causal networks in the discrete domain, when employing SBL in the continuous domain, one is either forced to move the problem to the discrete domain or use metrics such as BIC/AIC/BGe, and these approaches are often lacking: discretization can have an undesired impact on the accuracy of the results, and BIC/AIC/BGe can fall short of achieving the desired accuracy. In the third chapter, we introduce two new scoring metrics for scoring causal networks in the continuous domain: the three-part minimum description length and the renormalized normalized maximum likelihood metric. We

rely on the minimum description length principle in formulating these metrics. The metrics proposed are free of hyperparameters, decomposable, and are asymptotically consistent. We evaluate our solution by studying the convergence rate of the learned graph to the generating network and, also, the structural hamming distance of the learned graph to the generating network. Our evaluations show that the proposed metrics outperform their competitors, the BIC/AIC/ and BGe metrics.

In the fourth chapter we consider causal learning in the low sample regimes, where typical causal discovery algorithms, especially the ones using conditional independence tests, suffer greatly. In chapter four, we examine the possibility of causal discovery using only the pairwise dependency structure of a domain and investigate if this approach can lead to more robust and reliable solutions to causal discovery when the sample size is limited. We show that marginal dependence relations can be used to construct a novel causal discovery tool capable of distinguishing between direct and indirect causal effects and detecting a lower bound on direct causal effects. In our simulations we found that causal discovery using marginal dependency information can be significantly more accurate than causal knowledge obtained through conditional independence tests when the sample size is limited: our experiments suggests avoiding conditional independence tests can reduce error rate by up to 30 percent. Finally, using this marginal dependency based causal discovery tool, we propose four candidate genes that possibly contain the causal mutation in Major Depressive Disorder utilizing a database of only 59 samples.

Studying patterns of dependency loss has been a cornerstone heurstic when it comes to learning causal relations in gene expression studies. However, there are no theoretical results that substantiate the causal claims made in this manner. Therefore, it has not been easy to establish how well these algorithms can generalize—as causal discovery tools—beyond the very specific datasets they were designed to handle. In

chapter five, we present a theoretical framework based on Reichenbach's common cause principle that legitimizes causal discovery from dependency loss. We build a novel causal discovery tool that transforms dependency loss patterns into a set of partially directed graphs; these graphs serve as upper and lower bounds depicting the causal origins of the changes in the dependency.

CHAPTER 2

SOLVING ALL REGRESSION MODELS FOR LEARNING GAUSSIAN NETWORKS USING GIVENS ROTATIONS

2.1 Introduction

Bayesian networks (BN) are used to portray probabilistic dependencies. They are graphical models that encode a set of conditional independence statements through absence or presence of directed edges among nodes in a graph [1, 2]. The child and parent relations formed in a such a graph help capture the dependency structure of the domain. BNs have found great application in machine learning, knowledge modeling, desicion systems, and causal learning [3, 4, 5].

A significant statistical problem is to learn a BN from observational data. A promising approach for tackling this learning problem consists of a group of algorithms under the heading of score based learning algorithms (SBL) [6, 7, 8]. The initial step in the majority of the SBL algorithms consists of computing the *local* scores for all possible child and parent-set combinations [9, 10, 11]. In the case of Bayesian networks over continous domain, the local score for a particular child and parent-set pair is usually calculated as a function of the regression of the child variable over the parent-set. The sheer number of regressions models that need solving presents a computational challenge. For a network with m nodes, there are 2^{m-1} candidate parent sets for a particular node and a total of $m2^{m-1}$ candidate families (a family consisting of a particular child and parent-set pair). The exponential number of node and parent-set combinations neccesiates the need for efficient numerical strategies.

In this paper, we propose an algorithm for an efficient and exact calculation of regressions for all child and parent-set combinations of a given set of variables. The main data structure employed in the proposed algorithm is QR decomposition (QRD) as it both provides high numerical percision and can capture the dependency between the regressions for different families. Using QRD together with Givens rotations (a low-cost operation), we show how to form a sequence of R matrices that provide the necessary information required to solve the regressions for all the families. In the proposed algorithm, we find the shortest of such sequences.

We further compare the theoretical runtime of the suggested method with different algorithms, mainly those arising in all subset regression problems, and show that our algorithm has the fastest runtime. We also explain how to parallelize the proposed algorithm providing a linear speedup proportional to the number of processors utilized.

2.1.1 Previous Work

A brute strategy for calculating all regression models would be to solve the regression equations for each family independent of the other families. Given the QRD of a family with k parents, the number of flops (a flop consisting only of basic arithmetic operations such as a multipication, a division, a subtraction, or an addition) required for calculating the regression coefficients is of the order of $O(k^2)$. Forming the QRD itself requires $O(2nk^2)$ flops given n samples. Therefore, the total number of flops required for calculating the regressions for all parent-set and child pairs is:

$$O(naive) = m \sum_{k=1}^{m} (2nk^2 + k^2) C(k, m), \tag{2.1}$$

where $C(k, m)$ is the number of possible combinations of k objects out of m, and m is the number of variables.

We can achieve a faster runtime by using the covariance matrix of the data and Cholesky decompositions and form each QRD in $O(k^3/3)$. This however comes at cost of numerical accuracy [12]. Using (2.1) and replacing $2nk^2 + k^2$ with $k^3/3$, it can be shown that forming the QRDs for all parent-set and child pairs requires $O(m^2 2^{m-2})$ flops.

A better strategy would be to consider the problem of calculating the regression models for all the families as that of solving for multiple all-subset regression problems [13, 14, 15]. In other words, calculating the regression models for a BN over m variables can be thought of as m all-subset regression problems, one for each node. Using algorithms specialized to all-subset regression problems such as Clarke's algorithm or DCA, it is possible to achieve a linear speed up in solving for regression models [16, 17, 18]. However, such algorithms still solve the all-subset regression problem for each node independent of the other nodes and do not utilize the full structure of the problem. Thus we conjecture that it is possible to improve the runtime even further.

2.1.2 Givens Rotations

Reviewing the basics of Givens rotations would help us explain the workings of our proposed algorithm better. Pre-mutliplying a vector with a Givens rotation matrix, $G_\theta^{(i,j)}$, rotates the vector in the $(i, j)_{th}$ plane:

$$G_\theta^{(i,j)} = \left(\begin{array}{c:ccc:c} I & & 0 & & 0 \\ \hdashline & \cos(\theta) & & \sin(\theta) & \\ 0 & & & & 0 \\ & -\sin(\theta) & & \cos(\theta) & \\ \hdashline 0 & & 0 & & I \end{array}\right), \tag{2.2}$$

where $\cos(\theta)$ and $\sin(\theta)$ in $G_\theta^{(i,j)}$ appear at the intersections of i_{th} and j_{th} rows and columns. By choosing θ appropriately, it is possible to rotate a vector such that its j_{th} component becomes zero while preserving its norm [7].

2.1.3 Retriangularization with Givens Rotation

Consider a data matrix for the variables $\{v_1, .., v_i, v_{i+1}\ , ..., v_m\}$ and its corresponding QRD (with the same variable ordering). Having this QRD, it is then possible to compute the QRD of the data matrix with variables ordered as $\{v_1, .., v_{i+1}, v_i, ..., v_m\}$ (variables $\{i\}$ and $\{i+1\}$ are transposed in the new order) by pre-multiplying the R factor (after having its $\{i\}$ and $\{i+1\}$ columns swapped) by an appropriate Givens rotation matrix:

$$\hat{R} = G_\theta^{(i,i+1)} \begin{pmatrix} r_1, 1 & \dots & r_{1,i+1} & r_{1,i} & \dots & r_{1,m} \\ 0 & \dots & r_{2,i+1} & r_{2,i} & \dots & r_{2,m} \\ \vdots & & \vdots & \vdots & & \vdots \\ 0 & \dots & r_{i,i+1} & r_{i,i} & \dots & r_{i,m} \\ 0 & \dots & r_{i+1,i+1} & 0 & \dots & r_{i+1,m} \\ \vdots & & \vdots & \vdots & & \vdots \\ 0 & \dots & 0 & 0 & \dots & r_{m,m} \end{pmatrix}, \tag{2.3}$$

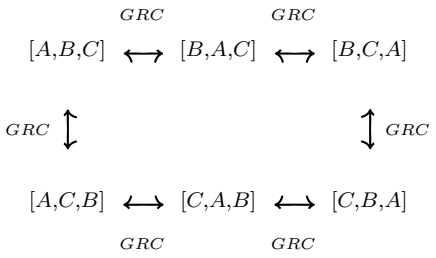


Figure 2.1: Each node represents a QRD. An edge indicates that one can calculate the QRD of a neighbouring node through using GRC operation.

where θ can be calculated using :

$$\begin{aligned} \cos(\theta) &= \frac{r_{i,i+1}}{\sqrt{(r_{i,i+1}^2 + r_{i+1,i+1}^2)}} \\ \sin(\theta) &= \frac{r_{i,i+1}}{\sqrt{(r_{i,i+1}^2 + r_{i+1,i+1}^2)}} \end{aligned} \tag{2.4}$$

Such matrix transformation requires $O(6 * (m - i + 1))$ flops. We can regard this procedure as an operator that given a starting QRD outputs another QRD by column swapping and retriangularization. We will call this operator GRC and will refer to it by $\xrightarrow{\text{GRC}}$. We use $\xrightarrow{\text{GRC}_i}$ to emphasize that column swapping occurs at index i.

2.2 Problem Definition

We will describe our problem's framework and our proposed algorithm by considering an example Bayesian network with three nodes, $\{A, B, C\}$. We use this example to construct a more general framework by the end of this section. In a BN over three variables, there are a total of nine regression models: $\{[A] \perp B, [A] \perp C, [B] \perp A, [B] \perp C, [C] \perp B, [C] \perp A, [A, B] \perp C, [A, C] \perp B, [B, C] \perp A\}$, where

$[x, y] \perp z$ represents the regression model with $\{x, y\}$ as predictor variables and z as the response variable.

For these variables, there are also six QRDs, each corresponding to a specific premutation of these variables. Each QRD can, in turn, be used to solve three regression models:

$$[A, B, C] \Rightarrow \{[A] \perp B, [A] \perp C, [A, B] \perp C\}, \tag{2.5}$$

where we have used $[A, B, C]$ for the QRD of the data matrix with variables ordered as $[A, B, C]$. We can go from one permutation to another (from one QRD to another) by using the operation introduced above and transposing two adjacent columns and retriangularization of the resulting matrix by using Givens transformation:

$$[A, B, C] \xrightarrow{\text{GRC}_2} [A, C, B]. \tag{2.6}$$

The GRC operation allows us to traverse between the six QRDs for the variables $[A, B, C]$ as shown in the graph of Fig 2.1.

Now consider the sequence of column transpositions giving rise to sequence of permutations:

$$\begin{aligned} [A, B, C] \quad &\xrightarrow{\text{GRC}_1} [B, A, C] \xrightarrow{\text{GRC}_2} [B, C, A] \\ &\xrightarrow{\text{GRC}_1} [C, B, A] \xrightarrow{\text{GRC}_2} [C, A, B]. \end{aligned} \tag{2.7}$$

Note that all 9 regression models are included in this permutation set. Therefore, it is possible to obtain all the QRDs necessary for solving all the regression models in a BN with three variables starting with an initial QRD, and forming four more by four adjacent column transpositions and application of four Givens rotations. Thus

the problem of solving for all nine regression models is reduced to performing four simple Givens rotations.

Using this example as a basis, we propose the following algorithm. The algorithm starts with calculating the QRD of the data matrix for m variables. This initial QRD can be used to solve regression models for $\frac{m(m-1)}{2}$ families. Then, through a sequence of adjacent column transpositions and retriangularizations, new QRDs are calculated. Each new QRD provides the information required to solve further regression models until all regression models are accounted for. The only remaining decision is to choose the sequence of column transpositions optimally such that minimum number of column transpositions are employed.

2.3 A Greedy Algorithm Using Givens Rotations

In our algorithm, we choose the sequence of column transpositions in a greedy manner, leading to a very simple and intuitive algorithm. The greedy choice at each step consists of chooing a column for swapping such that the number of new models specified by the new permutation is the highest. Following is the sequence of greedy column transpositions that provides a sequence of QRDs sufficient for solving all the

regressions of a BN with four variables, $\{A, B, C, D\}$ (the total number of models specified at each step is shown next to the permutations):

$$\begin{aligned}
[A, B, C, D] &\Rightarrow (6\ Models) \\
[B, A, C, D] &\Rightarrow (9\ Models) \\
[B, C, A, D] &\Rightarrow (11\ Models) \\
[C, B, A, D] &\Rightarrow (14\ Models) \\
[C, A, B, D] &\Rightarrow (16\ Models) \\
[C, A, D, B] &\Rightarrow (17\ Models) \\
[C, D, A, B] &\Rightarrow (19\ Models) \\
[D, C, A, B] &\Rightarrow (22\ Models) \\
[D, A, C, B] &\Rightarrow (24\ Models) \\
[D, A, B, C] &\Rightarrow (25\ Models) \\
[D, B, A, C] &\Rightarrow (26\ Models) \\
[D, B, C, A] &\Rightarrow (28\ Models)
\end{aligned} \tag{2.8}$$

On closer examination, one will see that the above greedy algorithm follows a recursive structure. More specifically, consider the variables $\{X_1, X_2, ..., X_{m-1}, X_m\}$. Assume that we are given the QR decomposition of the variables in the same order as written above. Note that the greedy algorithm starts at the left most position. Further, note that transposing at X_m, the last variable, at any step, leads to a permutation that only adds one single model. Therefore, a greedy algorithm can always limit its operation to transposing the first $m-1$ variables until all the $m(2^{m-2}-1)$ regression models having predictors in $\{X_1, X_2, ..., X_{m-1}\}$ are solved. When the permutations are such exhausted that transpositions in these positions add no new models, then the first transposition of X_m occurs: $\{\hat{X}_1, \hat{X}_2, ..., X_m, \hat{X_{m-1}}\}$. In the following steps, unless X_m is at first position, then no transposition on other variables is allowed

since such transpositions add no new models. This sequence continues until through transpositions on X_m, this variable comes to the first position of the permutation, $\{X_m, \hat{X}_1, \hat{X}_2, ..., \hat{X_{m-1}}\}$. At this step, a greedy algorithm for $\{X_m, \hat{X}_1, \hat{X}_2, ..., \hat{X_{m-1}}\}$ applies the same sequence of transpositions to $\{\hat{X}_1, \hat{X}_2, ..., \hat{X_{m-1}}\}$ that it previously applied to $\{X_1, X_2, ..., X_{n-1}\}$. Therefore, we can implement the proposed greedy algorithm through recursion.

In particular, assume that we have found a sequence of greedy column transpositions that generates a set of QRDs sufficient for finding all the $m(2^{m-1}-1)$ regression models for a BN with m nodes. Let us call this sequence of column transpositions $\Upsilon(m)$. The greedy sequence of column transpositions for a graph with $m+1$ nodes can then be formed in three steps. (1) $\Upsilon(m+1)$ starts with the same sequence of column transpositions as that of $\Upsilon(m)$, leading to permuatations necessary for calculation of the regressions of every node X_i over all possible parent-sets not containing X_{m+1}. (2) To calculate the regressions of nodes X_i over parent-sets containing X_{m+1}, we first move the variable X_{m+1} to the start of the ordering by applying m column transpositions. (3) The final sequence of column swapping in $\Upsilon(m+1)$ consists of swappings at indexes $i+1$ for $i \in \Upsilon(m)$. The resulting sequence of column transposition then form the sequence of greedy column transpositions that generate for finding permutaitons necessary the score table of a BN with $m+1$ nodes. The pseudocode GreedySwaps in Algorithm 1 uses this recursive structure to find the sequence of greedy swaps.

Clarke has proposed an algorithm for solving all subset regression problem with a similar recursive structure [18, 14]. There are, however, major differences between the two algorithms. First, in every QRD, we consider *all* possible combinations of regressor and predictor variables. For example, in the case of having the QRD of variables $\{A, B, C, D\}$, given in alphabetic ordering, Clarke is only concerned with the three regression models $\{[A, B, C]\perp D, [A, B]\perp D, [A]\perp D\}$, while we consider the regression

Algorithm 1 GreedySwaps(m)

Input: number of nodes in the BN m

Ouput: An array of length $2^m - m - 1$ of swapping indexes

if $m == 2$ **then**

  return [1]

else

  $\hat{\Upsilon} \leftarrow$ GreedySwaps($m-1$)

  return $(\hat{\Upsilon} : [m-1..1] : [i | i \in \hat{\Upsilon}])$

  # concatanation operator is shown by :

end if

models $\{[A, B]\perp C, [A]\perp B, [A]\perp C\}$ in addition to that of Clarke's. Furthermore, as discussed later in section 6, our algorithmic complexity analysis shows that the proposed algorithm results in a linear speed up compared to that of Clarke's.

Using the recurrence relation described above, we can find the length of the greedy sequence of column transpositions as a function of number of variables in the BN:

$$\begin{aligned} |\Upsilon(m)| &= 2|\Upsilon(m-1)| + m - 1, \\ |\Upsilon(m)| &= 2^m - 1 - m. \end{aligned} \tag{2.9}$$

2.4 Optimality of Greedy Algorithm

In this section, we show that the proposed greedy algorithm is optimal; that the number of GRC operations or column transpositions required for generating a sufficient set of QRDs for solving all the regression models of a BN using the greedy algorithm is minimal. Our proof makes use of an auxiliary problem. This problem is that of solving all subset regression problem for m predictors and one (imaginary)

response variable. We first show that our original problem is equivalent to this auxiliary problem when the only operation allowed is that of column transposition and retriangularization. Since the optimal solution for the auxiliary problem is known [6], we can then show that our solution is optimal for the original problem.

In the first step of our proof we show that the two following problems are equivalent:

(i) Problem I (Original). Forming a sequence of QRDs such that all the regression models for a BN with m variables are included in the QRD set. The constraints are that we only have access to the starting QRD with variables ordered as $[X_1, ..., X_m]$. Further, the only operation available is transposing two immediate column and retriangularization using the GRC operator.

(ii) Problem II (Auxiliary). Forming a sequence of QRDs such that all the regression models in an all subset rgression problem with m predictor variables are included in the QRD set. Again we are given a starting QRD with variables ordered as $[X_1, ..., X_m, Y]$ (Y is the response variable). We are also again constrained to only using the GRC operator.

We first show that every solution to Problem I is also a solution to problem II. Specifically, assume that we are looking to solve the regression of Y over predictor set $P = \{X_{1P}, .., X_{KP}\}$. In other words, we want a QRD where the variables, $V = \{X_1, .., X_m\}$, are ordered as $[perm_1(P), perm_2(V \backslash P)]$, and $perm_1(P)$ and $perm_2(V \backslash P)$ are some permutations of variables in the sets P and $V \backslash P$. Assume that the size of the predictor set P is smaller than m. Since we know the solution to Problem I, then we have access to QRDs that provide the solution to the regressions of node X_j, $X_j \in V \backslash P$, over all its possible parent sets. Thus, the solution to Problem I, has to provide a QRD of the form $[per\hat{m}_1(P), per\hat{m}_2(V \backslash P)]$. Choosing $perm_1(P)$ and $perm_2(V \backslash P)$ equal to $per\hat{m}_1(P)$ and $per\hat{m}_2(V \backslash P)$, we get the desired QRD. If

$P == V$, we simply choose a node X_j, and the solution to Problem I, provides access to a QRD of the form $[perm_1(V \backslash \{X_j\}), X_j]$, which is the desired QRD for solving problem II.

In the same manner, we can show that every solution to Problem II is also a solution to Problem I. Assume we wish to solve a regression for a specific parent set and node pair where the parents are in the $F = \{X_{1F}, .., X_{KF}\}$, and the node is $X_j \notin F$. In other words, we want a QRD where the variables are ordered as $[perm_1(F), perm_2(V \backslash F)]$, where $V = \{X_1, .., X_m\}$, and $perm_1(F)$ and $perm_2(V \backslash F)$ are some permutations of variables in the sets F and $V \backslash F$. Given the solution to Problem II, we have QRDs for all subset regression models. Therefore, we have access to a QRD where the first m variables are ordered as $[\hat{perm}_1(F), \hat{perm}_2(V \backslash F)]$. Choosing $perm_1(P)$ and $perm_2(V \backslash F)$ equal to $\hat{perm}_1(P)$ and $\hat{perm}_2(V \backslash F)$, we get the desired QRD. This concludes our proof that the above two problems are equivalent.

Our solution to Problem I, uses $2^m - m - 1$ column transpositions. It has been proven that the minimal number of column transpositions required for solving Problem II, is in fact $2^m - m - 1$. Therfore, the proposed greedy algorithm is optimal and no other algorithm can generate a sequence of QRDs of smaller length such that all regression models of a score table for a BN are included in the QRD set.

2.5 Algorithmic Complexity Analysis

In this section we calculate the runtime of our algorithm and compare it to three other methods for solving the regression models for all possible families of a BN. To

analyze the runtime of the proposed algorithm, we make use of the following recurrence relation for the sequence of column transpositions employed by our algorithm:

$$\begin{aligned}\Upsilon(m) = \quad & \Upsilon(m-1) : [m-1, ..., 1] \\ & : [i+1 \mid i \in \Upsilon(m-1)],\end{aligned} \tag{2.10}$$

where $\Upsilon(m)$ is the sequence of column transpositions for m variable case, $[m-1..1]$ is a sequence of column transpositions starting at index $m-1$ down to the first position, and : is concatanation operator. Noting that the number of operations required for applying Given's transformation to a matrix of size m is six more than its counterpart for a matrix of size $m-1$, we can write the following recurrence to describe the runtime of our algorithm:

$$\begin{aligned}T(m) = \quad & [T(m-1) + 6|\Upsilon(m-1)|] \\ & + \textstyle\sum_{i=1}^{m-1} 6(m-i+1) + T(m-1),\end{aligned} \tag{2.11}$$

where $|\Upsilon(i)|$ is the number of column transpositions employed by our algorithm when applied to a network with i nodes:

$$|\Upsilon(i)| = 2^i - 1 - i. \tag{2.12}$$

Combining these two equations, (2.11) and (2.12), we can find the runtime of our algorithm to be of the order:

$$T(m) = 3m2^m + 62^m - 3m^2 - 9m - 6. \tag{2.13}$$

The runtime of the proposed method for forming the QRDs of all possible parent-set and child combination is compared to that of Clakre's all subset regression algorithm [18], Dropping Column Algorithm (DCA) [17], and direct brute force using Cholesky decomposition and covariance structure in Table 2.1.

Table 2.1: Comparison of the runtime of the proposed method (in bold) to three other algorithms.

Algorithm	Runtime
Greedy Column Swapping	$\boldsymbol{O(3m2^m)}$
DCA	$O(9m2^m)$
Clarke	$O(1.5m^2 2^m)$
Brute Force	$O(0.5m^3 2^m)$

2.6 Parallelization

In order to design an efficient parallel version of the proposed algorithm, let us re-state the general problem framework introduced in section 4 so as to account for employing of multiple processors. Given p processors, we wish to find p initial QRDs and p sequences of adjacent column transpositions for each of the processors, such that the resulting variable permutations and their corresponding QRDs among all the processing cores solves for all the $m * (2^{m-1} - 1)$ regression models of a Bayesian network with m nodes.

To find the optimal performance gain using P processors, we first derive a bound on possible performance improvment in the case of all subset regression problem. Due to discussions in section 5, we know that this bound would be still in effect for the problem of forming all QRDs for all families in a BN. We then propose a near optimal parallelization scheme that achieves a performance gain close to this bound.

Assume that we have found p initial QRDs, R_i, $i = 1, ..., p$ and p sequences of column transpositions, Υ_i, $i = 1, ..., p$, for solving the all subset regression problem using p processors. We will denote the length of the sequence of column transpositions performed by processor i by $|\Upsilon_i|$. Since the p initial QRDs can at most account for

mp regression models and since a column transposition can at most add one new regression model, we have:

$$\sum_{i=1}^{p} |\Upsilon_i| > 2^m - 1 - mp. \tag{2.14}$$

Therefore we have:

$$\exists i : |\Upsilon_i| > \frac{2^m - 1 - mp}{p}. \tag{2.15}$$

Thus the number of required column transpositions when using p processors is at best of the order of $O(\frac{2^m}{p})$.

We propose a parallelization method that is a direct consequence of the following recurrence relation:

$$\begin{aligned} \Upsilon_0(m) = \quad & \Upsilon_0(m-1) : [m-1, ..., 1] \\ & : [i+1 \mid i \in \Upsilon_0(m-1)], \end{aligned} \tag{2.16}$$

where $\Upsilon_0(m)$ denotes the sequence of column transpositions performed by the single core algorithm of the previous section for the m variable case.

Consider the case where the number of processors, p, is two. In this case, we initialize the first core with the QRD of variables ordered as $v_1, ..., v_m$ and the second core with the QRD of variables ordered as $v_m, v_1 ..., v_{m-1}$. For the first processor, we choose the sequence of column transpositions equal to $\Upsilon_0(m-1)$, and for the second processor we employ the sequence of column transpositions $[i+1 \mid i \in \Upsilon_0(m-1)]$. In general, given $p = 2^k$ processors, we can use this recurrence relation k times to find the initial QRDs and the sequence of column transpositions for each processor.

More specifically let us represent each of the 2^k processors with a binary array of k-bits, mapping processor i to a binary array equivalent to its binary repressentation.

Algorithm 2 SeedPathCalculator(P_{id}, m)

Input: processor binary array code P_{id}, size of the Bayesian network m,

Output: initial permutation υ, sequence of column swapping indexes Υ

$k = m - len(P_{id})$

$\upsilon = [1..k]$

for $i = k + 1$ **to** n **do**

  if $P_{id}[m - i] == 0$ **then**

    $\upsilon.insert(0, i)$

  else

    $\upsilon.append(i)$

  end if

end for

$d = sum(P_{id})$

$\Upsilon = d + GreedySwaps(k)$

return (υ, Υ)

Then, Algorithm 2 can provide the initial permutation and the sequence of column swapping required to be performed at processor i.

The number of column transpositions performed by each processors in general is of the order of $O(2^{m-k} - 1 - (m - k))$ or $O(\frac{2^m}{p})$.

2.7 Conclusion

In this paper, we proposed an algorithm for an efficient and exact calculation of regressions for all the families of a BN. Noting that the regressions for the different families are dependent on each other, we utilized QR decomposition as a data structure for capturing these dependencies. We then used Givens rotations and column

transpositions as low-cost operations to efficiently trace a greedy path through the space of QRDs such that all the regression models are included. We showed how the proposed greedy method could be more easily implemented using recursions in section 3. In section 4 we provided a lower bound on the number of column transpositions required for solving the regressions for all the families and showed that the proposed greedy algorithm achieves this lower bound.

We further compared the runtime of our algorithm with specialized algorithms for all-subset regression problems in Table 2.1. We argued that spcialized all-subset regression algorithms and brute force algorithms do not utilize the whole of the dependency structure among the families. The faster runtime of our proposed method then proves that we make better use of the dependency structure. Although in terms of algorithmic complexity our proposed algorithm has only a constant factor of improvement compared to that of DCA, the memmory requirements of our algorithm is much lower than theirs. Specifically, the proposed algorithm utilizes a storage of size $O(m^2)$ (only a single R matrix needs to be stored at any moment) while DCA requires a storage of size $O(2^{m-3})$. In section 7 we further provided a near optimal parallelization scheme for our prosed algorithm.

CHAPTER 3

A RENORMALIZED NORMALIZED MAXIMUM LIKELIHOOD CRITERIA FOR LEARNING BAYESIAN NETWORKS

3.1 Introduction

A Bayesian network (BN) over a set of variables is a probabilistic graphical model where the dependencies between the variables are represented through a collection of edges among them in a directed acyclic graph (DAG)[19]. BNs have found extensive applications in diverse areas of engineering such as bioinformatics, image processing, and decision systems [20, 21, 4].

In simple cases, experts can design BNs using their domain knowledge. However, in most applications, this approach is impractical. It is, therefore, important to be able to learn and estimate a BN from observational data. There are two general approaches for learning a BN from data: *constraint-based learning* [22] and *score based learning* [7].

Score based learning has proved to be a promising approach for learning Bayesian networks [9, 23, 11, 10]. In score based learning, the learning of a Bayesian network from data is viewed as an optimization task. The optimization task consists of finding the network with the highest score where the candidate networks are scored with respect to the observations using a statistically suitable scoring metric. Understandably, the choice of scoring metric plays an important role in the success of score based learning algorithms.

For Bayesian networks on discrete domains, various scoring metrics have been proposed, each formulated based on different set of statistical assumptions and motiva-

tions [7, 24, 25]. For Bayesian networks over continuous domains, however, there are very few scoring criteria. Therefore, to employ score based learning in the continuous domain, one is most often either forced to use the BIC/AIC/BGe metrics or to convert the variables into discrete counterparts. These approaches, however, are often lacking.

The most common discretization method used in practice is to heuristically partition the range of the continuous features into several mutually exclusive and exhaustive regions. Doubtless, the choice of discretization policy can have a significant undesired impact on the accuracy of the results [26]. To minimize the unfavorable effects of discretization, Friedman et al. proposed a more principled discretization policy where discretization of a continuous feature is based on its relation to other variables in the network [27]. However, the scoring metric resulting from this discretization policy is not decomposable. Lack of decomposability makes the search for the highest scoring network computationally challenging as most search algorithms require the decomposability of the scoring metric. In the absence of discretization, there are only three scoring metrics that are directly applicable to BNs on domains containing continuous variables; the AIC, BIC and the BGe score [28, 29, 30].

In this paper, we describe two novel scoring metrics for Bayesian networks in the continuous domain based on the minimum description length principle (MDL). Our work draws inspiration from the use of the MDL principle in problems of variable selection in regression [31, 32]. Thus we expect it to carry the advantages that MDL offers versus the AIC and BIC in the problems of variable selection in regression to the problem of finding a suitable Bayesian network [33]; learning a Bayesian network can be thought of as selecting predictors for a set of variables where the predictors are constrained to follow a particular order. Both scoring metrics proposed here are free of hyperparameters, decomposable, and are asymptotically consistent.

In the next section, we formally introduce the Bayesian Gaussian networks. We then formulate a crude three-part code scoring metric for Bayesian networks in part 3 of our paper. In section 4, we propose a renormalized normalized maximum likelihood scoring scheme for continous domain Bayesian networks under the assumption that the model class under consideration consists of only Bayesian Gaussian networks. Afterward, we study the asymptotic properties of the two proposed scoring metrics in section 5. We evaluate and compare the performance of the proposed scoring metrics to the BIC, AIC, and the BGe metrics using simulated data in section 6. Our results suggest that the scoring metrics proposed here consistently perform better than the other metrics for both sparse and dense graphs.

3.2 Gaussian Networks

Throughout this paper, we consider a domain, $Xm = x_1, x_2, ..., x_m$ of m continuous variables. A Bayesian network over X_m is a pair $B = (B_P, B_S)$. B_S is usually displayed through a directed acyclic graph, with nodes corresponding to random variables in X_m, and its edges encode a set of conditional independence statements. We use Pa_i to denote the parents of a node x_i as specified by B_S. B_P is a set of local conditional probabilities associated with B_S: $P(x_i|Pa_i), i = 1, , m$. Together, they represent a joint distribution over the variables in X_m, $P(\vec{(}x))$, through the factorization:

$$P(\vec{(}x)) = \prod_{i=1}^{i=m} P(x_i|Pa_i) \tag{3.1}$$

Assuming that the joint probability distribution function of $\vec{x}$ is a multivariate normal distribution, we can write:

$$\rho(\vec{x}) = \eta(\vec{\mu}, \Sigma^{-1}) = (2\pi)^{-m/2}|\Sigma|^{-1/2}e^{(\vec{x}-\vec{\mu})'\Sigma^{-1}(\vec{x}-\vec{\mu})}, \tag{3.2}$$

where $\vec{\mu}$ is the m-dimensional mean vector, Σ is the $m \times m$ covariance matrix, $|.|$ is the determinant operation, and $(.)'$ is the transpose operation. This distribution can be factorized into the product of m conditional distributions:

$$\begin{aligned} \rho(\vec{x}) &= \prod_{i=1}^{i=m} \rho(x_i|x_1, ..., x_{i-1}) \\ &= \prod_{i=1}^{m} \eta(\mu_i + \sum_{j=1}^{i-1} b_{ij}(x_j - \mu_j), 1/\tau_i), \end{aligned} \tag{3.3}$$

where τ_i is the residual variance of the node x_i, μ_i is the unconditional mean of x_i, and b_{ij} is a measure of the extent of partial correlation between nodes x_i and x_j . Such a distribution corresponds to a Bayesian network (B_P, B_S) if:

$$\begin{aligned} \forall i : \rho(x_i|x_1, ..., x_{i-1}) &= \rho(x_i|\pi_i) \\ &= \eta(\mu_i + \sum_{x_j \in \pi_i} b_{ij}(x_j - \mu_j), 1/\tau_i), \end{aligned} \tag{3.4}$$

where π_i's correspond to the parent sets specified in B_s. In other words, a multivariate Gaussian distribution corresponds to a BN, (B_P, B_S), if $\forall b_{ij} \neq 0 : x_j \in \pi_i$ [34]. The Bayesian network is minimal when there is an arc from x_j to x_i if and only if $b_{ij} \neq 0$. This network is also referred to as the I-map of the probability distribution.

Instead of the above parametrization of a multivariate Normal distribution, we opt to work with the following parametrization:

$$\forall i : \rho(x_i|x_0 = 1, \pi_i) = \eta(\sum_{x_j \in \pi_i} \beta_{ij} x_j + \beta_{i0}, 1/\tau_i). \tag{3.5}$$

Every instantiation of parameters of one model corresponds to an instantiation of parameters of the second model; note that the new parametrization corresponds to a network with $m + 1$ nodes with $x_0 = 1$, and the unconditional means of all other nodes set to zero. Thus a Bayesian network, $B = (B_P, B_S)$, can be parametrized by $\{(\vec{\beta_i}, \tau_i)|i = 1, 2, ..., m\}$. We call such a Bayesian network, a Gaussian network; if further, the network is minimal, we call it a minimal Gaussian network. In the rest of

the paper, for notational convenience, we represent the set $\{\pi_i, x_0\}$ simply by π_i. We also write k_i for the cardinality of the π_i, $|\pi_i|$, and Pa_i for the values of the variables π_i.

3.3 A Crude Three-Part Code Scoring Metric

In this section, we propose our first scoring metric for Bayesian networks based on the MDL principle. We assume that the reader is familiar with the basics of MDL principle [35, 36]. In short, MDL views learning as a data compression task and states that the model most suitable for a given set of observations is the one that provides the shortest description for the observed data. Since the description method accounts for both the model complexity and its goodness-of-fit, MDL provides a mathematical framework that embodies a form of Occams Razor

We wish to measure how well a Bayesian network structure, B_S fits the observed data. Motivated by the MDL principle, we can alternatively evaluate how compact a description a Bayesian network structure can provide for the observations. We propose two methods for encoding observations using Gaussian networks. The first method proposed in this section is a crude and a heuristic encoding scheme. In the next section of the paper, we optimize this coding scheme to formulate a min-max optimal encoding of the observations known as normalized maximum likelihood coding [33].

The three-part code encodes the observations in three steps. The first part of the code describes the Bayesian network structure and the second and the third part encode the observations using the network structure coded in the first part. The total description length then serves as a measure of how well the network fits the observations.

In this three-part code, we extend the MDL formulations of Lam et al. for Bayesian networks over discrete domains to continuous domains [37]. First, to encode

the structure of the network, we simply enumerate the parents of each node. For a node with k_i parents we use $k_i \ln m$ nats to list its parents. Therefore, the encoding length of the structure of the network can be written as:

$$L_1 = L(B_S) = \sum_{i=1}^{m} k_i \ln(m). \tag{3.6}$$

Having coded the network structure, we now describe how we encode the observations in the remainder of the code. Assume that we have observed the data $y^n = [x_1^n, ..., x_m^n]$ of sample size n. Our coding scheme is to first encode the values of the root nodes (nodes without parents) and then to encode the values of nodes whose parent values has already been encoded. We continue this process iteratively, descending the Bayesian network until we reach the leaves of the network.

Now, suppose that we have already encoded the values for $\{x_1^n, ..., x_{i-1}^n\}$ and we wish to encode the values of x_i. Since we have assumed that data is sampled from a multivariate normal distribution, we have:

$$\rho(x_i^n | x_1^n, ..., x_{i-1}^n) = (2\pi\tau_i)^{-n/2} e^{\frac{-||x_i^n - Pa_i^n \vec{\beta}_i||^2}{2\tau_i}}, \tag{3.7}$$

where Pa_i^n is the $n \times k_i$ matrix of the values of the parents of x_i. Within this framework, the problem of optimal encoding of x_i^n and the parameters $(\vec{\beta}_i, \tau_i)$ is equivalent to the problem of encoding a response variable given the values of predictor variables in linear regression. Here, the response variable is x_i, and the predictor variables are π_i. Following works of Rissanen, in such a setting, the shortest code for encoding the values of x_i has a length of [38, 35]:

$$\begin{aligned} L_2 = L(x_i^n | x_1^n, ..., x_{i-1}^n) &= L_{21} + L_{22} \\ &= \frac{k_i}{2} \ln n - \ln \rho(x_i^n | Pa_i^n; \hat{\vec{\beta}}_i, \hat{\tau}_i) \\ &= \frac{k_i}{2} \ln n + \frac{n}{2} \ln (2\pi e \hat{\tau}_i), \end{aligned} \tag{3.8}$$

where $\hat{\tau}_i$ and $\hat{\vec{\beta}}_i$ are the maximum likelihood estimates (MLE) of τ_i and $\vec{\beta}_i$. More specifically, we encode the values of x_i^n in two parts; in the first part we encode the MLE parameters $\hat{\vec{\beta}}_i$, and in the second part we encode the values of x_i^n using the distribution $\rho(x_i^n|\pi_i^n;\hat{\vec{\beta}}_i,\hat{\tau}_i)$. Such a coding scheme is referred to as a crude two part coding in the MDL literature [33].

Thus the total description length of the observed data will be:

$$\begin{aligned} L(x_1,...,x_m|x_0,B_S) =& \frac{n}{2}\sum_{i=1}^{m}\ln\left(2\pi e\tau_i(\hat{B}_S)\right)+ \\ & [\ln(n)/2+\ln(m)]\sum_{i=1}^{m}k_i(B_S), \end{aligned} \tag{3.9}$$

where we have emphasized the dependence of k_i and $\hat{\tau}_i$ on the network structure B_S.

It is insightful to compare this coding metric to BIC for Gaussian networks and the MDL metric proposed for BNs over discrete domains. Comparing the penalty terms in the proposed MDL scoring metric for Gaussian networks with the MDL scoring metric of discrete networks, one observes that the penalty term for discrete networks is exponential in the number of parents while it only grows linearly for Gaussian networks. The reason is that the dimensionality of the parameter space for discrete Bayesian networks increases exponentially with an increasing number of parents while the parameter space of multivariate Gaussian distribution is polynomial in the number of parents. The proposed MDL metric is essentially the same as the BIC metric with the addition of the penalty term $\sum_{i=1}^{m}k_i\ln(m)$ which accounts for the network structure, B_S. The similarity between the BIC metric and the heuristic MDL metric has also been previously recognized in the case of problems of variable selection in regression. In the next section we show that this heuristic coding scheme is not optimal (this can also be easily seen by considering that the coding of the network structure and the corresponding penalty term in the metric is rather arbitrary). We then propose a min-max scoring metric known as normalized maximum likelihood

coding that alleviates many of the problems associated with the heuristic coding scheme proposed in this section.

3.3.1 Inefficieny of Two Part Codes

The coding scheme introduced above is not Kraft-tight [36]. In particular, consider the code proposed for encoding values of x_i^n given values of $\{x_1^n, ..., x_{i-1}^n\}$:

$$\begin{aligned} L(x_i^n | x_1^n, ..., x_{i-1}^n) &= L_{21} + L_{22} \\ &= \frac{k_i}{2} \ln n + \frac{n}{2} \ln (2\pi e \hat{r_i}). \end{aligned} \tag{3.10}$$

Note that once we decode L_{21} (L_{21} contains information on the MLE values for the regression coefficients ($\hat{\boldsymbol{\beta}}_{\boldsymbol{i}}$)) the set of possible values for x_i^n become restricted to those for which $\hat{\beta}_i(x_i^n)=(\hat{\boldsymbol{\beta}}_{\boldsymbol{i}})$. Therefore, this coding scheme is inefficient and the data can be coded using fewer bits. Normalized maximum likelihood (NML) codes are a variation of the two-part coding scheme where this inefficiency of the crude two-part coding is addressed [36, 39].

3.4 Normalized Maximum Likelihood

The normalized maximum likelihood distribution with respect to a class of probability distributions parametrized by a K dimensional parameter vector θ, $C_\theta(x) = \{P(x;\theta) | \theta \in R^K\}$, is defined as:

$$P_{nml}(x) = \frac{P(x; \hat{\theta}(x))}{\int P(y; \hat{\theta}(y)) dy}, \tag{3.11}$$

where

$$\hat{\theta}(x) = \underset{\theta}{argmax} \left[P(x;\theta) \right], P(x;\theta) \in C_\theta(x). \tag{3.12}$$

In normalized maximum likelihood codes, instead of using a three-part code, we encode each observation with a single code using the NML distribution:

$$L_{nml}(x) = -\ln (P_{nml}(x)). \tag{3.13}$$

We are now ready to formulate the NML pdf for a Gaussian network.

A Gaussian network structure over m variables defines a class of probability distributions parametrized by $\theta = \{(\vec{\beta}_i, \tau_i)|i = 1, ..., m\}$:

$$\begin{aligned} G_\theta(y^n) &= \{\rho(y^n;\theta)|\theta \in R^{m+\sum_{i=1}^m k_i}\}, \\ \rho(y^n;\theta) &= \rho(x_1^n, ..., x_m^n;\theta) = \prod_{i=1}^m \rho(x_i^n|\pi_i^n;\vec{\beta}_i,\tau_i), \end{aligned} \tag{3.14}$$

where:

$$\begin{aligned} \rho(x_i^n|\pi_i^n;\vec{\beta}_i,\tau_i) &= \eta(\vec{\beta}_i Pa_i^n, 1/\tau_i) \\ &= (\frac{1}{2\pi\tau_i})^{n/2} \exp(\frac{1}{2\tau_i}||x_i^n - Pa_i^n\vec{\beta}_i||^2). \end{aligned} \tag{3.15}$$

Let $\hat{\theta}(y^n) = \{(\hat{\vec{\beta}}_i(y^n), \hat{\tau}_i)|i = 1, ..., m\}$ denote the MLE estimates of $\vec{\beta}_i$ and τ_i:

$$\begin{aligned} &\hat{\vec{\beta}}_i(y^n) = \hat{\vec{\beta}}_i(x_i^n, Pa_i^n) = (n\Sigma_i)^{-1} Pa_i^{n'} x_i^n, \\ &\Sigma_i = n^{-1} Pa_i^{'} Pa_i, \\ &\hat{\tau}_i(y^n) = \hat{\tau}_i(x_i^n, Pa_i^n) = 1/n||x_i^n - Pa_i^n\hat{\vec{\beta}}_i||^2, \\ &\rho(x_i^n|Pa_i^n;\hat{\vec{\beta}}_i(y^n), \hat{\tau}_i(y^n)) = (2\pi e\hat{\tau}_i(y^n))^{-n/2}. \end{aligned} \tag{3.16}$$

Using (3.16) the numerator in the expression of NML distribution can be easily calculated. However, the integral in the denominator of the NML distribution does not exist for $G_\theta(y^n)$ [40]. We write down the constrained NML density as below:

$$P_{nml}(x;\theta^0) = \frac{P(x;\hat{\theta}(x))}{\int_{Y(\theta^0)} P(y;\hat{\theta}(y))dy}, \tag{3.17}$$

the constrained NML density is only defined for $y^n \in Y(\theta^0)$:

$$Y(\theta^0) = \{y^n|\hat{\theta}(y^n) \in \theta^0\}, \tag{3.18}$$

The constrained NML density can be thought of as the conditional distribution $P_{nml}(y^n|y^n \in Y(\theta^0))$. In the case of $G_\theta(y^n)$, we propose to specify θ^0 using the following set of hyperparameters:

$$\begin{aligned} &\theta^0 = (\tau^0, R^0), \\ &\tau^0 = \{\tau_i^0|i = 1, ..., m\}, \\ &R^0 = \{R_i^0|i = 1, ..., m\}. \end{aligned} \tag{3.19}$$

Let these hyperparameters define the $Y(\theta^0)$ as below:

$$\begin{aligned} Y(\theta^0) &= Y(\tau^0, R^0) \\ &= \{y^n : x_1^n \in X_1(\tau_1^0, R_1^0), ... \\ &, x_m^n \in X_m(\tau_m^0, R_m^0, x_1^n, ..., x_{m-1}^n)\}. \end{aligned} \tag{3.20}$$

and

$$\begin{aligned} X_i(\tau_i^0, R_i^0, x_1^n, ..., x_{i-1}^n) = \{x_i^n | \hat{\tau}_i(x_i^n, Pa_i^n) \geq \tau_i^0, \\ \hat{\vec{\beta}}_i'(x_i^n, Pa_i^n)\Sigma_i \hat{\vec{\beta}}_i(x_i^n, Pa_i^n) \leq R_i^0\}, \end{aligned} \tag{3.21}$$

The numerator of 3.17 can be easily calculated as:

$$\begin{aligned} \rho(y^n; \hat{\theta}(y^n)) &= \prod_{i=1}^{m} \rho(x_i^n | Pa_i^n; \hat{\tau}(x_i^n, Pa_i^n), \hat{\vec{\beta}}_i(x_i^n, Pa_i^n)) \\ &= \prod_{i=1}^{m} (2\pi e \hat{\tau}(x_i^n, Pa_i^n))^{-n/2}. \end{aligned} \tag{3.22}$$

We can calculate the denominator by first writing down the factored form of the density:

$$\begin{aligned} \int_{Y(\theta^0)} \rho(y^n; \hat{\theta}(y^n)) dy^n = \int_{X_1(\tau_1^0, R_1^0)} ... \\ \int_{X_m(\tau_m^0, R_m^0, Pa_i^n)} \prod_{i=1}^{m} \rho(x_i^n | Pa_i^n; \hat{\vec{\beta}}_i(y^n), \hat{\tau}_i(y^n)) dx_1^n ... dx_m^n. \end{aligned} \tag{3.23}$$

Since only the factor $\rho(x_m^n | Pa_m^n; \hat{\vec{\beta}}_m(y^n), \hat{\tau_m}(y^n))$ is a function of x_n^m, we can take the other factors out of the last integral. We now write down this last integral for a given value of $\{x_1^n, ..., x_{m-1}^n\}$:

$$\int_{X_m(\tau_m^0, R_m^0, Pa_i^n)} \rho(x_m^n | Pa_i^n; \hat{\vec{\beta}}_i(y^n), \hat{\tau}_i(y^n)) dx_m^n. \tag{3.24}$$

Note that both the region of integration and the MLE estimates are functions of $\{x_1^n, ..., x_{m-1}^n\}$. Using sufficient statistics, the value of this integral was calculated in [41] :

$$\begin{aligned} C^m(\tau^0, R^0) &= \int_{X_m(\tau_m^0, R_m^0, Pa_i^n)} \rho(x_m^n | Pa_i^n; \hat{\vec{\beta}}_i(y^n), \hat{\tau}_i(y^n)) dx_m^n \\ &= \frac{4n^{n/2} (\frac{R_m^0}{\tau_m^0})^{-k_m/2}}{(2e)^{n/2} k_m^2 \Gamma(n - k_m) \Gamma(k_m/2)}. \end{aligned} \tag{3.25}$$

Note that the above factor is independent of the values of $\{x_1^n, ..., x_{m-1}^n\}$. This independence does not arise from chance. Our proposed parametrization of Guassian networks and also that of contraining hyperparameters were designed with this purpose in mind. As we will see later, this independence will make way for a scoring metric that is decomposable and local to the nodes of the graph. As we had previously mentioned, the decomposablity of scoring metric is essential to SBL algorithms.

The denominator of the constrained NML distribution is then:

$$C(\tau^0, R^0) = \prod_{i=1}^{m} \frac{4n^{n/2}(\frac{R_i^0}{\tau_i^0})^{-k_i/2}}{(2e)^{n/2} k_i^2 \Gamma(n-k_i)\Gamma(k_i/2)}. \tag{3.26}$$

In the expression above, due to the terms $(\frac{R_i^0}{\tau_i^0})^{-k_i/2}$, the hyperparameters (τ^0, R^0) have different effects on the score of different networks structures. In the case of problems of variable selection in regression, to get rid of similar undesired effects of the constraining hyperparameters, Rissanen proposed a second level normalization [39]. In the next section, borrowing from this principle, we calculate a more robust scoring metric for Gaussian networks.

3.4.1 Renormalized Normalized Maximum Likelihood Scoring Metric

Let $\hat{\tau}^0(y^n) = \{\hat{\tau}_i^0(y^n) | i = 1, 2, ..., m\}$ and $\hat{R}^0(y^n) = \{\hat{R}_i^0(y^n) | i = 1, 2, ..., m\}$ denote the MLE estimates of τ^0 and R^0 :

$$\begin{aligned} \hat{\tau}_i^0(y^n) &= \hat{\tau}_i(x_i^n, Pa_i^n) = 1/n ||x_i^n - Pa_i^n \hat{\vec{\beta}}_i||^2, \\ \hat{R}_i^0(y^n) &= \hat{R}_i(x_i^n, Pa_i^n) = \hat{\vec{\beta}}_i(x_i^n, Pa_i^n)' \Sigma_i \hat{\vec{\beta}}_i(x_i^n, Pa_i^n), \end{aligned} \tag{3.27}$$

where

$$\hat{\vec{\beta}}_i(x_i^n, Pa_i^n) = (n\Sigma_i)^{-1} Pa_i^{n'} x_i^n. \tag{3.28}$$

The renormalized NML (RNML) probability distribution is then given by:

$$\bar{\rho}(y^n) = \frac{\rho_{nml}(y^n; \hat{\tau}^0(y^n), \hat{R}^0(y^n))}{\int_{Z(\tau^1, \tau^2, R^1, R^2)} \rho_{nml}(z^n; \hat{\tau}^0(z^n), \hat{R}^0(z^n)) dz^n}, \tag{3.29}$$

where the region of integration is given by the hyperparameters:

$$\begin{aligned}\tau^1 &= \{\tau_i^1 | i = 1, 2, ..., m\} \\ \tau^2 &= \{\tau_i^2 | i = 1, 2, ..., m\} \\ R^1 &= \{R_i^1 | i = 1, 2, ..., m\} \\ R^2 &= \{R_i^2 | i = 1, 2, ..., m\}\end{aligned} \tag{3.30}$$

with $Z(\tau^1, \tau^2, R^1, R^2)$ defined as:

$$\begin{aligned}Z(\tau^1, \tau^2, R^1, R^2) = \{z^n | \forall i = 1, 2, ..., m : \\ \tau_i^2 \geq \hat{\tau_i}^0(z^n) \geq \tau_i^1, R_i^1 \geq \hat{R}_i^0(z^n) \geq R_i^2\}\end{aligned} \tag{3.31}$$

Inserting the density for the NML distribution into (3.29), with the boundry conditions above, the RNML distribution can be calculated as:

$$\bar{\rho}(y^n) = \prod_{i=1}^{m} \frac{(\hat{\tau}_i)^{-n/2}(n\pi)^{-n/2}\Gamma(k_i/2)\Gamma(n-k_i)(\frac{\hat{R}_i^0(y^n)}{\hat{\tau}_i^0(y^n)})^{-k_i/2}}{\ln(\frac{\tau_i^2}{\tau_i^1}\ln(\frac{R_i^2}{R_i^1}))}, \tag{3.32}$$

where we have used the RNML calculations for Gaussian distribution from [41], together with the property of our parametriziation that allows the integrals to be calculated independent of each other in solving the RNML distribution. Dropping the terms independent of the network structure, the RNML code can be written as:

$$\begin{aligned}&L_{RNML}(y^n) = -\ln(\bar{\rho}(y^n)) \\ &= \sum_{i=1}^{m}\left(\frac{n}{2}\ln(\hat{\tau}_i(y^n)) - \ln(\Gamma(\frac{k_i}{2})) - \ln(\Gamma(\frac{n-k_i}{2})) + \frac{k_i}{2}\ln(\frac{\hat{R}_i^0(y^n)}{\hat{\tau}_i(y^n)})\right).\end{aligned} \tag{3.33}$$

Using Stirling's approximation, we can simplify the above expression:

$$\begin{aligned}&L_{RNML}(y^n) = -\ln(\bar{\rho}(y^n)) \\ &= \sum_{i=1}^{m}\left((n-k_i)\ln(\frac{\hat{\tau}_i(y^n)}{n-k_i}) + k_i\ln(\frac{\hat{R}_i^0(y^n)}{k_i}) + \ln(k_i(n-k_i))\right)\end{aligned} \tag{3.34}$$

Equations (3.34) and (3.33) provide a closed-form expression for the scoring of a Gaussian Bayesian network based on the RNML metric. Note that both these expressions are free of hyperparameter and are decomposable.

3.5 Asymptotic Behavior

It is well known that BIC prefers minimal I-maps over other network structures for large sample sizes [25]. Examining the equation (3.9), it is clear that the asymptotic behavior of the three-part coding metric is equivalent to that of the BIC metric. We now show that the RNML scoring metric also prefers networks that are minimal I-maps.

Theorem 1. Let X_m be a set of variables, T be an ordering on the variables in X_m and ρ be a probability distribution over X_m. Let y^n be a sample generated from ρ. Let B_s be a minimal I-map Bayesian network of ρ and let $B_{s'}$ be any other network structure. Furthermore, let both $B_{s'}$ and B_s be consistent with the ordering T. We have:

$$L_{RNML}(B_s, y^n) < L_{RNML}(B_{s'}, y^n) \tag{3.35}$$

That is the network corresponding to the minimal I-map has the lowest description length based on the RNML distribution.

Proof. We consider two cases. In the first case, we assume that $B_{s'}$ is a non-minimal I-map of P. In the second case, we consider a $B_{s'}$ that is not an I-map of P.

Assuming that $B_{s'}$ is a non-minimal I-map of P, then the variance of the residual, τ_i, of each node is the same in both $B_{s'}$ and B_s. Dropping the factors less than $O(n)$ we have:

$$\begin{aligned} &L_{RNML}(B_s, y^n) - L_{RNML}(B_{s'}, y^n) \\ &= -\sum_{i=1}^{m} -\ln(\Gamma(\frac{n-k_i^s}{2})) + \ln(\Gamma(\frac{n-k_i^{s'}}{2})). \end{aligned} \tag{3.36}$$

Since $B_{s'}$ is a non-minimal I-map of P we also have:

$$\forall i : k_i^s \leq k_i^{s'}. \tag{3.37}$$

Therefore, we have:

$$L_{RNML}(B_s, y^n) - L_{RNML}(B_{s'}, y^n) \leq 0 \tag{3.38}$$

Now assume that $B_{s'}$ is a non-minimal I-map of P. Thus there exists at least one node, x_i, such that $\pi_i^s \not\subseteq \pi_i^{s'}$ or equivalently, $\exists x_j \in \pi_i^s$ such that $x_j \notin \pi_i^{s'}$. Without loss of generality, we assume that this consists of the only difference between the two networks. Now consider a network structure, B_c, that is exactly equal to network B_s except for the parent set of the node x_i where $\pi_i^c = \pi_i^s \cup \pi_i^{s'}$. Therefore, B_c is a non-minimal I-map of the probability distribution and from the previous result we know that $L_{RNML}(B_s, y^n) \leq L_{RNML}(B_c, y^n)$. We now show that $L_{RNML}(B_c, y^n) \leq L_{RNML}(B_{s'}, y^n)$.

We can transform the network B_c to the network $B_{s'}$ by removing the nodes $\{x_k : x_k \in \pi_i^s, x_k \notin \pi_i^{s'}\}$ from $\pi_i^{s'}$. Doing so will increase the residual variance of the node x_i by at least $\beta_{ik}^2 \hat{\tau}_k(y^n)$. More specifically, using equation 3.34 dropping the node x_k will increase the $L_{RNML}(B_c, y^n)$:

$$\begin{aligned} &\Delta L_{RNML} = \\ &(n-k_i^c) ln(\frac{\hat{\tau}_i^c(y(n))}{n-k_i^c}) - (n-k_i^c-1) ln(\frac{\hat{\tau}_i^c(y(n)) - \beta_{ik}^2 \hat{\tau}_k(y^n)}{n-k_i^c-1}). \end{aligned} \tag{3.39}$$

Keeping only the terms of the factor $O(n)$, we can simplify the above expression:

$$\begin{aligned} \Delta L_{RNML} &= (n-k_i^c) ln \left(\frac{\frac{\hat{\tau}_i^c(y(n))}{\hat{\tau}_i^c(y(n)) - \beta_{ik}^2 \hat{\tau}_k^c(y^n)}}{\frac{n-k_i^c}{n-k_i^c-1}} \right) \\ &\overset{n\to\infty}{=} (n-k_i^c) ln \left(\frac{\hat{\tau}_i^c(y(n))}{\hat{\tau}_i^c(y(n)) - \beta_{ik}^2 \hat{\tau}_k^c(y^n)} \right) \\ &= O(n). \end{aligned} \tag{3.40}$$

Hence we can see that asymptotically as $n \to \infty$, $L_{RNML}(B_c, y^n)$ is smaller than $L_{RNML}(B_{s'}, y^n)$ by a factor of $O(n)$.

3.6 Numerical Evaluation

We evaluate and compare the performance of the proposed scoring metrics to the BIC, AIC, and BGe metrics using simulated data [1]. We study the properties of

[1]For setting the prior parameters for BGe, we used the first fifth of the data to estimate the parameters T and v using the equations (18) and (19) of [42] and we set the equivalent sample size for both T and v, equal to the number of samples used in their estimation.

these four metrics on two levels: a low dimensional setting where the number of the nodes of the graph is small, $m \in \{4, 5\}$, and lager graphs having $m \in \{8, 10\}$ nodes. In low dimensional setting, the number of possible generating graphs is limited. It is, therefore, possible to evaluate the performance of these four metrics in detail since we can calculate the score of all the possible generating networks. In this setting, we examine how the true generating network structure is ranked among all other networks by the different scoring criteria.

With larger DAGs, the number of possible generating graphs is exponentially high and such evaluation of the performance of the scoring metrics is computationally challenging. Hence, we decided to compare the performance of these metrics by using a structural distance measure between the highest scoring networks (we call these networks prime DAGs and use dynamic programming [10] to find them) and the true generating network.

We chose Structural Hamming Distance (SHD) as our measure of performance. SHD is a function of the number of edge addition/deletion or reversals required to convert one DAG to another [43]. While such a structural distance measure can only provide a heuristic summary of the performance of these metrics, nevertheless, a comparison in terms of such structural errors has a desirable intuitive interpretation. Furthermore, SHD compares the similarity of BNs in a causal context [43, 44]. Therefore, it also serves as a tool to examine the applicability of the proposed metrics in identifying the generating causal structure.

3.7 Performance in Low Dimensional Setting

In this experiment, we compare how the generating network structure is ranked among all the possible networks by the different scoring criteria.

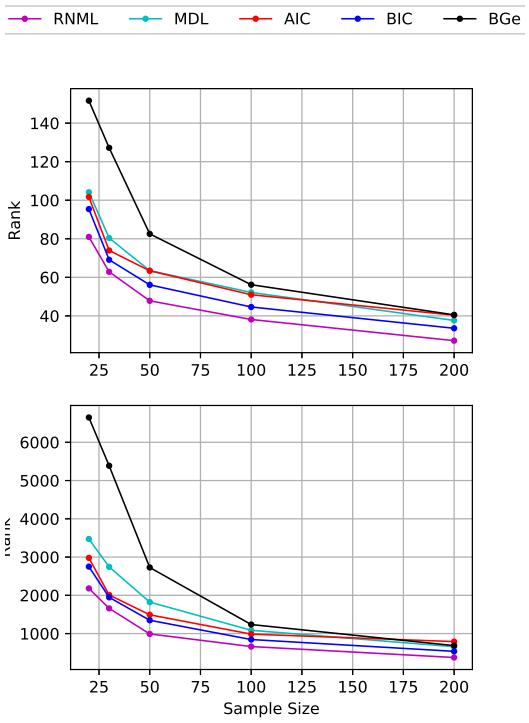


Figure 3.1: The average rank of the generating network structure for networks having m nodes, $m \in \{4, 5\}$, over 500 iterations plotted against the sample size. The upper plot shows the results for networks having four nodes, while the convergence rate for networks having five nodes is shown in the lower plot.

First, a random DAG was chosen among all DAGs on m nodes, $m \in \{4, 5\}$. A sample was then recursively generated from the selected DAG starting from the root node down to the leaves using the following equation:

$$x_i = \mu_i + \sum_{x_i \in \pi_i} b_{ij}(x_j - \mu_j) + \eta(0, 1/\tau_i) \tag{3.41}$$

where network parameters μ_i, τ_i were sampled from a uniform distribution $\mathcal{U}(0.1, 1)$, and b_{ij} was chosen as independent realizations of $\mathcal{U}(-1, 1)$. Afterwards, we computed the scores of all possible DAGs for the simulated data and calculated the

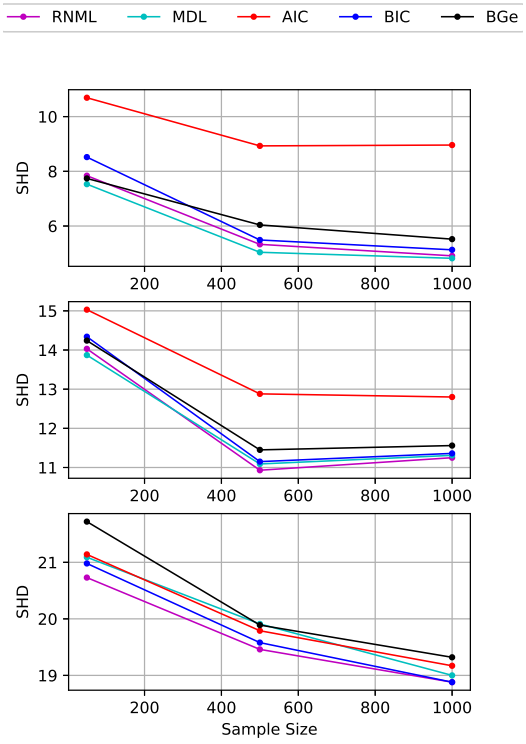


Figure 3.2: Average SHD between the generating DAG and the prime DAG in simulations with $m = 8$. The upper plot shows the results for graphs where the expected number of neighbours for each node was set to $nn = 2$. The middle and the lower plots show the results for $nn = 4$ and $nn = 6$.

rank of the generating DAG among all other DAGs. The simulation was repeated for 500 iterations. Figure 3.1 shows the average rank of the generating network structure for different scoring metrics as a function of the sample size.

As is shown in Figure 3.1, the convergence rate of the rank is the fastest for the RNML metric. The MDL metric and the BIC metric show similar performance while the AIC metric has the lowest convergence rate.

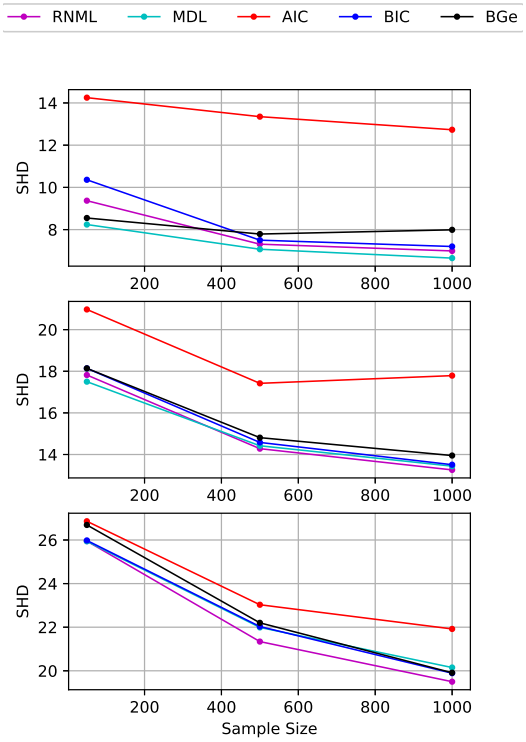


Figure 3.3: Average SHD between the generating DAG and the prime DAG in simulations with $m = 10$. The upper plot shows the results for graphs where the expected number of neighbours for each node was set to $nn = 2$. The middle and the lower plots show the results for $nn = 4$ and $nn = 6$.

3.8 Performance For Larger Networks

In this section, we analyze the performance of the four metrics against the sparsity of the generating graph for graphs having m nodes, $m \in \{8, 10\}$. We compare the performance of these metrics by evaluating the SHD of the prime DAG to the generating DAG.

Our simulations start by selecting a random graph having a specified sparsity. Similar to the work of Kalisch et al., we simulated graphs of different sparsity by controlling the expected number of connections, neighbours, of each node, $nn \in \{2, 4, 6\}$ [45]. Thus, we extended the scope of our simulations beyond Erdos-Renyi

(uniformly random) networks and examined the performance of the metrics for networks of varying sparsity. Afterward, a dataset of sample size N, $N \in \{50, 500, 1000\}$, was simulated based on the selected generating graph similar to the previous section. To find the prime DAG, we used the optimal dynamic programming method of Silander et al. [10]. In selecting the maximum parent size parameter in the implementation of this algorithm, we chose it equal to the maximum parent size of the generating graph itself. This way, the generating graph would be included in the possible solution set of our algorithm. For every of the 27 combinations of the control parameters (the control parameters being the number of nodes, expected number of neighbours, and sample size) we repeated the simulation for 100 iterations, each time for a randomly selected generating DAG. We then computed the average SHD between the generating and the prime DAG across these 100 iterations. Results are shown in Figures 2-3.

Figures 3.2-3.3 show that the RNML metric consistently outperforms the other metrics irrespective of the size and the sparsity of the generating network. The proposed three-part MDL metric comes second in terms of performance, surpassing the AIC and the BIC metric. On average, aggregated across all sample size, sparsity, and network size values, the RNML metric outperformed the MDL metric by an SHD value of 0.6356 and the other two metrics by an SHD value of 1.8704. The average reduction in the SHD value for the RNML metric as compared to MDL and the best (smallest) of AIC/BIC is shown in Table 1 as a function of sample size.

3.9 Conclusion

In this paper, we introduced two new scoring metrics based on the MDL principle for Gaussian networks. These scores are asymptotically consistent, have simple to calculate closed-form expressions, and are parameter-free. They are furthermore decomposable and therefore compatible with the most Bayesian network search proce-

Table 3.1: The average difference between the SHD of the prime graph to the generating graph when using RNML metric compared to that of MDL and the best (smallest) of AIC/BIC.

Sample Size	MDL	BIC/AIC
50	0.7333	3.5900
500	0.6078	1.4422
1000	0.5656	0.5789

dures. Our evaluation of the proposed metrics suggests that the proposed metrics have better performance than the AIC and the BIC metrics. The proposed RNML metric specially outperforms all other metrics consistently and regardless of the size and the sparsity of the generating graph and the sample size (see Figure 3.1 and Figures 3.2 3.3). The metric proposed here is also very accessible: it is easy to code, simple to understand, and any toolbox that already implements BIC or AIC for scoring BNs can be directly modified with little code to include these new metrics.

The RNML metric proposed here can be thought of as a continuous domain extension to the FNML metric proposed for discrete Bayesian networks [24]. However, unlike the FNML metric, the RNML metric was not tuned to be decomposable, rather, decomposability came naturally to our formulation. More specifically, a major theoretical contribution of ours in this paper was to show that a global RNML formulation for a Gaussian network leads to a scoring metric that can be written as function of scores resulting from application of RNML model selection criterion applied at each local distribution.

CHAPTER 4

LEARNING A LOWER BOUND ON DIRECT CAUSAL INFLUENCES FROM MARGINAL INDEPENDENCIES

4.1 Introduction

Discovering *causal relations* is a fundamental goal of science. Randomized controlled experiments were often considered to be the only reliable method for tackling this task. However, in recent years, a variety of causal discovery methods have been proposed that are capable of identifying causal relations from purely observational data. The performance of these causal discovery tools depends on the number of recordings and of available samples. Furthermore, practice suggests that the sample size requirement of most causal learning algorithms is not modest. Consider the numerical example of [46], where the ICA (independent component analysis) uses 10000 observations for identifying a network over 7 variables, or the simulation results in [47] which, suggests one requires 1000 samples for identifying a causal DAG with 5 nodes and 7 edges through TETRAD-II and MML-CI algorithms. SBL (score based learning) algorithms also require large sample sizes: the simulations in [48] suggest that over 1000 samples are required for identifying a very sparse (average in-degree of 2.2) causal DAG (directed acyclic graph) over 15 nodes. Consequently, one must be very cautious in interpreting the results given by these causal discovery tools, especially in domains such as gene expression analysis, where the available sample size (most of the time) is on the same order as the number of features. For example, currently, one of the largest gene expression datasets on the major depressive disorder (among all those that are publicly available in GEO and AE repositories) boasts a

modest sample size of only 59 measurements of 34 differentially expressed genes. In this paper, we are concerned with causal learning in similar low sample regimes: our goal is to characterize a type of causal knowledge that can be reliably learned in such circumstances where these popular causal learning algorithms suffer.

In these "small data" scenarios, experts usually employ additional assumptions and modifications to make causal discovery more reliable: various modified versions of the PC-algorithm (Peter-Clark Algorithm) or the SBL framework have been proposed that consider different such techniques [49, 50, 51]. These modifications usually come with two recurring themes: 1. searching for high ranking causal features and 2. consolidating the results of multiple unreliable conditional independence (CI) tests. For example, in [49], a modified SBL framework has been proposed where authors, utilizing a bootstrap procedure, calculate a confidence interval for the identified causal feature, they then only output the highest-ranking causal features. In another study, Colombo and Maathuis propose a modification to the PC-algorithm in order to consolidate the inconsistent results of CI tests and make the algorithm order-independent [52]. In [50], a similar strategy is seen where the PC-algorithm is complemented with additional rules to make up for incongruent CI test results. We can identify a similar theme in the work of Kalisch and Buhlmann [45]. They show that PC-algorithm scales significantly better for sparse networks: a special property of these networks is that they can be identified using low order CI tests.

These results and the recurring theme of inaccurate higher-order CI tests suggests the following: the poor performance of most causal discovery tools, when the sample size is limited, is due to only having crude estimates of higher-order CI relations. Therefore, we hypothesize that if we were to come up with a causal inference algorithm that utilizes only first order (or second-order) statistical information — thus avoiding the estimation of higher-order CI relations — we may be able to achieve a more robust

and reliable solution. We can contrast this approach to causal discovery to the work of [49]. Similar to [49], we are looking to identify high confidence causal features. However, instead of searching for high confidence causal features using bootstrapping, we attempt to identify *all* high confidence causal features directly by restricting the input—apriori to any processing—to only consist of the pairwise dependency of the variables in the domain. For this reason, we think it is worthwhile to examine the types of causal knowledge that can be discovered in the pairwise dependency structure of a domain and to investigate whether the causal knowledge obtained in this manner is actually more accurate or not.

4.1.1 Overview of Prior Work

We were able to identify two works prior to ours that consider characterizing causal knowledge implicated by a pairwise dependency structure [53, 54]. Unfortunately, most of the theorems in [54] are presented without proof. In section II.C, we focus primarily on this work: we discuss a theorem that characterizes the class of pairwise dependency graphs, provide its missing proof, and present some new insights. In our proof of this theorem, similar to the works of [53] and [54], we make use of functional causal models, local causal Markov condition, and d-separation rules. These mathematical constructs provide the necessary conditions for linking causality to conditional dependence relations and make causal discovery possible from observational data (see [22] for an overview of these concepts). We will, however, slowly diverge from these assumptions in section II.D: as we further develop the theory of causation from marginal dependencies, expanding upon an observation made first by [53], we present a novel perspective on causal discovery that is based solely on the Reichenbach's principle of common cause and that of transitivity of causal relations (section II.D).

4.1.2 Overview of Main Contributions

From theoretical perspective, our contributions consist of providing new proofs for theorems in [53] and [54] , and also, bringing a new outlook to data-driven causal discovery based on the transitivity of causal relations. Our work also holds a significant practical value missing in [53] and [54]. While the theoretical work presented in [53] and [54] is very valuable, it is difficult to directly make use of their results in practice: as shown in [53], for even a small graph with only 6 nodes, there can be about 25,000 causal structures consistent with a given pairwise dependency structure. When the number of candidate causal structures is this high, reducing causal uncertainty (even by ten folds) does not hold much practical value.

In section II.E, we show how we can make use of these results for tackling a popular practical problem which can be appropriatley named [*small data*] "causal feature selection" task: given limited measurements of a set of candidate features in two (or more) classes, identify those that are *direct* causal effects of the class variable. This problem arises frequently in practice [55, 56]: as a case study, we consider gene expression measurements of two populations of cases (depressed) and controls (healthy) and attempt to identify genes whose expression are *directly* causally influenced by the condition (major depressive disorder as phenotype)—see section I.D for more detail and section III.B for experimental results with real data. We refer to this specific instance of the causal feature selection problem as the ***causal gene selection*** problem. We show that without conditional independence tests and using only statistical information of the first order and second order type (co-expression and differential expression analysis as referred to in omics literature) one is still able to obtain causal knowledge regarding the relationship between the class variable and the candidate features. Interestingly, causal knowledge obtained in this manner consists of statements such as "the chromatic number of the complement of the class conditional

marginal independence graph is a *lower bound* on the outdegree of the class variable" (Lemma 8), or "the nodes in the class conditional marginal independence graph whose boundaries are minimal (w.r.t partial ordering induced by subset relation) consist of only attributes directly connected to the class variable" (Lemma 9). They relate some (*global*) structural property of the pairwise dependency graph to an approximate description of the underlying generative causal model in terms of lower bounds and *minimal elements*. To the best of our knowledge, these are new results in the field of causal discovery.

4.1.3 Overview of Results

In section III.A, we present our simulation results. We compare the precision and the recall of our proposed algorithm against that of CI tests (the PC-algorithm route to causal discovery) in identifying children of a node among its descendants. Our simulation results confirm our central hypothesis: in small sample limits, causal knowledge obtained through pairwise dependency structure is significantly more accurate than causal knowledge obtained through conditional independence tests. Finally, in section III.B, we use the causal discovery tools we have developed for tackling a causal gene selection problem by analyzing the dataset GSE201521. While the nature of the problem makes it hard to validate our results, we analyze the biological significance of the selected set of genes: we find among them two genes, CCL2 and MTRNRL8, for which there exists notable biological evidence that implicates a causal role for them in developing MDD [57].

4.1.4 Problem Setting: Causal Gene Selection

In this section we consider the problem of *causal gene selection* in more mathematical detail. We represent the class (case/control) [also referred to as phenotype]

through the binary random variable C and the gene expressions through continuous random variables $X_i : i \in [1..m]$. We approximate the underlying causal generative structure through a functional causal model and assume that a certain Directed Acyclic Graph (DAG) represents the causal mechanisms at work in the healthy (control) population. In other words, we assume that every measured attribute is a deterministic function of its direct causes (graphically, these direct causes correspond to a node's parents in the DAG) and an unobserved noise term: $X_i = f_i(Pa_{X_i}, \zeta_i)$. We consider a phenotype as a change in some of these causal mechanisms. Then, an *affected* gene whose expression in the healthy population, $C = 0$, is the result of the causal mechanism $X_i = f(Pa_{X_i}, \zeta_i)$, in the *case* population appears instead as the result of a faulty causal mechanism of the form $X_i = \hat{f}_i(Pa_{X_i}, \zeta_i, \hat{\zeta}_i)$. The change in causal mechanisms going from control population to the case population can thus be shown by setting the phenotype variable as a root node in the aforementioned DAG and connecting it to X_i's whose generating causal mechanisms are affected by phenotype: $X_i = g_i(Pa_{X_i}, \hat{\zeta}_i, \zeta_i, C)$ (for now, we assume there are no hidden confounders and that the random variables $\hat{\zeta}_i$ and ζ_i are mutually independent: $P(\hat{\zeta}_1, \zeta_1, ..., \hat{\zeta_m}, \zeta_m) = \prod_{i=1}^{i=m} P(\hat{\zeta}_i)P(\zeta_i)$, implying the manner in which C affects a variable is independent of the way it affects others and the other sources of variation in the domain).

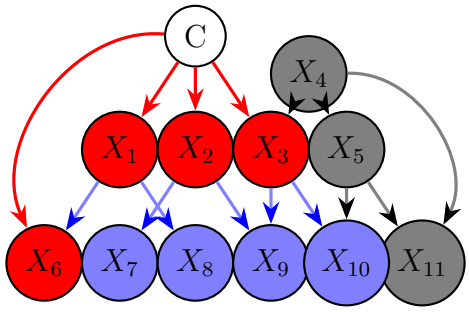


Figure 4.1: Our goal is to identify the direct children of the class variable, i.e. nodes marked by the red color.

To identify the direct children of phenotype from observational data (gene expression measurements in the two populations), we first make use of a tool commonly employed in gene expression studies and filter the set of candidate genes to only those which are differentially expressed. Differentially expressed genes (DEGs) are those genes whose average expression values are observed to be different in the populations under consideration $\mu_{X_i}^{C==0} \neq \mu_{X_i}^{C==1}$: this is similar to the first step of the PC-algorithm where the neighbors of a node are restricted to those that are correlated with the given node. From a graphical perspective, the DEGs are set of nodes which have an open path to the phenotype node, i.e., DEGs are a subset of the descendants of the phenotype node (this is assuming that one has adjusted for relevant covariates). They consist of ***all*** the descendants for probability distributions where $\mu_{X_i}^0 = \mu_{X_i}^1$ implies $P(X_i, C) = P(X_i)P(C)$. Gaussian distributions or binomial distributions are examples wherein the $DEGs \equiv Desc(C)$. Putting it more directly, we will use correlation tests (the t-test and the Fisher's Z-test) for measuring dependence in our experiments— the theory would remain unchanged if one uses other measures of dependence, such as spearman correlation, mutual information, etc.

Next, to provide a solution to the causal gene selection problem, we need to distinguish between those descendants that are children of the phenotype node and those that are indirectly connected to the phenotype node. To accomplish this task, one can employ conditional independence tests of the form $X_i \perp C \mid X \subset \{X_1, X_2, ..., X_m\}$. Causal inference methods such as PC-algorithm or score based learning algorithms, in essence, implement (score) these conditional independence tests. However, as discussed, such tests or similar algorithms are expensive in terms of the required sample size. Another set of tools that could possibly be employed to distinguish between the direct and indirect children of the phenotype consist of *feature selection algorithms*, such as those discussed in [58, 55]. However, as shown in [56], features selected by any typical

feature selection algorithm do not necessarily consist of *only* direct children of the phenotype. Furthermore, the size of this set is heavily influenced by the number of available samples. Finally, as any practical feature selection algorithm must employ a *local* greedy search in the space of candidate features, the selected set of features is usually not guaranteed to be the globally optimal one.

4.2 Methods

In this section, we review and discuss some results from the papers [53] and [54] that characterize the causal knowledge implicated by a pairwise dependency structure. We have collected these results in this section and have provided new proofs for them in the appendix for three reasons. First, some of the results are lacking proofs. Secondly, the current proofs use heavy mahinery from exisiting literature and its hard to parse them without familiarity with the used graph-theoretic concepts. And finally, studying these results will help us solve the causal feature selection problem in section II.E. The lemmas that appear below were first discussed in [54] (without proof) and more recently have been re-examined in [53].

4.2.1 Causal Assumptions

For now, we assume *faithfulness*, [1] and further, assume that there are no hidden confounders (i.e., all the common parents of all the attributes are included in the domain or take the same value across all units in the population). We will later allow

[1] We assume any independence observed arises due to the graph/poset structure rather than the parameters controlling the conditional distributions, i.e., two variables are statistically independent only if there is no open path between the two variables in the underlying DAG (and in the case of a poset, the two are independent only if they do not share a common ancestor).

the possibility of the existence of hidden confounders in section II.E and when solving the causal feature selection problem.

4.2.2 Notation

We encode the pairwise marginal independencies that hold amongst the variables in a domain in an undirected graph whose missing edges inform of pairwise independence between the two nodes. Following are the notations we use throughout this paper. We use the letter D when referring to a **DAG** (we reserve D_0 for the data generating DAG), the letter G when referring to **undirected graphs**, and finally, the letter P to denote partially ordered sets (posets) (P_0 is reserved for the poset of the data generating process).

Skeleton of a DAG is an undirected graph wherein every edge in the DAG is replaced by an undirected edge; we represent the skeleton of a DAG D through the notation $U(D)$. $Anc_D(X)$ is used to denote the set of **ancestors** of the node X in the DAG D; we assume the convention that X is included in $Anc_D(X)$. **Boundary** of a node, X, in an undirected graph, G, is denoted by $Bd(X)$ and consists of nodes which are adjacent to X; we assume the convention that X is included in $Bd(X)$. We use the notation $Bd_G(X)$ to refer to the boundary of X in the graph G. Operators $E(.)$ and $V(.)$ are used to denote the set of edges and vertices of a graph. In the case of a directed graph, we use $E(.)$ to refer to the edges in the skeleton of the said graph.

We say that a DAG, D, **implies** an undirected edge, $a - b$, if and only if, a and b share a common ancestor in D. We say a DAG, D, **generates** a marginal independence graph, G, if D implies all and only the edges of G.

We refer to a subset of vertices of an undirected graph that form a complete subgraph as a **clique**. A clique in an undirected graph is called **exterior** if it contains at least one node that is not contained in any other cliques of that graph. The letters

Γ and γ are used to denote a family of cliques and a particular clique. We say a family of cliques provides an **edge cover** for an undirected graph if every edge in the undirected graph is contained in at least one of the cliques. A **v-structure** in an undirected graph is a triplet $a - b - c$ with $a \neq c$. A **chordless 4-chain** is a chain in an undirected graph consisting of four nodes wherein only the immediate vertices in the sequence or the first and last vertices are adjacent, i.e. $c - a - b - d$ with $a \neq d$ and $c \neq b$. We use the notation $\Sigma(.)$ to refer to the marginal independence graph of a DAG: $(a, b) \in E(\Sigma(D))$ if and only if $Anc_D(a) \cap Anc_D(b) \neq \emptyset$.

4.2.3 From associative relations to causal implications: a review

The clique characterization of marginal independence graphs

The following lemma is concerned with characterization of the class of marginal independence graphs through clique structure. It was proposed in [54] without a proof (theorem 2:(ii)) and re-stated in [53] borrowing a theorem characterizing the *edge simplistic* properties of bound graphs [59].

Lemma 1 An undirected graph, G, is generated by a DAG, D_0, if and only if there exists a family of exterior cliques, Γ, of G that provides an edge cover for G. Furthermore, such a family is unique if G has no isolated vertices. (For proof see appendix A1.)

The sink completion characterization of marginal independence graphs

Lemma 3 is a strengthened version of a theorem presented in [54] without a proof (theorem 2:(i)). This lemma is also used in [53]; however, the authors do not provide its proof. This lemma is concerned with the characterization of the class of marginal independence graphs through a process referred to as *sink completion*. A sink completion of an undirected graph, G, is **any** DAG that results from the

following two-step procedure. In the first step, referred to as **sink orientation**, for every v-structure $a - b - c \in G$ (with $a \neq c$), we orient them as $a \rightarrow b \leftarrow c$. In the second step, we remove bi-directed edges and orient the remaining undirected edges in any arbitrary manner such that the ouput is a DAG. **We show the set of DAGs resulting from the sink completion of an undirected graph as** $S(G)$. In Lemma 2 we first characterize this process in terms of the boundaries of the nodes involved, and in Lemma 3, we use this procedure to characterize the class of marginal independence graphs.

Lemma 2 Given an undirected graph G, an edge $a - b$ is removed during its sink completion if and only if $Bd_G(a) \not\subseteq Bd_G(b)$ and $Bd_G(b) \not\subseteq Bd_G(a)$. In other words, $a - b$ is removed, if and only if this edges participates in a chordless 4-chain in G. Furthermore, $a - b$ is oriented as $a \rightarrow b$ during sink orientation if and only if $Bd_G(a) \subset Bd_G(b)$.

Lemma 3 An undirected graph, G, is generated by a DAG, D_0, if and only if any DAG resulting from the sink completion of G implies **all and only** the edges of G. (See proof in appendix A2)

It can be shown lemma 3 implies that a marginal independence graph restricts candidate causal structures to only DAGs which are subgraphs of those in the sink completion of the marginal independence graph [53].

4.2.4 A new perspective on causal implications of a marginal independence structure: from causal DAGs to causal posets

While in the former section we were primarily concerned with the work of [54], in this section, we mainly focus on one of the results of [53]. In [53], the authors note that Lemma 1, as proposed in [54], was previously discovered by McMorris and Zaslavsky in a different setting and for characterizing *upper-bound* graphs of

posets [60]: Textor, Idelberger, and Liśkiewicz use this observation to help develop an algorithm for enumerating causal DAGs consistent with a marginal independence graph. We will, however, utilize this observation to present a novel perspective on the identifiability of causal relations from purely observational data.

Let us present a simplified road map of the results of previous section and their proofs provided in the appendix. The starting point (similar to the work of [53, 54]) of lemmas 1 and 3 is the local causal Markov condition, and from there, using d-separation rules, one can arrive at lemnmas that characterize the class of marginal independence graphs. Then, faced with the question "what aspect or facet of causation restricts the class of marginal dependency graphs", our answer —similar to works of [53, 54]—- should be the local causal Markov condition. This answer, however, is not satisfactory to us as a marginal dependency structure does not contain any information on the conditional independence relations between the variables in a domain. The work of [60], on the other hand, shows that we can reach a similar conclusion in a different setting where the local causal Markov condition does not hold (note also the fact that in our proofs we hardly used the local causal Markov condition). Then, as far as these lemmas are concerned, it seems possible to relax the local causal Markov condition and replace it with a more fundamental and elementary aspect of causal relations.

To develop a suitable theory of causal discovery from the marginal independence graph, we require two things. First, we define causation ($\geq$) to be a binary relation between events that is reflexive (an event causes itself), anti-symmetric (no two events can cause/precede another), and transitive (if $a \geq b$ and $b \geq c$, then $a \geq c$) [we only require $\geq$ to be anti-symmetric and transitive, reflexivity is assumed for the sake of convinience]. Any set of variables can be thought of as a partially ordered set (poset) where the partial order is causation—from a DAG viewpoint of causality, any

causal DAG with the same transitive closure represents the same *causal poset*, and a causal poset represents the class of DAGs with the same transitive closure. Next, to connect causation to observable statistical relations we use Reichenbach's common cause principle: if two events, a and b, are dependent, then either one causes the other ($a \geq b$ or $b \geq a$) or they both have a common cause ($\exists c$ s.t. $c \geq b$ and $c \geq a$). It is the transitivity of causation together with Reichenbach's common cause principle that restricts the class of marginal independence graphs. The following lemmas, which are analogous to the lemmas proposed in the previous section, result directly from these two assumptions: they show that we can let go of the local causal Markov condition while preserving the structure of the class of marginal independence graphs. Here we show the underlying causal poset with $P = (V, \geq)$, and the undirected graph representing the pairwise dependence of the attributes with $G = (V, E)$. We say that a causal poset, P, generates a marginal independence graph, G, if and only if for any edge $a - b$ present in G, there exists a $z \in V$ such that $z \geq b$ and $z \geq a$ in P (in other words, we again assume that there are no hidden confounders, and further, assume faithfulness). The proofs of the following two lemmas are provided in the appendix.

Lemma 4 An undirected graph $G = (V, E)$ is generated by a causal poset $P = (V, \geq)$ if and only if there exists a family of exterior cliques, Γ, of G that provide an edge cover for G. Further, such a family is the only such if G has no isolated vertices. (For proof see appendix A5)

Lemma 5 G is generated by a causal poset P, if and only if, any *poset* in the sink completion of $P \in S(G)$ implies **all and only** the edges of G (we say that a poset P implies an edge $a - b$ in G if and only if there exists a z such that $z \geq b$ and $z \geq a$ in P). (For proof see appendix A6)

In summary, a well known aspect of causality, the transitivity, enables causal discovery from pairwise dependency structure of a domain independent of the local

causal Markov condition. This allows us to conduct causal inference in regimes where local causal Markov condition is not an appropriate assumption. The two prominent examples of such cases involve measurement errors and averaging.

Measurment Errors

If a variable, X_1, influences another, X_2, indirectly through a third variable, Z, then the local causal Markov condition can be used to identify the indirect nature of this influence as X_1 and X_2 are independent conditioned on Z. If, however, our measurments of the variables X_1, X_2, and Z, contain errors, a similar conditional independence will not hold for the measured variables. In general, conditional independencies of a set of variables $\{X_1, X_2, ..., X_n\}$ will not hold among the variables $\{X_1^m, X_2^m, ..., X_n^m\}$, where X_i^m's are noisy measurements of X_i's: $X_i^m = f_i(X_i, \delta_i)$, f is a deterministic function and δ_i's are jointly independent [21]. While the local causal Markov condition does not translate between these two domains, the Reichenbach's common cause principle and the transitivity condition do. Two measured variable X_1^m and X_2^m are dependent, if and only their *real* counterparts X_1 and X_2 have a common cause. Furthermore, if we find X_1^m and some X_i^m to be dependent, and, at the same time, see that X_2^m and X_i^m are independent, then we can conclude that X_1 is not a cause of X_2 (see Lemma 7): if X_1 is a cause of X_2, by transitivity of causation, we require any pattern in X_1 (a pattern being any X_j dependent on X_1) to also be observable in X_2 (for X_j and X_2 to be also dependent). As we have assumed faithfulness, we conclude any pattern in X_1^m should also be observable in X_2^m. This contradicts our assumption that there is a pattern X_i^m which is in X_1^m but not X_2^m. The translation of these two conditions between these sets can also be immediately noted by considering that the marginal independence graph of the variables $\{X_1^m, X_2^m, ..., X_n^m\}$ is identical to that of the variables $\{X_1, X_2, ..., X_n\}$; i.e X_1^m and X_2^m are dependent if and only

if X_1 and X_2 are dependent. This lets us conclude that a causal inference tool that operates only via the marginal independence graph of the measured variables will reach correct causal conclusions while one utilizing the local causal Markov condition does not.

Averaging

Gene expression experiments usually measure the average concentration of a gene in a large collection of cells rather than in a single cell. Consider a population of cells $\{C_1, ..., C_M\}$ and let X_i^m represent the expression level of gene X_i in the cell C_m. Assume that the same causal mechanisms are at work in every cell so that if X_i^m and X_j^m are independent conditional X_k^m for a given cell, then a similar independency holds among the expression levels of these genes in other cells. The important point here is that the corresponding conditional independency will not hold among the average expression levels [21]. On the other hand, the marginal independence graph of the average expression levels is identical to the marginal independence graph of the gene expression levels in each cell. Therefore, causal conclusions made based on the evidence of the average expression levels are incorrect if one uses the local causal Markov condition while those that are inferred solely via the marginal independence graph are correct.

4.2.5 Causal Feature Selection Using Marginal Independence Graph

Either through transitivity of causation or with the help of Lemmas 1 and 3, we can directly reach the following lemmas: these two lemmas are at the core of our causal feature selection algorithm.

Lemma 6 Consider the set of *maximal cliques* of the marginal independence graph: $\Gamma = \{\gamma_1, \gamma_2, ..., \gamma_m\}$. For any $x \in \gamma_i$, all its descendants are also in γ_i: $\{y|x \geq y\} \subseteq \gamma_i$.

The proof of the above is provided in appendix A6. From Lemma 6, we can immediately conclude the following.

Lemma 7 Consider the set of *maximal cliques* of the marginal independence graph: $\Gamma = \{\gamma_1, \gamma_2, ..., \gamma_m\}$. For an attribute X_i, show the set of cliques where X_i participates in as $\mu_i = \{j|X_i \in \gamma_j\}$. We refer to μ_i as X_i's (clique) membership. For any two attributes X_i and X_j, $X_i \geq X_j$ only if $\mu_i \subseteq \mu_j$ (assuming local causal markov condition, this is equivalent to saying that there exists a directed path from X_i to X_j only if $\mu_i \subseteq \mu_j$).

We can use Lemmas 6 and 7 to tackle the problem of causal feature selection. Let us recall the general formulation of this problem. We are given measurements of a set of candidate features in two (or more) classes and we are asked to identify those that are *direct* causal effects of the class variable. We assume the class variable is the root node (a node without any parents amongst the variables in the domain) in some underlying DAG.

Algorithm

To solve the causal feature selection problem, we first propose to identify the attributes whose probability distributions depend on the population: $\{X|P(X|C = 0) \neq P(X|C = 1)\}$. As discussed previously, these attributes consist of descendants of the class variable. From now on, we assume that the candidate features only consist of such attributes. In the next step, we calculate the marginal independence graph of the candidate attributes in every class. We then form the class conditional marginal independence graph G: for any pair of attributes, X_i and X_j, $X_i - X_j \in G$ if and only

if X_i and X_j are found to be dependent in at least one of the classes. Using lemmas 6 and 7 we have:

Lemma 8 The minimum clique cover number of G, the class conditional marginal independence graph, is a lower bound on the the outdegree of C, the class variable. (See appendix A7 for proof)

Lemma 9 Consider the set of *maximal cliques* of the class conditional marginal independence graph: $\Gamma = \{\gamma_1, \gamma_2, ..., \gamma_m\}$. For an attribute X_i, show the set of maximal cliques where X_i participates in as $\mu_i = \{j | X_i \in \gamma_j\}$. Then the attributes in $R = \{X_i | \forall j \neq i : \mu_j \not\subseteq \mu_i\}$ consist of a subset of the direct children of C, the class variable.(See appendix A8 for proof)

4.3 Evaluation

4.3.1 Simulations: Precision and Recall of Lemma 9 vs. CI tests

In this section, we wish to investigate if the causal knowledge acquired from the marginal independence graph is more accurate than the causal knowledge obtained through CI tests. However, these two methods produce causal objects at different *resolutions*, and it would be unreasonable to compare such causal objects directly. CI tests provide a detailed description of causal mechanisms in terms of an equivalence class of DAGs. A marginal independence graph is unable to offer such resolution and cannot distinguish between DAGs giving rise to the same marginal independence graphs– the equivalence class arising from pairwise dependency information is much larger than that arising from the CI tests. Still, it is possible to compare these two methods against each other in specific problem instances. Consider the causal discovery problem discussed throughout this paper, the causal feature selection problem. In the causal feature selection problem, we are given a root node (a node without any

parents amongst the variables in the domain), and are asked to find its children among its descendants. Interestingly, CI tests and marginal independence graph can both be used to tackle this problem, and they present two distinct approaches for distinguishing between direct and indirect effects of such a node. On the one hand, to identify children of this node through CI tests, we can apply the skeleton learning step of the PC-algorithm to the node under consideration. On the other hand, based on Lemma 9, we can identify a subset of the children of the node by considering the boundaries of the candidate nodes in the marginal independence graph. In the simulations presented in this section, we use this problem setting to compare the precision and recall of the CI tests against our proposed method in identifying direct children of a node among its descendants.

We set up our simulation environment through the following steps. We first create the adjacency matrix of our DAG. We control for the number of nodes in the DAG, $m \in \{7, 8, 9, 10, 11, 12, 13, 14, 15\}$, and also the *sparsity*, $s \in \{0.1, 0.2, 0.3, 0.4, 0.5\}$, of the DAG: sparsity of a DAG is the parameter that controls expected number of neighbors (parents or children) of a node and equals the ratio of the expected number of connections of a node to the total number of nodes in the graph. We generate this adjacency matrix similar to the process described in [45]: such a simulation setting is widely used in practice to probe the performance of causal learning algorithms. We then add a root node (the *class variable*) to the DAG just created and connect the root node to $k \in \{1, 2, 3, 4, 5\}$ randomly chosen nodes in the DAG. These are the direct children of the root node. Afterward, we generate $n \in \{30, 40, 50, 70, 90, 110, 130, 150\}$

samples from the DAG, recursively, starting from the root node, and down to the leaves using the following equation:

$$x_i = \mu_i + \sum_{x_i \in \pi_i} b_{ij}(x_j - \mu_j) + \mathcal{N}(0, 1/\tau_i), \tag{4.1}$$

where network parameters μ_i, τ_i were sampled from a uniform distribution $\mathcal{U}(0, 2)$, and b_{ij} was chosen as independent realizations of $\mathcal{U}(-2, 2)$. Afterward, using CI tests and our proposed method (Lemma 9), we try to distinguish between the children and the rest of the descendants of the root node (we identify the descendants of the root node directly through DAG adjacency matrix). We used Fisher's Z-test with a p-value of 0.01 to evaluate statistical dependencies (i.e., to convert partial correlation or correlation values to boolean statements of dependence and independence).

In figures 4.2-4.5, we present the precision and recall of the CI tests against our proposed method (referred to as the *boundary method*) in distinguishing between direct and indirect effects of a root node as a function of 1. sample size, 2. network size, 3. number of children, and 4. network sparsity; the results presented are the average of 2000 iterations. In these figures, we have also included a new algorithm, referred to as *combined method*. This algorithm is simply the AND of the CI method and the boundary method: in the combined method, a feature is selected as a child of the root node, if and only if both the CI tests and our proposed method select this feature.

4.3.2 Gene Expression Analysis: Dataset GSE101521

Dataset GSE101521 contains 59 recordings of whole-exome gene expression of three groups of controls (CON, N=29), major depressive disorder suicides (MDD-S, N=21), and MDD non-suicides (MDD, N=9) [57]. The data was collected using

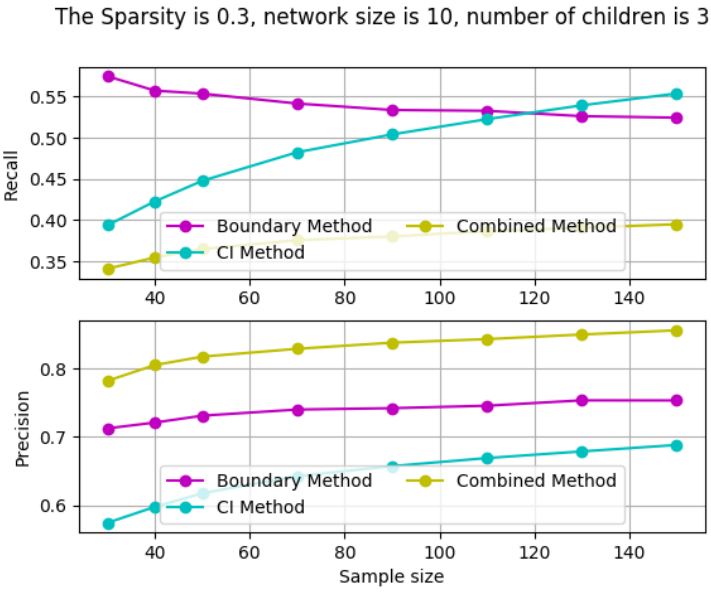


Figure 4.2: Precision and Recall of boundary method against CI tests in identifying children of a node among its descendants as a function of the sample size

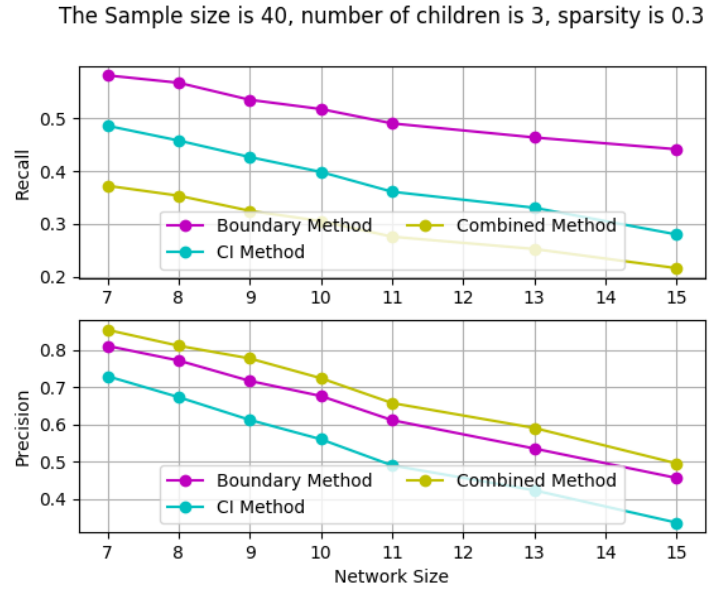


Figure 4.3: Precision and Recall of boundary method against CI tests in identifying children of a node among its descendants as a function of the network size

RNA-seq measurements from the dorsal lateral prefrontal cortex of sudden-death medication-free individuals postmortem. After adjusting for covariates age and gender,

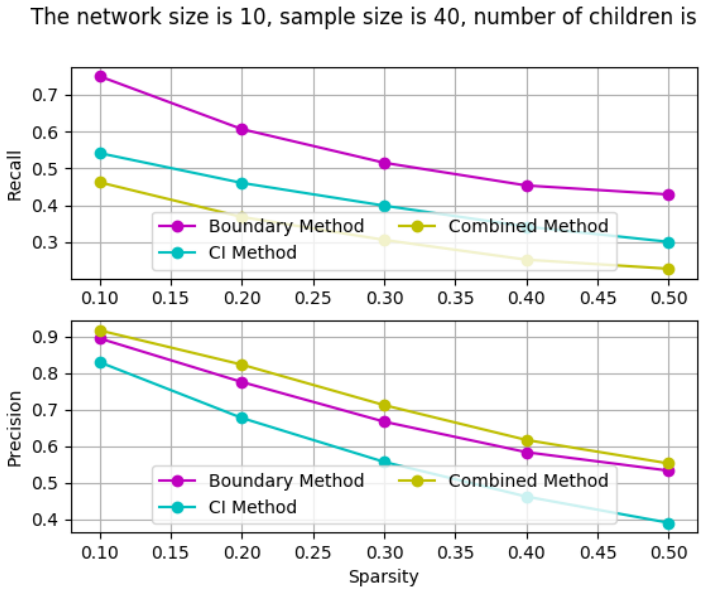


Figure 4.4: Precision and Recall of boundary method against CI tests in identifying children of a node among its descendants as a function of the network sparsity

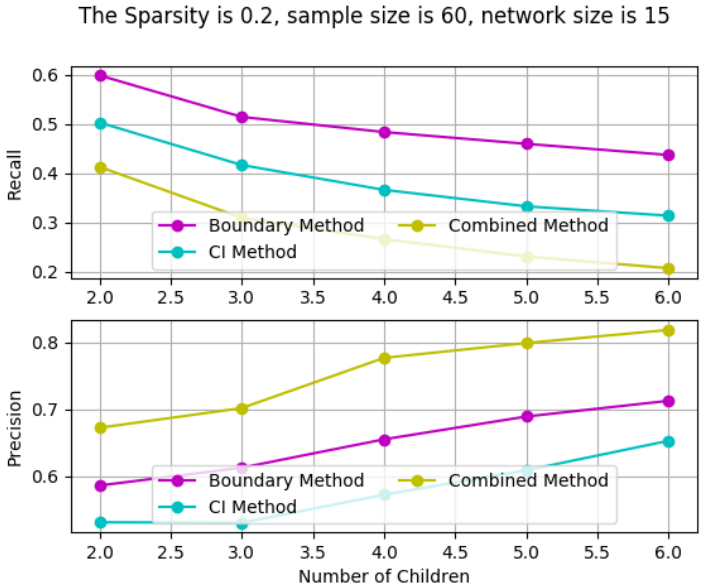


Figure 4.5: Precision and Recall of boundary method against CI tests in identifying children of a node among its descendants as a function of the number of children of root node.

the authors used DeSeq2 v.1.6.3 to assess differential expression among the groups [61]. Between the two groups of CON and MDD-S, the authors identified thirty-four differentially expressed genes at a false discovery rate of $p < 0.1$ (a list of the thirty-four differentially expressed genes is provided in the supplementary material of [57]).

Utilizing lemma 9, we examined which of the differentially expressed genes were directly affected by the phenotype (MDD-S): we used Fisher's Z-test with a p-value of 0.01 to form the class conditional adjacency matrix. Our algorithm selected four genes as the direct children of the phenotype; these genes, together with their functional significance, are shown in Table 4.1.

Gene ID	Functional Significance
CCL2	Protein kinase activity
GBP2	Interferon gamma signaling
MTRNR2L8	Neuroprotective and antiapoptotic factor
FER1L6	ATPase activity

Table 4.1: Among the 34 differentially expressed genes between the two groups of MDD-S and CON, we identied four that must be directly affected by the phenotype.

4.4 Discussion and Conclusions

The field of causal learning is rapidly advancing, and there is now a multitude of approaches for identifying causal relations from purely observational data. However, the most popular causal learning algorithms require sample sizes that are sometimes not met in practice. Causal discovery in these small sample limits was the central theme of our paper. We hypothesized that avoiding higher-order CI tests would benefit causal discovery in such scenarios. This lead us to study the types of causal knowledge that can be discovered in the pairwise dependency structure of a domain. Expanding upon the works of [54] and [53], we presented novel results characterizing

causal knowledge implicated by a marginal independence graph. We further showed that such causal insights embedded in the marginal independence graph stem from a basic and elementary aspect of causal relations, the transitivity of causation.

We used the pairwise dependency structure of a domain to tackle a special causal discovery problem, that of distinguishing between direct effects and indirect effects of a variable. In our simulations, we compared the precision and the recall of our proposed algorithm against that of CI tests in solving this causal discovery problem. Our simulation results confirmed our main hypothesis. In small sample limits, causal knowledge obtained through the marginal independence graph is significantly more accurate than the causal knowledge obtained through CI tests. In a specific problem instance where our sample size was equal to thirty, the number of candidate variables was ten, and three among them were the children, and the sparsity of the network was set to 0.3, we saw a 30% reduction in error rate, increasing the precision from 0.57 to 0.70, while at the same time improving the recall. We also suggested that it is possible to employ our algorithm in supplement to CI tests: this combination resulted in a further 23% reduction in the error rate at the cost of the recall. We further used this causal discovery tool to examine the dataset GSE201521 and identify genes that are directly affected by MDD. While the nature of the problem makes it hard to validate our results, there is still significant biological evidence that implicates a causal role for two of the four selected genes (CCL2 and MTRNRL8) [57].

Supplementary Materials

A1. An undirected graph G is generated by a DAG D_0 if and only if there exists a family of exterior cliques, Γ, of G that provides an edge cover for G. Further, such a family is the only such if G has no isolated vertices.

Proof

Only if: Assume any DAG D. The roots of D, $\{\psi_1, \psi_2, .., \psi_m\}$ together with all their descendants form a family $\Gamma = \{\gamma_1, \gamma_2, ..., \gamma_m\}$ of exterior cliques in G which cover every edge in G. Every $\gamma_i \in \Gamma$ is an exterior clique since adjacency of any root node, ψ_i, is included in γ_i: $Bd_G(\psi_i) = \gamma_i$. To see that Γ is an edge cover, we note that any adjacency (x_i, x_j) in G implies x_i and x_j share a common ancestor, z, in D. The edge (x_i, x_j) is covered by the exterior clique γ_k where $z \in$ descendants(ψ_k).

If: Assume there exists a family of exterior cliques in G that provides an edge cover for G. The following construct then immediatley gives a DAG, D, that can generate G. Assume $\Gamma = \{\gamma_1, \gamma_2, ..., \gamma_m\}$ is a set of exterior cliques in G which cover every edge in G. Find the exterior node, ψ_i, for all the exterior cliques γ_i; i.e. ψ_i is only contained in γ_i. Form D by connecting ψ_i to every other node in γ_i, $\psi_i \rightarrow x_i : \forall x_i \in \gamma_i$.

Γ **is unique:** Consider there are two families of exterior cliques Γ_1 and Γ_2. Consider any exterior clique $\gamma_i \in \Gamma_1$ and find the exterior node in this clique ψ_i. Since $Bd(\psi_i) = \gamma_i$, then any clique containing ψ_i must be equal to γ_i and thus γ_i must also be in Γ_2.

A2. An undirected graph, G, is generated by a DAG, D_0, if and only if any DAG resulting from the sink completion of G implies **all and only** the edges of G.

$$\underset{\substack{\text{Generating}\\ \text{DAG}}}{D_0} \xrightarrow{\Sigma} G \xrightarrow{\text{SC}} D \xrightarrow{\Sigma} \Sigma(D)$$

Figure 4.6: The sketch of the graphs and their transformations used in the proof of Lemma 3 in A2. SC refers to the transformation corresponding to sink completion and Σ refers to that of forming the marginal independence graph.

Proof

Existence: There is at least one DAG in the sink completion of any undirected graph. This is due to the fact that every sink orientation is acyclic and any partially oriented acyclic graph can be fully oriented to form a DAG (see appendix A3.)

All: Let D_0 denote the data generating DAG. We first show that every edge in its marginal independence graph, $G = \Sigma(D_0)$, is also present in $\Sigma(.)$ of any DAG in G's sink completion. Without any loss, we show this for an arbitrary $D \in S(G)$. In mathematical terms, we wish to show $(a, b) \in E(G) \Rightarrow (a, b) \in E(\Sigma(D))$.

If (a, b) is present in G then (a, b) must have a common ancestor in D_0. We refer to this common ancestor as z. Since z is an ancestor of a (b) we have $Bd_G(z) \subseteq Bd_G(a)$ $(Bd_G(z) \subseteq Bd_G(b))$.

Now, if (a, b) is present in D, then, (a, b) will be present in $\Sigma(D)$ and we are satisfied. If (a, b) is missing in D, then, they must have participated in a chordless 4-chain $(c - a - b - d)$ in G (Lemma 2). This, in turn, implies that there is no directed path between a and b in D_0: $z \neq a, z \neq b$ (a directed path from a to b would imply that $Bd_G(a) \subseteq Bd_G(b)$). Furthermore, z cannot be adjacent to c or d as that would imply the chords $c - b$ or $a - d$ in G since $Bd_G(z) \subseteq Bd_G(a)$ and $Bd_G(z) \subseteq Bd_G(b)$. Therfore, we will have two v-structures in G, one in the form of $c - a - z$ with $c \neq z$ and the other being $z - b - d$ with $d \neq z$.

The sink orientation process will have these two v-structures oreinted as $c \rightarrow a \leftarrow z$ and $z \rightarrow b \leftarrow d$. The last piece of the puzzle is to note that the edge $z - a$ $(z - b)$ cannot be removed during sink completion as we have assumed z is an ancestor of a (b), and therefore, $Bd_G(z) \subseteq Bd_G(a)$ $(Bd_G(z) \subseteq Bd_G(b))$. The formation of the structure $a \leftarrow z \rightarrow b$ in D implies that the edge (a, b) will be present in $\Sigma(D)$.

Only: We wish to show that every edge in $\Sigma(D)$ is also present in G. In other words, we wish to show $(a, b) \in \Sigma(D) \Rightarrow (a, b) \in E(G)$. Since $E(D) \subset E(G)$, any

edge in $\Sigma(D)$ not present in G must also not be present in D. The difference between $\Sigma(D)$ and D can be due to *transitive edges* or *trek edges*.

Only: transitive edges We show that any DAG, D, formed during sink completion is a poset. To prove this, we show that no structure of the form $a \to b \to c$ with $a \not\to c$ is present in the sink completion of any undirected graph G. Assume the contrary that such a structure exists in D.

First we note that the edges (a, b) and (b, c) must have been in G as $E(D) \subseteq E(G)$. Furthermore, the edge (a, c) must also have been in G as without (a, c), $a - b - c$ would have formed a v-structure in G that would have been oritented as $a \to b \leftarrow c$ during sink orientation. Since we have assumed that the edge (a, c) is not present in D, then this edge must have been removed during the sink completion process. This lets us conclude a and c must have participated in a chordless 4-chain in G: $e - a - c - f$. Thus, the structure shown in figure 4.7 is present in G. Regardless of the existence of edges $e - b$ or $b - f$ in this structure, no sink completion of it leads to a chain of the form $a \to b \to c$. First, if the edge $e - b$ (or $b - f$) is missing in G, then $a - b$ (or $a - c$) will be removed during sink completion as $e - a - b - c$ (or $a - b - c - f$) form a chordless-4 chain. If both edges $e - b$ and $b - f$ are present in G, then we have v-structures $a - b - f$ (with $a \neq f$) and $c - b - e$ (with $c \neq e$) in G causing $a - b$ and $b - c$ to be oriented as $a \leftarrow b$ and $b \leftarrow c$ during sink orientation. This contradicts our assumption that the edge $b - c$ is oriented as $b \to c$ in D.

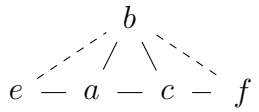


Figure 4.7: The marginal independence structure of a non-transitive triple chain in $D \in S(G)$. The presence of edges marked by dashed lines is not certain (they can be both present or absent).

Only: trek edges We define a trek edge, (a, b), as an edge that is added to D in process of forming its marginal independence graph, $\Sigma(D)$, and connects nodes that have a common ancestor, c, where $c \neq a, c \neq b$. We previously proved that D is a poset. Therefore, any common ancestor of two nodes is directly connected to both of the nodes in D. This common ancestor must also be adjacent to both a and b in G as $E(D) \subseteq E(G)$. We now must show that $a \leftarrow c \rightarrow b \in D$ only if $a - b \in G$. Assume $a - b \notin G$. Then, since in constructing D, sink orientation would orient any structure of the form $a - c - b \in G$ as a collider ($a \rightarrow c \leftarrow b \in D$) we face a contradiction as we had assumed that c is a common parent of a and b in D.

A3. Sink orientation of any undirected graph, G, is acyclic.

Proof

Assume that there is a cycle formed during sink orientation. Let this cycle consis of the nodes $\{c_1, ..., c_n\}$. Assume that the orientations are of the form $c_i \rightarrow c_{i+1}$ for the nodes with indices in $i \in 1, ..., n-1$ and the cycle is completed with the orientation $c_n \rightarrow c_1$ for the nodes c_n and c_1. The orientations $c_i \rightarrow c_{i+1}$ imply the boundary relations $Bd_G(c_i) \subset Bd_G(c_{i+1})$ (see Lemma 2). Thus, $Bd_G(c_1) \subset Bd_G(c_n)$. However, this contradicts the orientation $c_n \rightarrow c_1$ which implies $Bd_G(c_n) \subset Bd_G(c_1)$.

A4. An undirected graph $G = (V, E)$ is generated by a causal poset $P = (V, \geq)$ if and only if there exists a family of exterior cliques, Γ, of G that provide an edge cover for G. Further, such a family is the only such if G has no isolated vertices.

This can be seen by considering Lemma 1 and the fact that any DAG D and the poset corresponding to its transitive closure give rise to the same marginal independence graph. However, as noted in [53], Lemma 4 is essentially the theorem proposed originally by McMorris and Zaslavsky [60] and this result predates the statement of [54].

A5. G is generated by a causal poset P, if and only if, any *poset* in the sink completion of $P \in S(G)$ implies **all and only** the edges of G (we say that a poset P implies an edge $a - b$ in G if and only if there exists a z such that $z \geq b$ and $z \geq a$ in P).

To see this, we note that any DAG in the sink completion, as shown in the proof of Lemma 3 in A2, is a poset (i.e. its transitive closure is itself).

A6. Consider the set of *maximal cliques* of the marginal independence graph: $\Gamma = \{\gamma_1, \gamma_2, ..., \gamma_m\}$. For any $x \in \gamma_i$, all its descendants are also in γ_i: $\{y|x \geq y\} \subseteq \gamma_i$.

Proof Consider any x_b in the boundary of x (a node is said to be in the boundary of another node if these two nodes are connected through an edge; by convention, we assume that a node is in its own boundary). From Reichenbach's principle of common cause we conclude there exists a z such that $z \geq x_b$ and $z \geq x$. Transitivity of causation implies that $z \geq x_d$ for any $x_d \in \{y : x \geq y\}$, concluding that $\{y|x \geq y\} \subseteq \gamma_i$.

A7. The minimum clique cover number of G, the class conditional marginal independence graph, is a lower bound on the the outdegree of C, the class variable.

Proof Consider a missing edge in G: $X_i - X_j \notin G$. From this missing edge, we can conclude that X_i and X_j don't share any common ancestors ($X_i \leftarrow Z \rightarrow X_j$) nor there is a directed path connecting these two to each other ($X_i ... \rightarrow ... X_j$). Since both these attributes are connected to the class node ($X_i \leftarrow ... \leftarrow C \rightarrow ... \rightarrow X_j$), then these attributes must be connected to the class variable via different *causal pathways*: in the underlying DAG (poset), the directed paths from C, the class node, to these two attributes cannot have any overlap as that would imply these two attributes share a common ancestor. Therefore, the paths from C to these two attributes must involve non-overlapping edges directed out from C. This allows us to conclude that the minimum number of edges required to connect the class variable C to all the candidate attributes cannot be smaller than the minimum clique cover number of G.

A8. Consider the set of *maximal cliques* of the class conditional marginal independence graph: $\Gamma = \{\gamma_1, \gamma_2, ..., \gamma_m\}$. For an attribute X_i, show the set of maximal cliques where X_i participates in as $\mu_i = \{j | X_i \in \gamma_j\}$. Then the attributes in $R = \{X_i | \forall j \neq i : \mu_j \not\subseteq \mu_i\}$ consist of a subset of the direct children of C, the class variable.

Proof According to Lemma 7, for any $X_r \in R$, the causal effect of class variable to these attributes cannot be mediated by any other observed variable in the domain. Therefore, these attributes must be directly connected to the class variable.

CHAPTER 5

CAUSAL DISCOVERY FROM HETEROGENEOUS DATA: DECIDING IF CHANGES IN MARGINAL DEPENDENCY STRUCTURE ADMIT A CAUSAL EXPLANATION

5.1 Introduction

Differential Connectivity Analysis (DCA) is a blanket term referring to a large and varied collection of algorithms [62, 63, 64, 65, 66]. At their core, these algorithms are concerned with the *causal structure* that orchestrates and connects the actions of processes in any functional system. This orchestrated action or harmony is manifest in the form of the various statistical dependence relations that hold in a domain. A DCA algorithm is used to inspect how this orchestrated action changes across different conditions and the unstable statistical dependence relations and those that are persistent are distinguished from one another. Often, experts employ DCA as a springboard to causal postulates. For instance, experts have used DCA to attribute complex diseases to, and as being caused by, the loss of certain desired statistical dependencies [67, 68, 66]. However, bridging the gap from persistent and unstable statistical dependence relations to causal conclusions has so far exclusively relied on heuristics and expert knowledge[1] [71, 68, 72, 73]. In this paper, we ask whether theoretical results, and not heuristics or expert knowledge, can substantiate the use of DCA for causal discovery.

[1]The following excerpt perhaps best indicates the general sentiment and heuristic that is often invoked in bridging this gap: "As with shadows, these correlational patterns are incomplete - and potentially ambiguous - projections of the original causal processes. As with shadows, we can infer much about the underlying causal process if we can learn to study their details [...]" [69, 70].

Such theoretical results can greatly help propel and expand the use of DCA as a causal discovery tool. First, without a theory that supports the use of DCA as a springboard to causal postulates, it has not ben easy to establish how well a particular DCA algorithm can generalize—as a causal discovery tool—beyond the specific data-set it was designed to handle. Thus, DCA algorithms have remained very much domain and data-set specific. For the same reason, even when it comes to the same domain or the same data-set, it is still challenging to establish which DCA paradigm is best suited for causal discovery [66]. In practice, DCA algorithms are often deployed as heuristic feature selection algorithms where the processes whose dependence relations are deemed to be highly unstable are picked out for further analysis and empirical evaluation. In some instances, expert knowledge and experimental data have validated such attempts at causal discovery through DCA, however, there are other instances where DCA has failed [74]. It still remains difficult to explain why DCA is a valuable causal discovery tool only in certain occasion and not the others. While these results paint DCA in a negative light, nevertheless, the undeniable fact is that statistical dependence relations have been observed to be unstable time and again, and these dependence relations are a reflection of the underlying causal forces; the only question is whether we can unravel the changes in dependency relations in terms of causal relations in polynomial time.

Among the massive collection of the different DCA paradigms, in this paper, we study what is the most basic and prevalent one [75, 63, 67, 66]. We consider only *marginal* dependence relations and are only concerned with the undirected graph of marginal dependencies wherein edges connect nodes representing statistically dependent processes or variables. We assume we have constructed a domain's marginal dependency graph under two conditions, i.e. case and control, and have distinguished the edges that are persistent from those that are unstable. Our main concern in

this paper is to evaluate DCA as a causal discovery tool, to understand what causal facts are concealed under the complex web of dependency changes, and if there are polynomial time algorithms that can reveal these causal facts. For this purpose, it is imperative that we first examine the following decision problem. Can we decide in polynomial time if the changes in the marginal dependency graph admit a causal explanation? We show that under certain set of assumptions, the simplest case of which is having only dependency loss, and in a more complex setting we have a relaxed form of faithfulness, a valid causal explanation, if one exists, can be constructed in polynomial time using Reichenbach's common cause principle and the formalism of functional causal models. While these decision type results are the main output of this paper, we also propose a novel causal discovery algorithm that given the unstable and stable marginal dependence relations, estimates a *informed* upper bound on the set of causal mechanisms that are unstable.

Before moving forward, we briefly pause here and compare our work to causal learning in settings where distribution shifts occur. In domain adaptation or distribution shift problems, similar to our setting, the causal mechanisms, specifically, the parameters associated with the causal structure generating the observed data, are assumed to change across different conditions [76, 77, 78, 79]. The principle causal discovery tool employed in these scenarios is that of the invariance of causal mechanisms [78, 79, 80]: $P(cause)$ and $P(effect|cause)$ should change independently across conditions while a similar relation does not hold in the reverse direction and the changes in $P(cause|effect)$ and $P(effect)$ are dependent. This asymmetry is often used for distinguishing cause and effect in such scenarios. When it comes to persistent and unstable marginal dependence relations, we cannot employ this principle since we have no information on the conditional distribution of the variables involved, and we cannot test for their invariance. Furthermore, these work often assume that the observed

distributions are *faithful* to some directed *acyclic* graph where the condition or the context variable is included as an exogenous node along the nodes corresponding to other processes in the domain. First, the faithfulness assumption requires, to a great extent, the stability of the generating causal structure, and thus the stability of almost all statistical dependence relations across the different conditions. In other words, these works assume that only the parameter *values* change across the conditions while the generating causal structure is not affected (i.e. a positive correlation value in one condition becomes negative in the other or some other similar parameter change setting). Second, these works require the generating causal structure to be acyclic, while we consider cyclic causal structures as well.

5.2 Preliminaries

In this work, we are concerned with a domain over m variables $X = \{x_1, ..., x_m\}$. We are given the marginal dependency relations in the domain under two conditions, $C \in \{1, 2\}$, in the form of undirected graphs (UG), G_1 and G_2, where an edge $x_i - x_j \in G_1(G_2)$ connects the two nodes x_i and x_j in $G_1(G_2)$ if and only if the nodes are statistically dependent when the conditioning variable or the context takes the value $C = 1(C = 2)$. We assume that underlying each domain is a causal system giving rise to the observed marginal dependency relations. Using the formalism of functional causal models, we represent this causal structure with a directed graph (DG) $D_1(D_2)$: $x_i \to x_j \in D_1(D_2)$ if and only if x_i is a direct cause of x_j (x_i to directly control the regulation and the production of x_j) when the conditioning variable or the context takes the value $C = 1(C = 2)$ [81].

Through Reichenbach's common cause principle, from the causal structure, we can directly obtain the marginal dependency graph by connecting nodes x_i, x_j, $i \neq j$, that share a common ancestor. We say a causal structure, D, **generates** a

marginal dependency graph, G, if G can be obtained from the causal structure in this manner and write $\Sigma(D) = G$. While the marginal dependency graph can be directly calculated from the causal structure, the causal structure is not uniquely identifiable from the marginal dependency graph. We study the causal implications of a marginal dependency graph in the next section; before that, we first need to review some graph-theoretic notions.

We use the abbreviations **UG**, **DG**, **DAG**, **PDG**, and **PDAG** in place of undirected graphs, directed graphs, directed acyclic graphs, partially directed graphs, and partially directed acyclic graphs. We say two nodes, a and b, are **adjacent** in a UG, G, if they are connected by an edge in G, and write $a - b \in G$. We refer to the collection of vertices adjacent to a node in a UG, together with the node itself, as the **boundary** of that node in the graph. We use the notation $\Omega_G(\xi)$ to denote the boundary of the node ξ in the graph G. We say the nodes a, b, and c form a **v-structure**, $a - b - c$, in a UG, G, if the nodes a and b, and the nodes b and c are both connected in the UG, $a - b \wedge b - c \in G$, and the nodes a and c are disconnected in G. We refer to a subset of vertices of a UG that form a maximal complete subgraph as a **clique**. A clique in a UG is called **external** if it contains at least one node that is not contained in any other cliques of that graph. A node, ξ, in a UG, G, is called an **external node** if $\Omega_G(\xi) \subseteq \Omega_G(x)$ for every $x \in \Omega_G(\xi)$. Every external clique contains at least one external node. We say a family of cliques provides an **edge cover** for a UG if every edge in the UG is contained in at least one of the cliques.

The **sink graph** of a UG, G, is a PDG, P, obtained by orienting an edge $a - b$ in G as (1) $a \to b$ if $\Omega_G(a) \subset \Omega_G(b)$, (2) $a \leftarrow b$ if $\Omega_G(a) \supset \Omega_G(b)$, (3) $a - b$ if $\Omega_G(a) == \Omega_G(b)$, and (4) $a \leftrightarrow b$ if neither (1), (2), or (3) are applicable. We use the notation $\Lambda(G)$ to denote the sink graph of graph G: $\Lambda(G) = P$. The **reduced sink graph** of a UG, G, is the PDAG, P, obtained by removing the doubly oriented

edges of its sink graph. We use the notation $\Lambda^o(G)$ to denote the reduced sink graph of graph G. We say a DAG, D, is in the **acyclic completion** of some PDAG, P, if D can be obtained by orienting the undirected edges in P. We say a DG, D, is in the **completion** of some PDG, P, if D can be obtained orienting the undirected edges in P (we are free to orient an undirected edge as $\leftrightarrow$, $\rightarrow$ or $\leftarrow$). When we write $a \rightarrow b \in P$ for some nodes a and b in some PDG (DG), P, we mean that a and b are connected in P either with a doubly oriented edge or through an edge oriented from a to b. However, when we say the edge connecting a and b is oriented as $a \rightarrow b$ in P, we specifically mean that the edge between a and b is not doubly oriented.

We say the node x_1 is an **ancestor** of the node x_m in the DG D, if there exists a sequence of nodes $\{x_1, ..., x_{m-1}, x_m\}$ where for any two immediate nodes in the sequence, x_i and x_{i+1}, $x_i \rightarrow x_{i+1} \in D$. We use the notation $x_1 \geq x_m \in D$ to indicate that x_1 is an ancestor of x_m in the DG D. We say a node x_m is a **descendant** of the node x_1 in a DG, D, if x_1 is an ancestor of x_m in D. For convenience, we assume every node is its own descendant as well as its own ancestor. We say a node, ξ, is a **root node** in a DAG if it does not have any ancestors. We say a node is a **root*** node in a DG if every ancestor of the node is also its descendant.

We say a PDAG, P, is **strongly transitive** if (1) for any triplet (x, y, z) forming a **directed path** in P, the edges between x, y and z are oriented as $x \rightarrow y \rightarrow z$ or $x - y \rightarrow z$ or $x \rightarrow y - z$ in P, the nodes x and z are also connected in P and the edge $x - z$ is oriented as $x \rightarrow z$ in P, and (2) for any triplet (x, y, z) forming an **undirected path** in P, the edges between x and y, and y and z are undirected in P, the nodes x and z are also connected in P and the edge $x - z$ is undirected in P also. We say a PDAG, P, is **weakly transitive** if (1) for any triplet (x, y, z) forming a directed path in P, the nodes x and z are also connected in P and the edge $x - z$ is either undirected in P or is oriented as $x \rightarrow z$ in P, and (2) for any triplet (x, y, z)

forming an undirected path in P the nodes x and z are also connected in P and the edge $x - z$ is either undirected in P or is oriented as $x \to z$.

We say a node, ξ, is **strongly transitive** in a PDG, P, if (1) whenever there is a directed path from ξ to node y through node x, then ξ and y are also connected in P and the edge between ξ and y is oriented as $\xi \to y$ in P, and (2) whenever there is an undirected path from ξ to node y through node x, then ξ and y are also connected in P through an undirected edge. We say a node, ξ, is **weakly transitive** in a PDG, P, if (1) whenever there is a directed path from ξ to node y through node x, then ξ and y are also connected in P and the edge between ξ and y is either undirected in P or is oriented as $\xi \to y$ in P, and (2) whenever there is an undirected path from ξ to node y through node x, then ξ and y are also connected in P and the edge $\xi - y$ is either undirected in P or is oriented as $\xi \to y$ in P.

5.2.1 Class Σ of Undirected Graphs

The following results are the foundations on which we build our causal theory of DCA: they characterize the causal knowledge encoded in a marginal dependency graph [59, 54, 53, 82]. These results assume the local causal Markov condition, which states a node is independent of its non-descendants conditional on its direct parents.

Lemma 1.1. An undirected graph, G, is generated by a DAG, if and only if any DAG in the acyclic completion of the reduced sink graph of G generates G.

Lemma 1.2. An undirected graph, G, is generated by the DAG D, only if, D is a subgraph of some DAG that is in the acyclic completion of the reduced sink graph of G.

Note 1 (regarding edge orientations in the sink graph). In the DAG graphical representation of a causal structure, an edge $a \to b$ records a as a direct cause of b. A

similar orientation in the sink graph, however, only implies that $b \not\geq a$, or b is not an ancestor (a direct or an indirect cause) of a.

Lemma 1.3. An undirected graph, G, is generated by a DAG, if and only if there exists a unique family of external cliques, Γ, of G that simultaneously provides an edge cover and a node cover for G.

Definition. Σ ***Class:*** We say an undirected graph G is in the Σ class when G can be generated by a DAG.

Note 2 (bounds on the generating causal structure). Lemmas 1.1 and 1.2 characterize two bounds for the target causal structure generating the observed marginal dependency relations. The reduced sink graph serves as an upper bound of the target causal structure; according to lemma 1.2, the target causal structure is a subgraph of a DAG that is in the acyclic completion of the reduced sink graph. Lemma 1.3 points to a set of minimal causal structures that can generate the sought for marginal dependency graph. These minimal causal structures correspond to any DAG obtained by (1) selecting *one* external node, ξ_i, for every external clique, and (2) connecting every external node in this selection to all the nodes in their boundaries: $\xi_i \rightarrow x_j, \forall x_j \in \Omega_G(\xi_i)$ [53].

These results assume the prior knowledge that the generating causal structure is acyclic. We will now show that irrespective of whether the generating causal structure is acyclic or not, the Σ class remains the same. The main factor that allows us to extend these lemmas in such a manner is that the local causal Markov condition is not the only bridge between the causal structure generating the data and the observed marginal dependency graph: we can, instead, reconcile this gap using the Reichenbach's common cause principle.

Lemma 1.4. An undirected graph, G, is generated by a DG, only if there exists a DAG generating G. (Supplementary material A.1.)

5.3 Causal Learning From Dependency Loss: Decision Problem

In the dependency loss setting, we take one condition to depict a functional system and the other to delineat a possible malfunction where some of the causal mechanisms have been disrupted and removed. In other words, we take one causal structure to be a subgraph of the other. In the dependency loss setting we use a slightly different noation and we distinguish the two conditions by labeling the marginal dependency graph of the functional system as G and its counterpart with G_1. We also use the notation D and D_1 to denote the corresponding causal structures: $a \to b \in D_1 \implies a \to b \in D$. Let us now start with the decision version of the dependency loss problem: given two marginal dependency graphs, G and G_1, with $G_1 \subseteq G$ and both in Σ, can we decide in polynomial time if there exist a pair of DAGs, D and D_1, with $D_1 \subseteq D$, that generate the given marginal dependency relations?

In answering this question, we introduce two new notions that will also be important to our causal interpretation of dependency change in section 3.2. These two concepts are mirrored sink graphs and the idea of a node's outer externality. These are extensions of the notions of a sink graph and a node's externality discussed previously in section 2.1.

[]-Mirrored Sink Graph Given a pair of undirected graphs, G and G_1, the $G-$mirrored sink graph of G_1, $\Lambda^G(G_1)$, is a partially directed graph whose skeleton is the same as that of the sink graph of G_1. The orientations of the edges of $P^G_{G_1}$ are obtained by transferring edge orientations from $\Lambda(G)$ to $\Lambda(G_1)$: $x \to y \in \Lambda^G(G_1) \iff$ x and y are connected in G_1 and $x \to y \in \Lambda(G) \vee x \to y \in \Lambda(G_1)$(see Table:5.1).

We also consider $\mho$-mirrored sink graphs, where $\mho$ is a collection of undirected graphs. The $\mho$-mirrored sink graph of G_1, $\Lambda^{\mho}(G_1)$, is a partially directed graph whose skeleton is the same as that of the sink graph of G_1, and the orientations of the edges of $\Lambda^{\mho}(G_1)$ are obtained by transferring edge orientations from the sink graph of every

undirected graph in $\mho$ to $\Lambda(G_1)$: $x \to y \in \Lambda^{\mho}(G_1) \iff x$ and y are connected in G_1 and $\exists G \in \mho \cup \{G_1\}$ s.t $x \to y \in \Lambda(G)$.

Reduced []-Mirrored Sink Graph Given a pair of undirected graphs, G and G_1, the reduced $G-$mirrored sink graph of G_1 is the partially directed acyclic graph obtained by removing the doubly oriented edges of the $G-$mirrored sink graph of G_1.

[]-Outer External Nodes Given two undirected graphs, G_1 and G, we say a node ξ is G_1-outer external in G, if $\forall x \in \Omega_{G_1}(\xi), \Omega_G(\xi) \subseteq \Omega_G(x)$.

Table 5.1: The orientation of the edges of the $G-$mirrored sink graph of G_1, $\Lambda^G(G_1)$, as a function of the orientations of $\Lambda(G)$ and $\Lambda(G_1)$, the sink graphs of G and G_1.

$\Lambda(G)$	$\Lambda(G_1)$	$\Lambda^G(G_1)$
$-$	$-$	$-$
$-$	$\to$	$\to$
$-$	$\leftarrow$	$\leftarrow$
$-$	$\leftrightarrow$	$\leftrightarrow$
$\to$	$-$	$\to$
$\to$	$\to$	$\to$
$\to$	$\leftarrow$	$\leftrightarrow$
$\to$	$\leftrightarrow$	$\leftrightarrow$
$\leftrightarrow$	$-$	$\leftrightarrow$
$\leftrightarrow$	$\to$	$\leftrightarrow$
$\leftrightarrow$	$\leftarrow$	$\leftrightarrow$
$\leftrightarrow$	$\leftrightarrow$	$\leftrightarrow$

Using these two constructs, we answer the decision version of the dependency loss problem in Theorems 1.1 and 1.2 below.

Theorem 1.1. Two undirected graphs, G and G_1, with G and G_1 both in the Σ class and G_1 a subgraph of G, are generated by DAGs, D and D_1, with D_1 a subgraph of D, if and only if any DAG in the acyclic completion of the reduced $G-$mirrored sink graph of G_1 generates G_1. (Supplementary material B.1.)

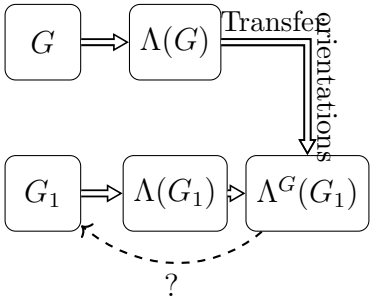
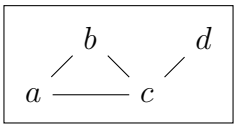


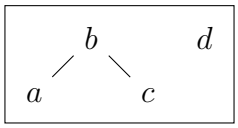
Figure 5.1: The flowchart of the process described in Theorem 1.1. To identify if there exists a pair of DAGs satisfying the conditions of the dependency loss problem for a given pair of marginal dependency graphs, G and G_1, we first form their sink graphs $\Lambda(G)$ and $\Lambda(G_1)$. Then, we transfer the edge orientations from $\Lambda(G)$ to $\Lambda(G_1)$, constructing the G-mirrored sink graph of G_1, $\Lambda^G(G_1)$. There exists a solution to the dependency loss problem, if and only if any DAG in the acyclic completion of the reduced G-mirrored sink graph of G_1 generates G_1.

Figure 5.1 shows the flowchart of the process described in Theorem 1.1. Figure 5.2 depicts an example application of Theorem 1.1, showing that there exists no pair of DAGs (D, D_1), $D_1 \subseteq D$, satisfying the conditions of the dependency loss problem for the marginal dependency graphs depicted.

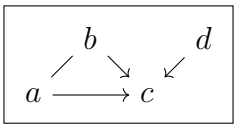
Theorem 1.2. Two undirected graphs, G and G_1, with G and G_1 both in the Σ class and G_1 a subgraph of G, are generated by DAGs, D and D_1, with $D_1 \subseteq D$, if and only if for every external clique of G_1, there exists at least one node external in G_1, ξ, that is also G_1-outer external in G: $\forall x \in \Omega_{G_1}(\xi) : \Omega_G(\xi) \subseteq \Omega_G(x)$. (Supplementary material B.2.)

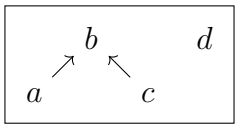
The main idea behind these theorems is straightforward. Let us assume that the two DAGs D and D_1 are a possible solution to the dependency loss problem: $\Sigma(D) = G$, $\Sigma(D_1) = G_1$, and $D_1 \subseteq D$. We explained that an edge $a \to b$ in $\Lambda(G)$, the sink graph of the marginal dependency graph of D, implies that b is not an ancestor of a in D (see *Note 1*). Now, since D_1 is assumed to be a subgraph D, then b cannot be an ancestor of a in D_1 either. Thus, every orientation in $\Lambda(G)$ not only encodes a



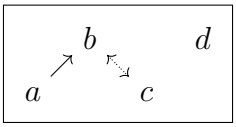


(a) Marginal dependency graphs G (on the left) and G_1 (on the right).





(b) Sink graphs $\Lambda(G)$ (on the left) and $\Lambda(G_1)$ (on the right).



(c) The G-mirrored sink graph of G_1, $\Lambda^G(G_1)$. The edge connecting b and c is dashed signifying that it will be removed in the reduced G-mirrored sink graph of G_1.

Figure 5.2: No DAG in the acyclic completion of the reduced G-mirrored sink graph of G_1 can generate G_1, since the nodes b and c are disconnected in all such DAGs. Therefore, according to Theorem 1.1, we conclude that there exists no pair of DAGs, D and D_1, with $D_1 \subseteq D$, that solve the depicted dependency loss problem.

constraint with respect to the causal relations in D, but the same constraint must also hold for the causal relations in D_1. In the sink transfer process we encode this constraint by preserving the orientation $a \to b \in \Lambda(G)$ when forming the G-mirrored sink graph of G_1.

Similar to the lemmas 1.1-1.3, these results can also be extended to instances where the generating causal structures are possibly cyclic and we have:

Theorem 1.3. Two undirected graphs, G and G_1, with G and G_1 both in the Σ class and $G_1 \subseteq G$, are generated by DGs, D and D, with $D_1 \subseteq D$ only if there exists two DAGs, D^* and D_1^*, with $D_1^* \subseteq D^*$, generating the target marginal dependency relations. (Supplementary material B.3.)

5.4 Causal Learning From Dependency Change: Decision Problem

In practice, the changes in marginal dependency relations across conditions have been observed to often times include both gain and loss of dependency relations. This implies that the data generating causal structure must have both experienced the removal of some causal mechanisms and also the incorporation of new causal mechanisms. Now, if we assume that there are no restrictions on where new causal mechanisms are admitted and which causal mechanisms can be removed, then a marginal dependency graph could change in any arbitrary manner from one condition onto the next. Therefore, as the causal structures are to be independent, then the causal information in one condition would not help resolve the causal uncertainty in the other. However, it is but natural to suggest that causal relations are not to be reversed across conditions. This leads us to consider the following acyclicity restrictions on the generating causal structures:

Weakly Acyclic Directed Graphs We say two directed graphs D_1 and D_2 are weakly acyclic if when a is ancestor of b in D_1, then b is an ancestor of a in D_2 only if b is an ancestor of a in D_1: $a \geq b \in D_1 \wedge b \geq a \notin D_1 \implies b \geq a \notin D_2$.

Strongly Acyclic Directed Graphs We say two directed graphs D_1 and D_2 are strongly acyclic if when a is ancestor of b in $D_1(D_2)$, then b is an ancestor of a in D, the union of the two DGs $D = D_1 \cup D_2$, only if b is an ancestor of a in $D_1(D_2)$: $a \geq b \in D_1 \wedge b \geq a \notin D_1 \implies b \geq a \notin D$.

5.4.1 Weak Acyclicity

If the intersection of the given marginal dependency graphs was empty, then there would always exists two weakly acyclic DAGs generating the target marginal dependencies; one could take any two DAGs in the completion of the reduced sink graphs of the two marginal dependency graphs and one would obtain two weakly

acyclic DAGs generating the sought for marginal dependency graphs. When the given marginal dependencies overlap, however, it is difficult to decide if a weakly acyclic solution to the dependency change problem exists or not. On the one hand, we are not aware of any polynomial time algorithms that could make such a decision. On the other hand, it would be challenging, if not impossible, to enforce the weak acyclicity constraint using only the orientations on the edges of the sink graphs of the marginal dependencies. The orientation on the edges of a sink graph encode constraints such as $a \not\geq b \in D_1$, while weak acyclicity requires constraints of the form $a \geq b \in D_1 \implies b \geq a \notin D_2 \vee b \geq a \in D_1$. For this reason, we replace the constraint of weakly acyclic DAGs, with a stronger condition[2], which we refer to as **intersection matched DAGs**, so that whenever there exists two intersection matched DAGs generating the target marginal dependencies, then there also exists two weakly acyclic DAGs.

Intersection Matched Directed Graphs We say two directed graphs, D_1 and D_2, are intersection matched if $\Sigma(D_1 \cap D_2) = \Sigma(D_1) \cap \Sigma(D_2)$.

Theorem 2.1. Given a pair of undirected graphs, G_1 and G_2, with G_1 and G_2 both in Σ, there exists a pair of intersection matched DAGs generating the given marginal dependency graphs, if and only if any DAG in the acyclic completion of the reduced $\{G_1, G_2\}-$mirrored sink graph of $G_\cap$, $G_\cap = G_1 \cap G_2$, generates $G_\cap$. (Supplementary material C.1.)

Theorem 2.2. Given a pair of undirected graphs, G_1 and G_2, with G_1 and G_2 both in Σ, there exists a pair of intersection matched DAGs generating the given marginal dependency graphs, if and only if for every external clique of $G_\cap$,

[2]Intersection matched constraint neither implies or is implied by weak acyclicity. We say intersection matched is a stronger condition in the sense that whenever there exists two intersection matched DAGs generating the target marginal dependencies, then there also exists two weakly acyclic DAGs.

$G_\cap = G_1 \cap G_2$, there exists at least one node that is (1) external in $G_\cap$, and (2) $G_\cap$-outer external in G_1 and G_2. (Supplementary material C.2.)

Similar to the previous sections, we can extend these results to directed graphs:

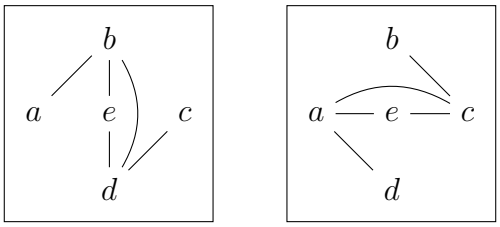
Theorem 2.3. Given a pair of undirected graphs, G_1 and G_2, with G_1 and G_2 both in Σ, there exists a pair of intersection matched DGs generating the given marginal dependency graphs, only if there exists a pair of intersection matched DAGs generating G_1 and G_2. (Supplementary material C.3.)

Finally, we have:

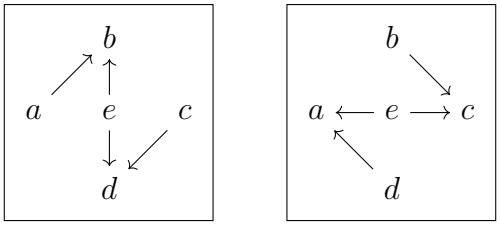
Theorem 2.4. If there exists a pair of intersection matched DAGs generating undirected graphs G_1 and G_2, then there exists a pair of weakly acyclic DAGs generating G_1 and G_2. (Supplementary material C.4.)

5.4.2 Strong Acyclicity

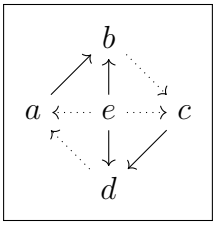
When the intersection of the target marginal dependencies is empty, unlike the case for weak acyclicity, we can no longer guarantee a strong acyclic solution (consider the example marginal dependencies shown in Figure 5.3). Then, the intersection matched constraint of the previous section is no longer a sufficient condition when it comes to searching for a strong acyclic solution. Searching for a strong acyclic solution would require one to iterate over all the possible pair of DGs generating the target marginal dependencies, forming their union, and validating if their union has cycles only where there are cycles in one of the generating DGs. We are not aware of any polynomial time algorithm that can find a way around this exponential search. Similar to the case for weak acyclicity, we instead enforce a relaxed form of faithfulness, marginal and ancestral faithfulness, so that whenever there exists a solution comprising of a pair of DGs that are marginally and ancestrally faithful, and also intersection matched, then there exists a strongly acyclic solution as well.



(a) Marginal dependency graphs G_1 (on the left) and G_2 (on the right).



(b) The only two DGs that can generate the target marginal dependencies.



(c) A strong acyclic solution does not exist, since in the only possible solution the union of the two causal structures contains a cycle, $C = \{a, b, c, d\}$.

Figure 5.3: Even when the intersection of the target marginal dependencies is empty we cannot guarantee a strong acyclic solution.

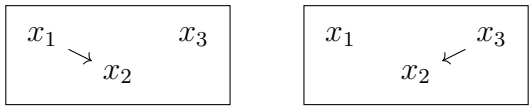
Marginally Faithful Directed Graphs We say two directed graphs D_1 and D_2 are marginally faithful if $\Sigma(D_1 \cup D_2) = \Sigma(D_1) \cup \Sigma(D_2)$.

Ancestrally Faithful Directed Graphs We say two directed graphs D_1 and D_2 are ancestrally faithful if $(D_1 \cup D_2)^+ = D_1^+ \cup D_2^+$. The operator $(.)^+$ returns the transitive closure of the input directed graph.

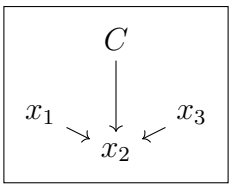
Before we show how these constraints help us find a strong acyclic solution, we briefly discuss our reasoning behind referring to these conditions as some relaxed form

of faithfulness. In the domain adaptation setting, the change in the parameters of a causal structure are graphically captured with a directed graph where the condition or the context variable is included as an exogenous node along the nodes corresponding to the other variables in the domain. This node is then set as a direct parent of those variables whose generating causal mechanisms are modified across conditions. We can use the same graphical representation to capture unstable statistical dependencies and causal mechanisms. Consider for instance the causal structures shown in Figure 5.4.a . Let us refer to these causal structures as D_1 and D_2. To capture the different causal mechanisms in play across the two conditions, we take the union of D_1 and D_2, $D = D_1 \cup D_2$, and further include a fourth node corresponding to the conditioning or the context variable and connect it to the node whose generating causal mechanism is unstable across the two conditions (Figure 5.4.b). Now, while this graphical representation succinctly captures the causal relations in the domain across the two conditions, it is not a faithful representation of the original probability distribution implied by the causal structures in Figure 5.4.a . According to the graph in Figure 5.4.b, the nodes x_1 and x_3 are d-connected conditional on the set $\{x_2, C\}$. However, in neither contexts, the variables x_1 and x_3 are dependent conditional on x_2, since in the corresponding causal structures, i.e. Figure 5.4.a, these two nodes are d-separated conditioned on x_2. Faithfulness would require any two sets of nodes, X and Y, to be d-connected in D given a third set of nodes Z, only when X and Y are d-connected given Z in D_1 or D_2. It can be easily shown that faithfulness, in this sense, requires D_1 and D_2 to be a pair of marginally faithful and ancestrally faithful DGs. We now show how marginal and ancestral faithfulness can help us construct a strongly acyclic solution using the **merged sink graph**.

Merged Sink Graph Given a pair of undirected graphs, G_1 and G_2, their intersection $G_\cap = G_1 \cap G_2$, and their union $G = G_1 \cup G_2$, we call the $\{G_1, G_2, G_\cap\}-$mirrored



(a) The causal relations in effect in the domain under two different conditions.



(b) We can graphically capture the change in the causal relations by including the condition or the context variable as an exogenous node and connecting it to the node x_2.

Figure 5.4: While the graphical representation in Figure 5.4.b succinctly captures the causal relations in the domain, it is not a faithful representation of the probability distribution implied by the causal structures shown in Figure 5.4.a .

sink graph of G the merged sink graph of G_1 and G_2. The merged sink graph of G_1 and G_2 can be obtained transferring edge orientations from $\Lambda(G_1)$, $\Lambda(G_2)$, and $\Lambda(G_\cap)$ to $\Lambda(G)$: $x \to y \in \Lambda^{\{G_1,G_2,G_\cap\}}(G) \iff x$ and y are connected in G and $x \to y \in \Lambda(G) \vee x \to y \in \Lambda(G_1) \vee x \to y \in \Lambda(G_2) \vee x \to y \in \Lambda(G_\cap$.

Theorem 3.1. Given two marginal dependency graphs G_1 and G_2, there exists two ancestrally faithful, marginally faithful, intersection matched DAGs generating G_1 and G_2 if and only if the following three conditions hold:

(1) For every external clique of G_1 , there exists at least one node that is (1.1) external in G_1, (1.2) G_1-outer external in $G = G_1 \cup G_2$, (1.3) external in $G_\cap = G_1 \cap G_2$, (1.4) $G_\cap$-outer external in G_2, and (1.5) ***weakly*** transitive in $\{G_1, G_2, G_\cap\}-$mirrored sink graph of G, the merged sink graph of G_1 and G_2.

(2) For every external clique of G_2 , there exists at least one node that is (2.1) external in G_2, (2.2) G_2-outer external in G , (2.3) external in $G_\cap$, (2.4) $G_\cap$-outer external in G_1, and (2.5) ***weakly*** transitive in the merged sink graph of G_1 and G_2.

(3) For every external clique of $G_\cap$, there exists at least one node that is (3.1) external in $G_\cap$, (3.2) $G_\cap$-outer external in G_1, G_2, and G, and (3.3) ***weakly*** transitive in the merged sink graph of G_1 and G_2. (Supplementary material D.1.)

Theorem 3.2. Given two marginal dependency graphs G_1 and G_2, there exists an ancestrally faithful, marginally faithful, intersection matched pair DGs generating G_1 and G_2 only if there exists an ancestrally faithful, marginally faithful, intersection matched pair of DAGs generating the given marginal dependency graphs. (Supplementary material E.1.)

Theorem 3.3. If there exists an ancestrally faithful, marginally faithful, intersection matched pair of DGs generating G_1 and G_2, then there exists a strongly acyclic pair of DGs generating G_1 and G_2. (Supplementary material E.2.)

5.5 Causal Learning From Dependency Change: A Novel Causal Discovery Tool

Consider an instance of dependency loss problem where the marginal dependencies are denoted with G and G_1. Often, for a given (G, G_1), there exists more than one pair of DGs satisfying the conditions of the dependency loss problem. We refer to this collection of pairs of DGs as the solution-set of the dependency loss problem and denote it with $\mathcal{S}(G, G_1)$:

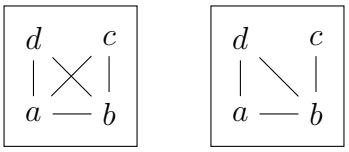
$$\mathcal{S}(G, G_1) = \{(D, D_1) | D_1 \subseteq D, \Sigma(D) = G, \Sigma(D_1) = G_1\}.$$

We sometimes write $D \overline{\in} \mathcal{S}(G, G_1)$ to imply that there exists at least one DG, D_1, such that $(D, D_1) \in \mathcal{S}(G, G_1)$. Similarly, we write $D_1 \underline{\in} \mathcal{S}(G, G_1)$ to imply that there exists at least one DG, D, such that $(D, D_1) \in \mathcal{S}(G, G_1)$.

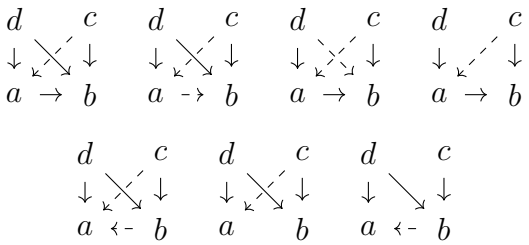
We are interested in learning a causal explanation of the observed dependency loss: that is, we would would like to characterize the set $\Delta(G, G_1)$:

$$\Delta(G, G_1) = \{\delta D | \delta D = D - D_1, (D, D_1) \in \mathcal{S}(G, G_1)\}.^3$$

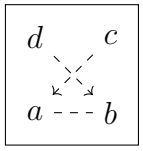
An example $\mathcal{S}(G, G_1)$ and $\Delta(G, G_1)$ are shown in Figure 5.5.



(a) G_1 is plotted on the left, G_2 is plotted on the right.



(b) The set of all pairs (D, D_1) satisfying the conditions of the dependency loss problem for the marginal dependency structures shown in Figure 5.5-(a). There are only seven such pairs in the (acyclic) solution-set. A pair $(D, D_1) \in \mathcal{S}(G, G_1)$ is shown here by a single DAG: D corresponds to the DAG over both the dashed and the solid edges, and D_1 corresponds to the DAG with the dashed edges removed.



(c) Upper bound for the set $\Delta(G, G_1)$

Figure 5.5: An example $\mathcal{S}(G, G_1)$ and $\Delta(G, G_1)$.

We now show that an upper bound for $\Delta(G, G_1)$ can be easily characterized in terms of the *differences* between the sink graph of G and the $G-$mirrored sink graph of G_1. These differences are calculated using the Λ_δ operator:

Definition. Λ_δ **operator** Given a set of UGs, $\mho$, and two undirected graph G_1 and G, the operator Λ_δ outputs a PDG, $\Lambda_\delta(\mho, G, G_1)$, whose skeleton is a subgraph of the sink graph of G, and whose edge orientations are obtained from the sink graph of

[2] $D_i - D_j$ refers to the DG constructed by removing all the edges in D_j from D_i. Similarly, $G_i - G_j$ refers to the UG constructed by removing all the edges in G_j from G_i.

G and the $\mho \cup \{G\}-$mirrored sink graph of G_1 according to the procedure shown in table 5.2.

Table 5.2: The orientation of the edges $\Lambda_\delta(\mho, G, G_1)$, as a function of the orientation of the edges of sink graph of G, the $\mho-$mirrored sink graph of G_1. An empty cell signifies an edge that is removed.

$\Lambda(G)$	$\Lambda^{\mho\cup\{G\}}(G_1)$	$\Lambda_\delta(\mho, G, G_1)$
$-$		$-$
$-$	$\leftrightarrow$	$-$
$-$	$\rightarrow$	$\leftarrow$
$-$	$-$	
$\rightarrow$		$\rightarrow$
$\rightarrow$	$\leftrightarrow$	$\rightarrow$
$\rightarrow$	$\rightarrow$	
$\leftrightarrow$		
$\leftrightarrow$	$\leftrightarrow$	

To see how using Λ_δ we can estimate an informed upper bound on $\Delta(G, G_1)$, first, let us remove a set of undesirable solutions in the solution-set, $\mathcal{S}(G, G_1)$. It is often the case that for any $D \overline{\in} \mathcal{S}(G, G_1)$, one can find multiple D_1's corresponding to that choice of D in the solution-set. Let us refer to the set of all such D_1's for some choice of D with $S_{(G,G_1)}(D)$:$S_{(G,G_1)}(D) = \{D_1 | (D, D_1) \in \mathcal{S}(G, G_1)\}$. We immediatley note that $D_1^* = \cup_{D_1 \in S_{(G,G_1)}(D_1)} D_1$ also belongs to $S_{(G,G_1)}(D)$ and it is the one with the minimum Hamming distance to D (the number of edge deletions required to modify D to a $D_1 \in S_{(G,G_1)}(D)$ is the smallest for D_1^*). From the pairs $(D, D_1) \in \mathcal{S}(G, G_1)$, we only consider those where D_1 corresponds to the DG(DAG) with the minimum Hamming distance to its D and remove all other solution instances; it can be easily shown that this DG(DAG) exists and is unique. From now on, when using the notations $\mathcal{S}(G, G_1)$ and $\Delta(G, G_1)$, we assume that this pre-processing step has been applied. We can now show:

Theorem 4.1. $\Lambda_\delta(\{G\}, G, G_1)$ is a tight upper bound for $\Delta(G, G_1)$. That is, for all δD in $\Delta(G, G_1)$, δD is a subgraph of $\Lambda_\delta(\{G\}, G, G_1)$: $a \to b \in \delta D$ only if $a \to b$ or $a - b$ are in $\Lambda_\delta(\{G\}, G, G_1)$. (Supplementary material F.1.)

We can easily extend this result and argue that in the case of dependency change, given marginal dependency relations G_1, G_2, and their intersection $G_\cap$, if the solution-set is restricted to those DGs that are intersection matched, a similar upper bound can be characterized in terms of $\Lambda_\delta(\{G_1, G_2\}, G_1, G_\cap)$:

Corollary $\Lambda_\delta(\{G_1, G_2\}, G_1, G_\cap)$ provides an upper bound for $\Delta(G_1, G_\cap)$. That is, for all δD in $\Delta(G_1, G_\cap)$, δD is a subgraph of $\Lambda_\delta(\{G_1, G_2\}, G_1, G_\cap)$: $a \to b \in \delta D$ only if $a \to b$ or $a - b$ are in $\Lambda_\delta(\{G_1, G_2\}, G_1, G_\cap)$. (Supplementary material F.2.)

5.6 Simulations

The goal of this paper is to evaluate DCA as a causal discovery tool. In the previous two sections we showed that, under certain assumptions, we can find a causal explanation that matches the observed changes in marginal dependency relations in polynomial time. We further characterized a tight upper bound for the set of all causal mechanisms whose impaired function can account for the observed unstable marginal dependency relations. The important question that we have not yet answered concerns characterizing the causal information content of DCA. Consider an instance of dependency loss problem where the marginal dependencies are denoted with G and G_1. We are interested in obtaining a measure of the causal information content of DCA (or equivalently its causal uncertainty): that is, we would would like to characterize the cardinality of the set $\Delta(G, G_1)$.

If the set $\Delta(G, G_1)$ contains only few elements, then DCA, as a causal discovery tool, offers a very high causal resolution with a desirable causal information content. On the other hand, if the cardinality of the set $\Delta(G, G_1)$ is large, then we can only

ascertain the causal origins of the unstable marginal dependence relations with an equally large uncertainty. Unfortunately, we don't know of any easy way to enumerate the elements of this set as a function of the marginal dependencies G and G_1. In this section, we use a series of simulations to examine this question and to obtain a measure of the causal uncertainty of DCA. Our simulations suggest that the causal uncertainty of DCA in uncovering the causal origins of K unstable marginal dependence relations is similar to the causal uncertainty one would face if one were to come up with a causal explanation for a marginal dependence graph with K edges.

First let us characterize the complexity of our proposed bound $\Lambda_\delta(\{G\}, G, G_1)$ and examine the number of edges or causal mechanisms that are implicated by this bound. In practice, experts often use the unstable marginal dependence relations as pointers and a springboard to causal postulates. Our simulations suggest that our proposed bound has a complexity lower than the set of unstable marginal dependence relations. Thus, an expert who has been successful in sifting through the set of unstable marginal dependence relations to find a suitable causal explanation should find it no more difficult to sift through the causal mechanism proposed by the bound $\Lambda_\delta(\{G\}, G, G_1)$. In our simulations we construct a random DAG, D, with $m = \{10, 12, 14, 16, 18, 20, 25\}$ nodes such that the *expected number of neighbours* of any nodes is $nn = 3$. We then randomly choose and remove $l = \{1, 3, 5, 7\}$ causal mechanisms from the DAG we had constructed in the previous step. We compare the number of marginal dependency relations affected to the number of edges in the bound $\Lambda_\delta(\{G\}, G, G_1)$ where G and G_1 correspond to the marginal dependency graphs of the causal structures constructed. We repeat this process for each parameter setting for 1000 iterations. The average ratio of the number of edges in the bound $\Lambda_\delta(\{G\}, G, G_1)$ to that of the count of unstable marginal dependency relations, together with the Hamming distance between the two structures are reported in Fig 5.6.

Next, to evaluate the causal information content of DCA, we need a suitable *x-axis* that provides a good measure of the difficulty or the challenge of identifying the causal origins of the changes in marginal dependency relations. The network size, the sparsity, and the number of unstable causal mechanisms can all contribute to the difficulty of resolving the responsible unstable causal mechanisms. However, even in the same DAG, removing some causal mechanisms can cause rather large changes in marginal dependency relations and removing some other causal mechanisms may barely affect the marginal dependence relations in the domain. This is shown in Fig 5.7 where the number of unstable marginal dependency relations are plotted as a function of the network size and the number of unstable causal mechanisms. Our previous results in Fig: 5.6 suggests that a good x-axis to denote the complexity or the challenge of resolving the responsible faulty causal mechanisms is that of the number of unstable marginal dependence relations. The higher the number of unstable marginal dependence relations, the higher the number of causal mechanisms that are implicated by our proposed bound $\Lambda_\delta(\{G\}, G, G_1)$.

Next, to evaluate the causal information content of DCA, instead of enumerating the set $\Delta(G, G_1)$, we compare the causal mechanisms implicated by the bound $\Lambda_\delta(\{G\}, G, G_1)$ to the actual causal mechanisms that were responsible for the observed unstable marginal dependency relations. We first construct a random DAG, D, with $m = \{10, 12, 14, 16, 18, 20, 25\}$ nodes such that the *expected number of neighbours* of any nodes is $nn = 3$. We then randomly choose and remove $l = \{1, 3, 5, 7\}$ causal mechanisms from the DAG we had constructed in the previous step. Next, we form the partially directed graph of the bound $\Lambda_\delta(\{G\}, G, G_1)$ where G and G_1 correspond to the marginal dependency graphs of the causal structures constructed. Finally we measure how precisely the bound $\Lambda_\delta(\{G\}, G, G_1)$ represents the causal mechanisms that were removed from D and we further calculate the Hamming distance between

the two. We repeat this process for each parameter setting for 1000 iterations. The average precision and the average Hamming distance are plotted as a function of the number of unstable marginal dependency relations in Figure 5.8. In the same figure we also include the average precision and the average Hamming distance of the heuristic bound corresponding to that unstable marginal dependence relations. Compared to this heuristic that experts have often deployed in their research, our proposed upper bound offers a significantly better precision.

We note here that the precision offered through the bound $\Lambda_\delta(\{G\}, G, G_1)$ in uncovering the causal origins of K unstable marginal dependence relations is similar to the precision one can hope for if one were to come up with a causal explanation for a marginal dependence graph with K edges. This is shown in Figure 5.9 where we show the precision a sink graph of a marginal dependency graph offers in identifying the causal structure that generated the marginal dependency graph.

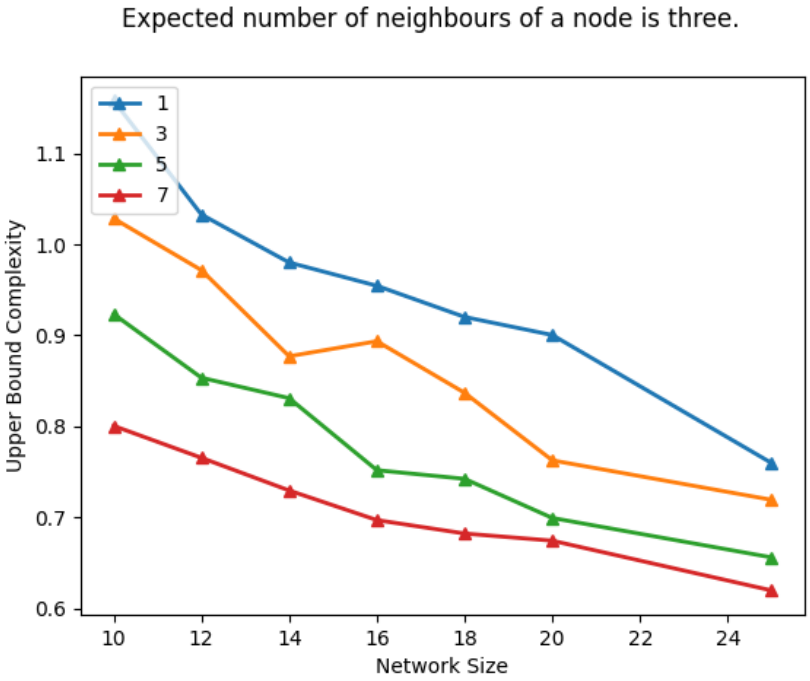
Finally, our simulations suggest that our proposed upper bound offers a desirable precision when it comes to identifying the nodes whose causal mechanisms are affected using simply the information pertaining to unstable and stable marginal dependency relations. Here, we again first construct a random DAG, D, with $m = \{10, 12, 14, 16, 18, 20, 25\}$ nodes such that the *expected number of neighbours* of any nodes is $nn = 3$. We then randomly choose and remove $l = \{1, 3, 5, 7\}$ causal mechanisms from the DAG we had constructed in the previous step. Next, we form the partially directed graph of the bound $\Lambda_\delta(\{G\}, G, G_1)$ where G and G_1 correspond to the marginal dependency graphs of the causal structures constructed. We then measure how precisely the nodes with incoming edges in $\Lambda_\delta(\{G\}, G, G_1)$ represent the nodes whose generating mechanisms were manipulated in the experiment. Again, we plot the average precision as a function of the number of unstable marginal dependency relations in Figure 5.10. In the same figure we also include the average precision

offered by the heuristic bound corresponding to that unstable marginal dependence relations. Once again, compared to this heuristic that experts have often deployed in their research, our proposed upper bound offers a significantly better precision.

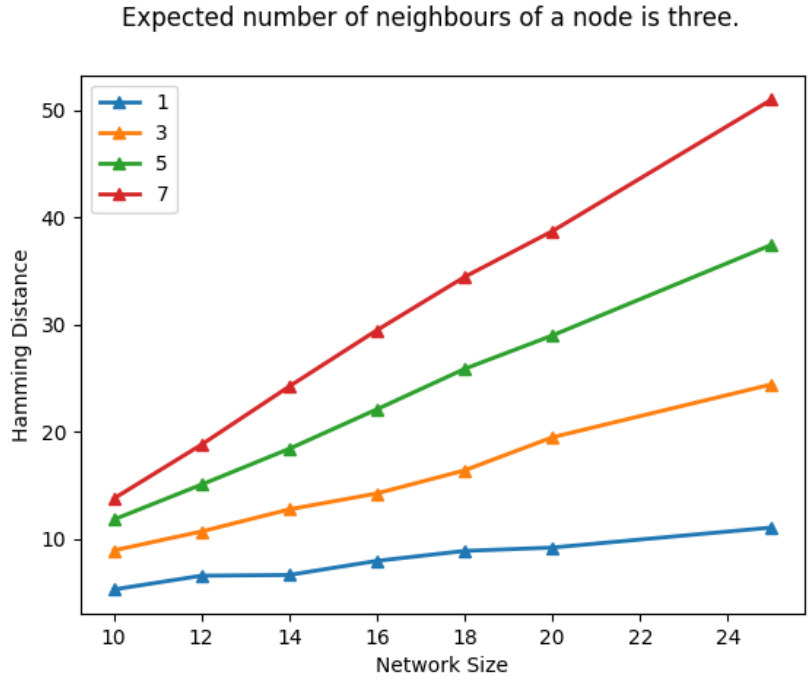
5.7 Conclusions and Future Work

In this paper, we proposed a novel causal discovery tool for tracing patterns of dependency change to their causal origin. Unlike the standard approaches to causal discovery—where causal relations are learned from statistical dependencies that hold in a domain—our causal discovery tool constructs a causal object based on the *differential* dependencies between two conditions. Furthermore, the theory presented is not bounded by the limitations of the local causal Markov condition: thus, our results are still applicable even when the generating causal structure contains cycles.

There are still many avenues left to explore in this setting. For instance, we only considered a causal characterization of dependency change across two conditions: a major expansion would be to develop this theory for handling an arbitrary number of conditions and contexts. Our theoretical results also permit the transfer of causal information from one domain to another. Consider, for instance, Theorem 1.1, where by forming the mirrored sink graph, we obtain a better picture of the causal structure underlying one condition by observing the dependencies in another condition. We expect the problem setting we studied in this paper to have significant practical application, especially in the biomedical field, where differential marginal dependency relations have often helped guide experts to identify the causal origins of a disease.



(a) The ratio of the number of edges in the bound $\Lambda_\delta(\{G\}, G, G_1)$ to that of the count of unstable marginal dependency relations as a function of network size, m, and the number of unstable causal mechanisms $l = \{1, 3, 5, 7\}$ averaged over 1000 randomly sampled DAGs with $nn = 3$.



(b) The Hamming distance between the bound $\Lambda_\delta(\{G\}, G, G_1)$ and the undirected graph corresponding to that of unstable marginal dependency relations as a function of network size, m, and the number of unstable causal mechanisms $l = \{1, 3, 5, 7\}$ averaged over 1000 randomly sampled DAGs with $nn = 3$

Figure 5.6: The number of edges implicated by the bound $\Lambda_\delta(\{G\}, G, G_1)$ is almost always smaller than the number of unstable marginal dependency relations.

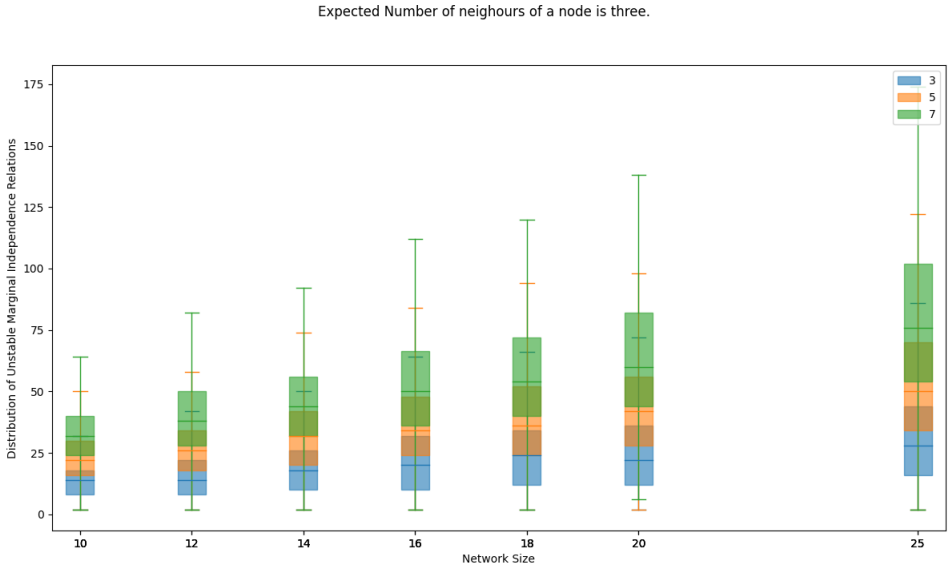
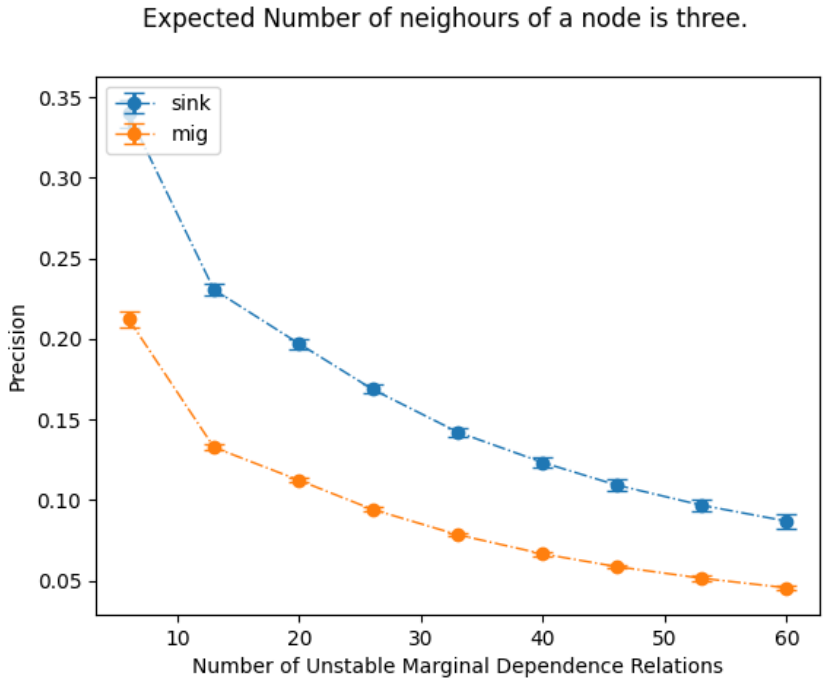
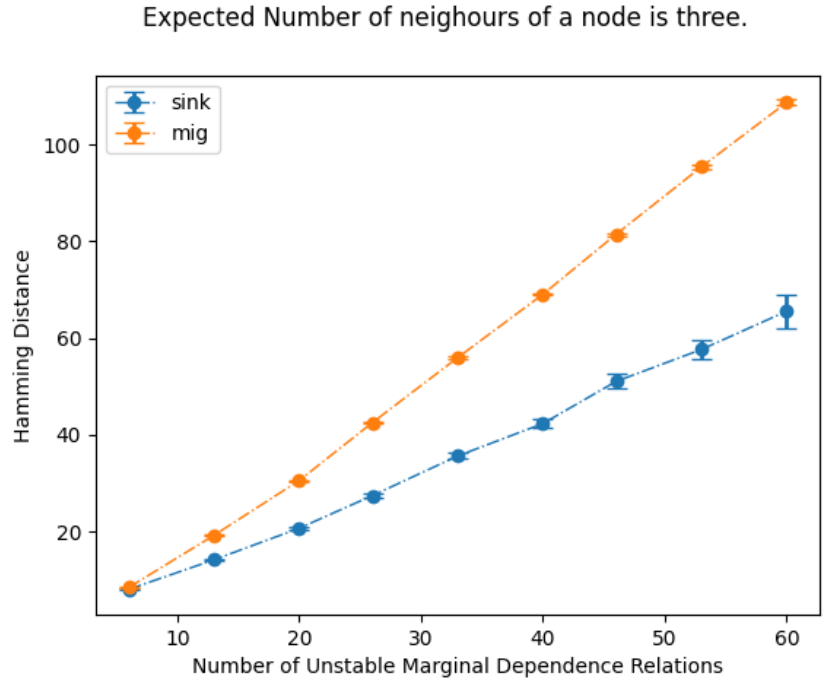


Figure 5.7: The number of unstable marginal dependency relations as a function of the network size, $m = \{10, 12, 14, 16, 18, 20, 25\}$, and the number of causal mechanisms that are manipulated $l = \{3, 5, 7\}$.

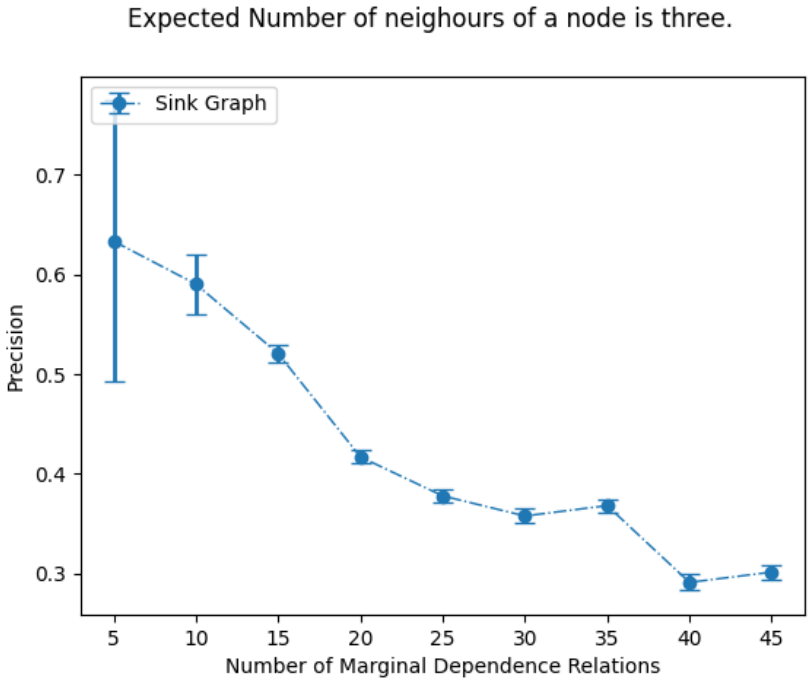


(a) The precision by which the bound $\Lambda_\delta(\{G\}, G, G_1)$ (the blue curve) and the heuristic bound corresponding to the unstable marginal dependency relations (the blue curve) represent the $l = \{1, 3, 5, 7\}$ causal mechanisms that had caused the unstable marginal dependency relations. The precision is plotted as a function of the number of unstable marginal dependency relation.

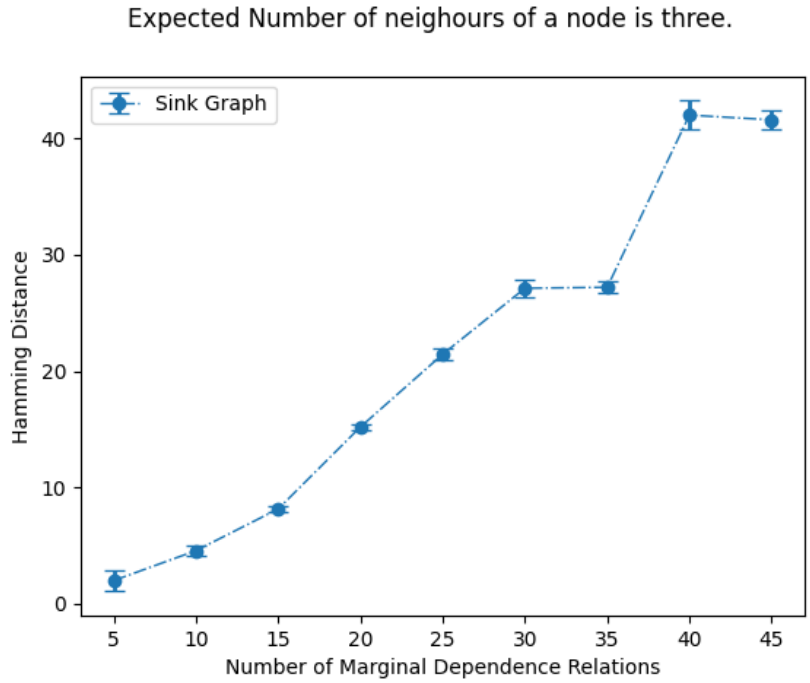


(b) The Hamming distance from the bound $\Lambda_\delta(\{G\}, G, G_1)$ (the blue curve) and the heuristic bound corresponding to the unstable marginal dependency relations (the blue curve) to the $l = \{1, 3, 5, 7\}$ causal mechanisms that had caused the unstable marginal dependency relations. The Hamming distance is plotted as a function of the number of unstable marginal dependency relation.

Figure 5.8: Compared to the heuristic bound corresponding to the unstable marginal



(a) The precision by which the sink graph of a marginal dependency graph represents the causal structure generating the sought for marginal dependency relations.



(b) The Hamming distance between the sink graph of a marginal dependency graph and the causal structure generating the sought for marginal dependency relations.

Figure 5.9: The precision offered through the bound $\Lambda_\delta(\{G\}, G, G_1)$ in uncovering the causal origins of K unstable marginal dependence relations is similar to the precision one can hope for if one were to come up with a causal explanation for a marginal dependence graph with K edges.

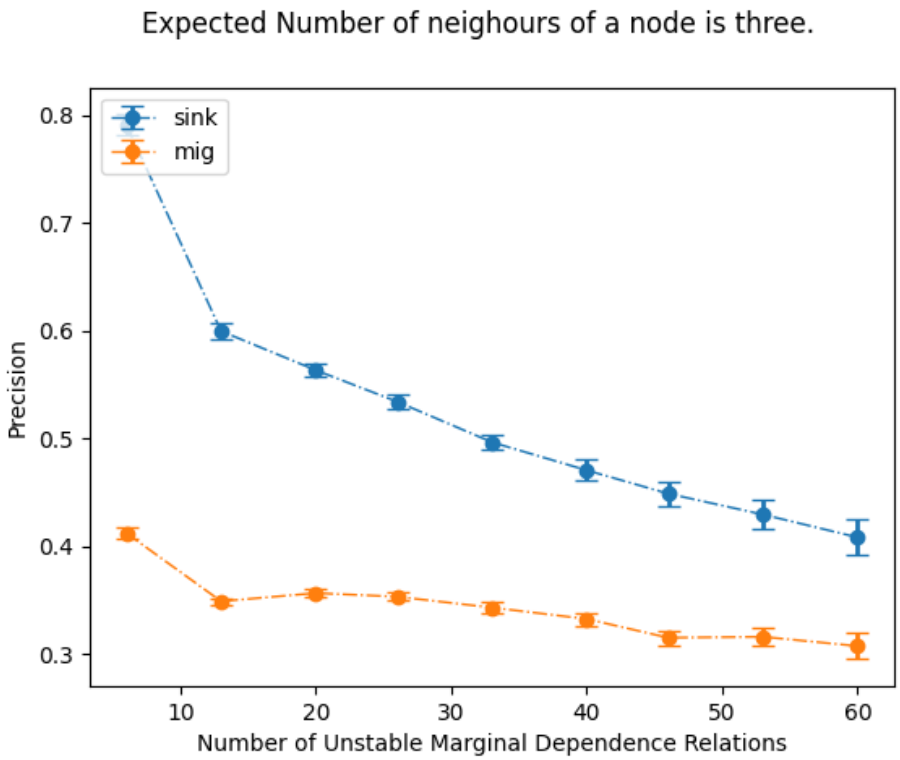


Figure 5.10: The precision offered by the bound $\Lambda_\delta(\{G\}, G, G_1)$ in identifying the nodes whose generating causal mechanisms were manipulated (in blue curve) compared to the precision of the heuristic bound of unstable marginal dependency relations.

5.8 Chapter 5 Appendix

A.1

Lemma 1.4. An undirected graph, G, is generated by a DG, only if there exists a DAG generating G.

Proof. To prove this lemma we use Lemma S.A.1 below. Lemma S.A.1 states an undirected graph, G, is generated by a DG graph only if *any* DG in the completion of the reduced sink graph of G generates G. Since there is at least one DAG in the completion of the reduced sink graph of G [54], by Lemma S.A.1, this DAG must generate G. We conclude an undirected graph is generated by a DG only if there exists a DAG generating it.

Lemma S.A.1. An undirected graph, G, is generated by a DG, only if *any* DG in the completion of the reduced sink graph of G generates G.

We first show for any DG, D, in the completion of the reduced sink graph of some UG, G, the marginal dependency graph of D is a subgraph of G, $\Sigma(D) \subseteq G$. In the second part, we show if this UG is known to have been generated by some DG D^*, then the marginal dependency graph of any DG, D, in the completion of the reduced sink graph of G, will be equal to G.

To show for any DG, D, in the completion of the sink graph of some UG, G, the marginal dependency graph of D is a subgraph of G, $\Sigma(D) \subseteq G$, we first show any such D is *transitive*. Using this result, we show two nodes share a common ancestor in D only if they are directly connected in G.

Consider any *directed path* $x_1 \to x_2 \to ... \to x_n$ in D, where D is any DG in the completion of the reduced sink graph of some undirected graph G. An edge, $x_i \to x_{i+1}$, is oriented as such in D, only if it is oriented as $x_i \to x_{i+1}$ or it is undirected in the sink graph of G. This implies the boundary of x_i in G is a subset of or equal to the

boundary of x_{i+1}, $\Omega_G(x_i) \subseteq \Omega_G(x_{i+1})$ [53]. Therefore, the above directed path lets us conclude (1) x_1 is in the boundary of x_n in G, and (2) $\Omega_G(x_1) \subseteq \Omega_G(x_n)$. This result is what we meant by transitivity of D.

Now, we show two nodes, x_i and x_j, have a common ancestor in D, only if x_i and x_j are directly connected in G, implying $\Sigma(D) \subseteq G$. Let's assume z is a common ancestor x_i and x_j in D. The argument in the previous paragraph lets us conclude (1) z is directly connected to both x_i and x_j in G, and (2) boundary of z in G is a subset of the boundary of both x_i and x_j. As $x_j \in \Omega_G(z)$ and $\Omega_G(z) \subseteq \Omega_G(x_i)$, we conclude x_j must be in the boundary of x_i in G. This proves that for any DG, D, in the completion of the reduced sink graph of some UG, G, the marginal dependency graph of D is a subgraph of G.

We now show if G is generated by some DG, D^*, any DG, D, in the completion of reduced the sink graph of G, *implies* all the edges in G. We prove this by showing those edges that are removed from G when its reduced sink graph is formed are necessarily *implied* in any such D.

Any such must have been doubly oriented in the sink graph of G. Consider such a doubly oriented edge $a \rightarrow b \leftrightarrow c \leftarrow d$. Since there is a doubly oriented edge between b and c in the sink graph of $G = \Sigma(D^*)$, there cannot be any directed paths between b and c in D^*. If there was a directed path from b to c in D^*, b would have been an ancestor of c in D^* and $\Omega_{\Sigma(D^*)}(b) \subseteq \Omega_{\Sigma(D^*)}(c)$—there would be no doubly oriented edge between b and c in the sink graph of G if the boundary of b in G were to be a subset of (or equal to) the boundary of c. Therefore, since b and c are connected in $\Sigma(D^*)$, and since there is no directed path in D^* from either b to c or c to b, we conclude they must have had a common ancestor, $z(z \neq b, z \neq c)$, in D^*. This common ancestor, z, must be directly connected to both b and c in G since G is the marginal dependency graph of D^*. Furthermore, z cannot be connected to neither d

nor a in G. If d was connected to z in G, then d must have also been connected to b: z is assumed to be an ancestor of b in D^* and any node in the boundary of z in $G = \Sigma(D^*)$ must also be in the boundary of b ($\Omega_G(z) \subseteq \Omega_G(b)$). This lets us conclude that the edges $z - b$ and $z - c$ in G are oriented as $z \to b$ and $z \to c$ in the sink graph of G (d is a node in the boundary of c not connected to z; $d - c - z$ form a v-structure and force $c - z$ to be oriented as $c \leftarrow z$; $c - z$ cannot be doubly oriented since z is an ancestor of c in D^* and $\Omega_G(z) \subseteq \Omega_G(c)$). Therefore, in any DG in the completion of the reduced sink graph of G, b and c share z as a common ancestor, and they are directly connected in the marginal dependency graph of any such DG.

□

B.1

To prove the theorem 1.1, we first require the following two results:

Lemma S.B.1. The reduced sink graph of any UG, G, does not contain a *semi-directed cycle*. A semi-directed cycle in a PDG, P, is a sequence of nodes $\{x_1, ..., x_n, x_1\}$, where, (1) for every two consecutive nodes, x_i and x_{i+1}, in the sequence, x_i and x_{i+1} are connected in P and the edge $x_i - x_{i+1}$ is either undirected in P or it is oriented as $x_i \to x_{i+1}$ in P, and (2) at least *one* such edge in the sequence is directed.

Proof. An edge is oriented as $a \to b$ in the reduced sink graph of some undirected graph, G, only if the boundary of a is a subset of the boundary of b in G [53]. Similarly, two nodes are connected by an undirected edge, $a - b$, in the sink graph only if that the boundaries of the nodes a and b are the same in G, $\Omega_G(a) = \Omega_G(b)$. Now, assume we have a semi-directed cycle, $C = \{x_1, ..., x_n, x_1\}$, in the sink graph of some undirected graph G. There is at least one directed edge in this sequence: without loss of generality, let's then say the edge connecting x_n and x_1 is oriented as $x_n \to x_1$ in the sink graph

of G. We then conclude that $\Omega_G(x_n) \subset \Omega_G(x_1)$. On the other hand, the sequence $\{x_1, ..., x_n\}$ in this semi-directed cycle implies $\Omega_G(x_1) \subseteq \Omega_G(x_n)$, which contradicts the previous statement. Therefore, there cannot be any semi-directed cycles in the sink graph of any undirected graph.

□

Lemma S.B.2. Given two undirected graphs, G and G_1, with $G_1 \subseteq G$, the reduced $G-$mirrored sink graph of G_1 is acyclic.

Proof. If there is a cycle $\{x_1, ..., x_n, x_1\}$ in the reduced $G-$mirrored sink graph of G_1, then, since both the reduced sink graph of G and the reduced sink graph of G_1 are acyclic, there must exist a semi-directed cycle in the sink graph of G_1. By Lemma S.B.1. this is not possible.

□

Theorem 1.1. Two undirected graphs, G and G_1, with G and G_1 both in the Σ class and G_1 a subgraph of G, are generated by DAGs, D and D_1, with D_1 a subgraph of D, if and only if any DAG in the acyclic completion of the reduced $G-$mirrored sink graph of G_1 generates G_1.

Proof. First we prove the *only if* part of this theorem. We do this in two steps. We show for any DAG, D_1, in the acyclic completion of the reduced $G-$mirrored sink graph of G_1, the marginal dependency graph of D_1 is a subgraph of G_1, $\Sigma(D_1) \subseteq G_1$. We then show if there exists two DAGs, D^* and D_1^*, $D_1^* \subseteq D^*$, generating G and G_1, for any DAG, D_1, in the acyclic completion of the reduced $G-$mirrored sink graph of G_1, G_1 is a subgraph of the marginal dependency graph of D_1, $G_1 \subseteq \Sigma(D_1)$.

In proving the first step, we consult Lemma S.A.1. First, note any DAG, D_1, in the acyclic completion of the reduced $G-$mirrored sink graph of G_1 is a subgraph of some DG in the completion of the reduced sink graph of G_1: $a \to b \in D_1$ only if

the edge connecting a and b in the reduced sink graph of G_1 is either undirected or is oriented as $a \to b$. By Lemma S.A.1, for any DG in the completion of the reduced sink graph of G_1, its marginal dependency graph is equal to G_1. Now, since D_1 is a subgraph of some such DG, we conclude the marginal dependency graph of D_1 is a subgraph of G_1.

Now, we wish two show if there exists two DAGs, D^* and D_1^*, $D_1^* \subseteq D^*$, generating G and G_1, then, for any DAG, D_1, in the acyclic completion the reduced $G-$mirrored sink graph of G_1, G_1 is a subgraph of the marginal dependency graph of D_1— in other words, we wish to prove any D_1 in the acyclic completion of the reduced $G-$mirrored sink graph of G_1 implies *all* the edges in G_1 whenever there exists two DAGs satisfying the conditions of the dependency loss problem. If a and b are connected in G_1, they remain connected in all such DAGs unless the edge $a - b$ becomes doubly oriented in the $G-$mirrored sink graph of G_1. An edge, $a - b$, in G_1 becomes doubly oriented in the $G-$mirrored sink graph of G_1 under 3 conditions: (1) the edge $a - b$ is doubly oriented in the sink graph of G_1, (2) the edge $a - b$ is oriented in opposite directions in the sink graphs of G_1 and G, (3) the edge $a - b$ is doubly oriented in the sink graph of G .

1. The edge $a - b$ is doubly oriented in the sink graph of G_1:

Since (1) D_1^* generates G_1, and (2) the edge connecting a and b is doubly oriented in the sink graph of G_1, there cannot be a directed path between a and b in D_1^*. If a was an ancestor of b in D_1^*, then the boundary of a in $G_1 = \Sigma(D_1^*)$ would be a subset of or equal to the boundary of b in G_1. However, if that was the case the edge between a and b in the sink graph of G_1 would not have been doubly oriented. Therefore, a and b must have a common ancestor, z ($z \neq a$ and $z \neq b$), in D_1^*. As G_1 is the marginal dependency graph of D_1^*, z must be connected to both a and b in G_1. Furthermore, the edges $z - a$ and $z - b$ must be oriented as $z \to a$ and $z \to b$ in the

sink graph of G_1. Since z is an ancestor of a and b in D_1^*, we have $\Omega_{G_1}(z) \subseteq \Omega_{G_1}(a)$ and $\Omega_{G_1}(z) \subseteq \Omega_{G_1}(b)$. As the edge between a and b is doubly oriented in the sink graph of G_1, then there must exist a c such that $c \in \Omega_{G_1}(a) \wedge c \notin \Omega_{G_1}(b)$. Then we conclude that $\Omega_{G_1}(z)$ must be a subset of the $\Omega_{G_1}(a)$. Therefore, the edge $z - a$ has to be oriented as $z \to a$ in the sink graph of G_1. Likewise, the edge $z - b$ must be oriented as $z \to b$ in the sink graph of G_1. Now, since D_1^* is a subgraph of D^*, z is also a common ancestor of a and b in D^*. Therefore, z must also be connected to a and b in G. Finally, the connections $z - a$ (and $z - b$) are either undirected or oriented as $z \to b$ in the sink graph of G: z is an ancestor of b in D^* and the boundary of z in $G = \Sigma(D^*)$ must be a subset of or equal to the boundary of a. Finally, the orientations of the edges $z - a$ and $z - b$ in the sink graphs of G and G_1 imply that the edges $z - a$ and $z - b$ are oriented as $a \leftarrow z \to b$ in the $G-$mirrored sink graph of G_1. This proves that a and b are directly connected in the marginal dependency graph of any DAG in the acyclic completion of the reduced $G-$mirrored sink graph of G_1.

2. The edge $a - b$ is oriented as $a \to b$ in the sink graph of G_1 and $a \leftarrow b$ in the sink graph of G:

The orientation $a \leftarrow b$ in the sink graph of G implies there is no directed path from a to b in D^*. Given that D_1^* is a subgraph of D^*, there cannot be any directed paths from a to b in D_1^* either. Therefore, and since $a - b$ is oriented as $a \to b$ in the sink graph of G_1, there are no directed paths between a and b, whether from a to b or from b to a, in D_1^*. As a and b are connected in the G_1, then they must have had a common ancestor, z ($z \neq a$ and $z \neq b$), in D_1^*. Since G_1 is the marginal dependency graph of D_1^*, z must be directly connected to both a and b in G_1. As G_1 is a subgraph of G, z must also connect to both a and b in G. Now, the orientation $a \leftarrow b$ in the sink graph of G_1 implies there exists a node, c, in the boundary of a in G_1 that is not in the boundary of b. This node c cannot be in the boundary of z in

G_1 either: otherwise, since z is an ancestor of b in D_1^*, b would have been connected to c in G_1 also. Now, since z is an ancestor of a in D_1^* and given that $c \in \Omega_{G_1}(a)$ and c is not in the boundary of z in G_1, then $\Omega_{G_1}(z) \subset \Omega_{G_1}(a)$, and the edge $z - a$ must be oriented as $z \to a$ in the sink graph of G_1. The same edge must be oriented as $z - a$ or $z \to a$ in the sink graph of G: z is an ancestor of a in both D_1^* and D^*, and therefore $\Omega_{\Sigma(D^*)}(z)$ must be a subset of or equal to $\Omega_{\Sigma(D^*)}(a)$. Then, the edge $z - a$ must be oriented as $z \to a$ in the $G-$mirrored sink graph of G_1. Likewise, the edge $z - b$ must be also oriented as $z \to b$ in the $G-$mirrored sink graph of G_1. Therefore, $a - z - b$ is oriented as $a \leftarrow z \to b$ in the $G-$mirrored sink graph of G_1. This implies a and b are directly connected in the marginal dependency graph of any DAG in the acyclic completion of the reduced $G-$mirrored sink graph of G_1.

3. The edge $a - b$ is doubly oriented in the sink graph of G:

Since D^* generates G, and the edge $a - b$ is doubly oriented in the sink graph of G, there cannot be any directed paths from a to b or b to a in D^*. Now, since a and b are directly connected in G_1 and D_1^* is a subgraph of D^*, we conclude that a and b must have had a common ancestor, z ($z \neq a$ and $z \neq b$), in D_1^*. As D_1^* is a subgraph of D^*, z is also a common ancestor of a and b in D^* and is directly connected to both of them in G. Now, the edges $z - a$ and $z - b$ must be oriented as $z \to a$ and $z \to b$ in the sink graph of G. Since z is an ancestor of a and b in D^*, we have $\Omega_G(z) \subseteq \Omega_G(a)$ and $\Omega_G(z) \subseteq \Omega_G(b)$. As the edge between a and b is doubly oriented in the sink graph of G, then there must exist a node c such that $c \in \Omega_G(a) \wedge c \notin \Omega_G(b)$. Then we conclude that $\Omega_G(z)$ must be a subset of the $\Omega_G(a)$ as c cannot be in the boundary of z in G, since otherwise, the boundary of z in G would no longer be a subset of the boundary of b. Therefore, the edge $z - a$ has to be oriented as $z \to a$ in the sink graph of G. Furthermore, the same edges ,$z - a$ and $z - b$, are either undirected or oriented as $z \to a(b)$ in the sink graph of G_1 since z is an ancestor of both a and b in

D_1^*. Therefore, $a - z - b$ is oriented as $a \leftarrow z \rightarrow b$ in the $G-$mirrored sink graph of G_1. This implies a and b are directly connected in the marginal dependency graph of any DAG in the acyclic completion of the reduced $G-$mirrored sink graph of G_1.

Part 2: if

We show if there exists a DAG, D_1, in the completion of the reduced $G-$mirrored sink graph of G_1 that generates G_1, then there exist a DAG D, with $D_1 \subseteq D$, generating G. We construct D, using the sink graph of G as the starting point, as follows.

First, we form a PDAG by removing the doubly oriented edges in the sink graph of G and copying the orientations of the edges in D_1 to it: for any edge $a - b$ in the sink graph of G, we orient this edge as $a \rightarrow b$, if a and b are connected with the orientation $a \rightarrow b$ in D_1. No edge is reversed during this process since D_1 is a DAG in the acyclic completion of the reduced $G-$mirrored sink graph of G_1; $a \rightarrow b$ is in D_1 only if a and b are connected in the reduced $G-$mirrored sink graph of G_1 and the edge $a - b$ is undirected or is oriented as $a \rightarrow b$ in both the sink graph of G_1 and the sink graph of G. This constructions gives rise to a partially directed acyclic graph: since both D_1 and the sink graph of G (after removing its doubly oriented edges) are acyclic, a cycle in this construction would imply a semi-directed cycle in the sink graph of G, which is impossible (see Lemma S.B.1). We finish constructing D by orientating all the remaining undirected edges in any arbitrary manner so that the resulting graph is acyclic. D is a DAG in the acyclic completion of the sink graph of G and its marginal dependency graph is equal to G by Lemma 1.1; by construction, we also have $D_1 \subseteq D$. □

B.2

Theorem 1.2. Two undirected graphs, G and G_1, with G and G_1 both in the Σ class and G_1 a subgraph of G, are generated by DAGs, D and D_1, with $D_1 \subseteq D$, if

and only if for every external clique of G_1, there exists at least one node external in G_1, ξ, that is also G_1-outer external in G: $\forall x \in \Omega_{G_1}(\xi) : \Omega_G(\xi) \subseteq \Omega_G(x)$.

Proof. We first prove the *only if* part of this theorem. Assume two DAGs D^* and D_1^*, with $D_1^* \subseteq D^*$, generate the given undirected graphs: $\Sigma(D^*) = G$, $\Sigma(D_1^*) = G_1$. We show that the roots nodes (nodes who don't have any parents) of D_1^* are external in G_1 and G_1-outer external in G_1.

First we show any such root node, ξ_i, is external in G_1: $\Omega_{G_1}(\xi_i) \subseteq \Omega_{G_1}(x)$ for all x in the boundary of ξ_i in G_1. Any node x in the boundary of ξ_i in G_1 must by definition have a common ancestor with ξ_i in D_1^*, and since ξ_i is a root node in D_1^*, we conclude that any such x must be a descendant of ξ_i in D_1^*. Therefore, ξ_i is an external node in G_1, and together with the collection of all its descendants, it forms an external clique in G_1. On the other hand, ξ_i's are G_1-outer external in G. Since D_1^* is a subgraph of D^*, any x in the boundary of ξ_i in G_1, which by aforementioned argument is a descendant of ξ_i in D_1^*, remains a descendant of ξ_i in D^*. Therefore, as G is the marginal dependency graph of D^*, then $\Omega_G(\xi_i) \subseteq \Omega_G(x)$ for any x in the boundary of ξ_i in G_1, and the ξ_i's G_1-outer external in G.

To prove the if part of the theorem, we construct two DAGs D and D_1, with $D_1 \subseteq D$, that generate the given marginal dependency relations. We first construct D_1. We do so by selecting *one* external node, ξ_i, per external clique of G_1, that is also G_1-outer external in G. Then, for every ξ_i in this selection, we connect ξ_i to all nodes, x, in its boundary with the following orientation: $\xi_i \to x$. The resulting directed graph by construction is acyclic and generates G_1. This completes the construction of D_1.

We construct D, using the sink graph of G as the starting point, as follows. First, we form a PDAG by (1) removing the doubly oriented edges of the sink graph of

G, and (2) copying the orientations of the edges in D_1 (as constructed in the previous step) to the edges in the sink graph of G: for any edge, $a - b$, in the sink graph of G, we orient this edge as $a \to b$ if a and b are connected in D_1 and the edge $a - b$ is oriented as $a \to b$ in D_1. No edge is reversed during this process since we had assumed that ξ_i's are a set of nodes G_1-outer external in G: for any edge $\xi_i \to x$ in D_1, ξ_i and x must be connected in G and the edge $\xi_i - x$ must either be undirected in the sink graph of G or it must be oriented as $\xi_i \to x$. This constructions gives rise to a partially directed acyclic graph: since both D_1 and the reduced sink graph of G are acyclic, a cycle in this construction would imply a semi-directed cycle in the reduced sink graph of G, which is impossible (see Lemma S.B.1). We finish the construction of D by orientating all the remaining undirected edges in any arbitrary manner so that the resulting graph is acyclic. D is a DAG in the acyclic completion of the reduced sink graph of G and its marginal dependency graph is equal to G by Lemma 1.1; by construction, we also have $D_1 \subseteq D$. □

B.3

Theorem 1.3. Two undirected graphs, G and G_1, with G and G_1 both in the Σ class and $G_1 \subseteq G$, are generated by DGs, D and D, with $D_1 \subseteq D$ only if there exists two DAGs, D^* and D_1^*, with $D_1^* \subseteq D^*$, generating the target marginal dependency relations.

Proof. To prove this result we use the Lemma S.B.3 below. By Lemma S.B.3. there exists two DGs D and D_1, $D_1 \subseteq D$, generating G and G_1, only if *any* DG in the completion of the reduced $G-$mirrored sink graph of G_1 generates G_1. This lets us conclude that all the DAGs in the completion of the reduced $G-$mirrored sink graph of G_1 (if any) generate G_1 also. Now, by Lemma S.B.2. we know the reduced

$G-$mirrored sink graph of G_1 is acyclic so that there exists at least one DAG in its completion. Therefore, Lemma S.B.3 below allows us to conclude whenever there exists a pair of DGs satisfying the conditions of the dependency loss problem and generating G and G_1, then there also always exists a pair of DAGs satisfying the conditions of the dependency loss problem and generating G and G_1. □

Lemma S.B.3. Two undirected graphs, G and G_1, with G and G_1 both in the Σ class and G_1 a subgraph of G, are generated by DGs, D and D_1, with D_1 a subgraph of D, only if any DG in the acyclic completion of the reduced $G-$mirrored sink graph of G_1 generates G_1.

Proof. First, we show for any DG, D_1, in the completion of the reduced $G-$mirrored sink graph of G_1, the marginal dependency graph of D_1 is a subgraph of G_1, $\Sigma(D_1) \subseteq G_1$. We then show if there exists two DGs, (D^*, D_1^*), $D_1^* \subseteq D^*$, generating G and G_1, then for any DG, D_1, in the completion of the reduced $G-$mirrored sink graph of G_1, G_1 is a subgraph of the marginal dependency graph of D_1, $G_1 \subseteq \Sigma(D_1)$.

In proving the first step, we consult Lemma S.A.1. First, note any DG, D_1, in the completion of the reduced $G-$mirrored sink graph of G_1 is a subgraph of some DG in the completion of the reduced sink graph of G_1: $a \to b \in D_1$ only if the edge connecting a and b in the sink graph of G_1 is either undirected or is oriented as $a \to b$. By Lemma S.A.1, for any DG in the completion of the sink graph of G_1, its marginal dependency graph is equal to G_1. Now, since D_1 is a subgraph of some such DG, we conclude the marginal dependency graph of D_1 is a subgraph of G_1.

Now, we wish two show if there exists two DGs, D^* and D_1^*, $D_1^* \subseteq D^*$, generating G and G_1, then, for any DG, D_1, in the completion of the reduced $G-$mirrored sink graph of G_1, G_1 is a subgraph of the marginal dependency graph of D_1— in other words, we wish to prove any D_1 in the completion of the reduced $G-$mirrored sink

graph of G_1 implies *all* the edges in G_1 whenever there exists two DGs satisfying the conditions of the dependency loss problem. If a and b are directly connected in G_1, they remain directly connected in all such DGs unless the edge $a - b$ becomes doubly oriented in the $G-$mirrored sink graph of G_1. An edge, $a - b$, in G_1 becomes doubly oriented in the $G-$mirrored sink graph of G_1 under 3 conditions: (1) the edge $a - b$ is doubly oriented in the sink graph of G_1, (2) the edge $a - b$ is oriented in opposite directions in the sink graphs of G_1 and G, (3) the edge $a - b$ is doubly oriented in the sink graph of G_1 .

1. The edge $a - b$ is doubly oriented in the sink graph of G_1:

Since (1) D_1^* generates G_1, and (2) the edge connecting a and b is doubly oriented in the sink graph of G_1, there cannot be a directed path between a and b in D_1^*. If a was an ancestor of b in D_1^*, then the boundary of a in $G_1 = \Sigma(D_1^*)$ would be a subset of or equal to the boundary of b in G_1. However, if that was the case the edge between a and b in the sink graph of G_1 would not have been doubly oriented. Therefore, a and b must have a common ancestor, z ($z \neq a$ and $z \neq b$), in D_1^*. As G_1 is the marginal dependency graph of D_1^*, z must be connected to both a and b in G_1. Furthermore, the edges $z - a$ and $z - b$ must be oriented as $z \to a$ and $z \to b$ in the sink graph of G_1. Since z is an ancestor of a and b in D_1^*, we have $\Omega_{G_1}(z) \subseteq \Omega_{G_1}(a)$ and $\Omega_{G_1}(z) \subseteq \Omega_{G_1}(b)$. As the edge between a and b is doubly oriented in the sink graph of G_1, then there must exist a node, c, such that $c \in \Omega_{G_1}(a) \wedge c \notin \Omega_{G_1}(b)$. Then we conclude that $\Omega_{G_1}(z)$ must be a subset of the boundary of a in G_1 and also the boundary of b in G_1. Therefore, the edge $z - a$ has to be oriented as $z \to a$ in the sink graph of G_1. Now, since D_1^* is a subgraph of D^*, z is also a common ancestor of a and b in D^*. Therefore, z must also be connected to a and b in G. Finally, the connections $z - a$ and $z - b$ are either undirected or oriented as $z \to b$ and $z \to a$ in the sink graph of G: z is an ancestor of b in D^* and the boundary of z in $G = \Sigma(D^*)$

must be a subset of or equal to the boundary of a. Finally, the orientations of the edges $z - a$ and $z - b$ in the sink graphs of G and G_1 imply that the edges $z - a$ and $z - b$ are oriented as $a \leftarrow z \rightarrow b$ in the $G-$mirrored sink graph of G_1. This proves that a and b are directly connected in the marginal dependency graph of any DAG in the acyclic completion of the reduced $G-$mirrored sink graph of G_1.

2. The edge $a - b$ is oriented as $a \rightarrow b$ in the sink graph of G_1 and $a \leftarrow b$ in the sink graph of G:

The orientation $a \leftarrow b$ in the sink graph of G implies there is no directed path from a to b in D^*. Given that D_1^* is a subgraph of D^*, there cannot be any directed paths from a to b in D_1^* either. Therefore, and since $a - b$ is oriented as $a \rightarrow b$ in the sink graph of G_1, there are no directed paths between a and b, whether from a to b or from b to a, in D_1^*. As a and b are connected in the G_1, then they must have had a common ancestor, z ($z \neq a$ and $z \neq b$), in D_1^*. Since G_1 is the marginal dependency graph of D_1^*, z must be directly connected to both a and b in G_1. As G_1 is a subgraph of G, z must also connect to both a and b in G. Now, the orientation $a \leftarrow b$ in the sink graph of G_1 implies there exists a node, c, in the boundary of a in G_1 that is not in the boundary of b. This node c cannot be in the boundary of z in G_1 either: otherwise, since z is an ancestor of b in D_1^*, b would have been connected to c in G_1 also. Now, since z is an ancestor of a in D_1^* and given that $c \in \Omega_{G_1}(a)$ and c is not in the boundary of z in G_1, then $\Omega_{G_1}(z) \subset \Omega_{G_1}(a)$, and the edge $z - a$ must be oriented as $z \rightarrow a$ in the sink graph of G_1. The same edge must be oriented as $z - a$ or $z \rightarrow a$ in the sink graph of G: z is an ancestor of a in both D_1^* and D^*, and therefore $\Omega_{\Sigma(D^*)}(z)$ must be a subset of or equal to $\Omega_{\Sigma(D^*)}(a)$. Then the edge $z - a$ must be oriented as $z \rightarrow a$ in the $G-$mirrored sink graph of G_1. Likewise, the edge $z - b$ must be oriented as $z \rightarrow b$ in the $G-$mirrored sink graph of G_1. Therefore, $a - z - b$ is oriented as $a \leftarrow z \rightarrow b$ in the $G-$mirrored sink graph of G_1. This implies a and b are

directly connected in the marginal dependency graph of any DG in the completion of the reduced $G-$mirrored sink graph of G_1.

3. The edge $a - b$ is doubly oriented in the sink graph of G:

Since D^* generates G, and the edge $a - b$ is doubly oriented in the sink graph of G, there cannot be any directed paths from a to b or b to a in D^*. Now, since a and b are directly connected in G_1 and D_1^* is a subgraph of D^*, we conclude that a and b must have had a common ancestor, z ($z \neq a$ and $z \neq b$), in D_1^*. As D_1^* is a subgraph of D^*, z is also a common ancestor of a and b in D^* and is directly connected to both of them in G. Now, the edges $z - a$ and $z - b$ must be oriented as $z \to a$ and $z \to b$ in the sink graph of G. Since z is an ancestor of a in D^*, we have $\Omega_G(z) \subseteq \Omega_G(a)$. As the edge between a and b is doubly oriented in the sink graph of G, then there must exist a c such that $c \in \Omega_G(a) \wedge c \notin \Omega_G(b)$. Then we conclude that $\Omega_G(z)$ must be a subset of the $\Omega_G(a)$. Therefore, the edge $z - a$ has to be oriented as $z \to a$ in the sink graph of G. Furthermore, the same edges, $z - a$ and $z - b$, are either undirected or oriented as $z \to a$ and $z \to b$ in the sink graph of G_1, since z is an ancestor of a in D_1^*, and the boundary of z in G_1 is a subset of or equal to the boundary of a and the boundary of b . Therefore, $a - z - b$ is oriented as $a \leftarrow z \to b$ in the $G-$mirrored sink graph of G_1. This implies a and b are directly connected in the marginal dependency graph of any DG in the completion of the reduced $G-$mirrored sink graph of G_1.

□

C.1

Theorem 2.1. Given two undirected graphs, G_1 and G_2, with G_1 and G_2 both in Σ, there exists two intersection matched DAGs generating the given marginal dependency graphs, if and only if any DAG in the acyclic completion of the reduced $\{G_1, G_2\}-$mirrored sink graph of $G_\cap$, $G_\cap = G_1 \cap G_2$, generates $G_\cap$.

Proof. We first prove the *only if* part of this theorem. Let's say the pair D_1 and D_2 are a pair of intersection matched DAGs generating G_1 and G_2: $\Sigma(D_1) = G_1$, $\Sigma(D_2) = G_2$, and $\Sigma(D_1 \cap D_2) = G_1 \cap G_2$. Let us refer to the intersection of these two DAGs with $D_\cap$. Now we show any DAG in the acyclic completion of the reduced $\{G_1, G_2\}-$mirrored sink graph of $G_\cap$ generates $G_\cap$.

First, note any DAG D in the acyclic completion of the reduced $\{G_1, G_2\}-$mirrored sink graph of $G_\cap$ is a subgraph of some DG in the completion of the reduced sink graph of $G_\cap$: $a \to b \in D$ only if the edge connecting a and b in the sink graph of $G_\cap$ is either undirected or is oriented as $a \to b$. By Lemma S.A.1, for any DG in the completion of the sink graph of $G_\cap$, its marginal dependency graph is equal to $G_\cap$. Now, since D is a subgraph of some such DG, we conclude the marginal dependency graph of D is also a subgraph of $G_\cap$.

Now, we wish two show if there exists two intersection matched DAGs, D_1^* and D_2^*, $\Sigma(D_1^* \cap D_2^*) = \Sigma(D_1^*) \cap \Sigma(D_2^*)$, generating G_1 and G_2, then, for any DAG, D, in the acyclic completion the reduced $\{G_1, G_2\}-$mirrored sink graph of $G_\cap$, the marginal dependency graph of D is a supergraph of $G_\cap$— in other words, we wish to prove any D in the acyclic completion of the reduced $\{G_1, G_2\}-$mirrored sink graph of $G_\cap$ implies *all* the edges in $G_\cap$ whenever there exists two DAGs satisfying the conditions of the theorem. If a and b are connected in $G_\cap$, they remain connected in all such DAGs unless the edge $a - b$ becomes doubly oriented in the $\{G_1, G_2\}-$mirrored sink graph of $G_\cap$. An edge, $a-b$, in $G_\cap$ becomes doubly oriented in the $\{G_1, G_2\}-$mirrored sink graph of $G_\cap$ under four conditions: (1) the edge $a - b$ is doubly oriented in the sink graph of $G_\cap$, (2) the edge $a - b$ is oriented in opposite directions in the sink graphs of $G_\cap$ and G_1(or G_2), (3) $a - b$ is doubly oriented in the sink graph of G_1 (or G_2), (4) the edge $a - b$ is oriented in opposite directions in the sink graphs of G_1 and G_2.

1. The edge $a - b$ is doubly oriented in the sink graph of $G_\cap$:

Since (1) $D^*_\cap = D^*_1 \cap D^*_2$ generates $G_\cap$, and (2) the edge connecting a and b is doubly oriented in the sink graph of $G_\cap$, there cannot be a directed path between a and b in $D^*_\cap$. If a was an ancestor of b in $D^*_\cap$, then the boundary of a in $G_\cap = \Sigma(D^*_\cap)$ would be a subset of or equal to the boundary of b in $G_\cap$. However, if that was the case, the edge between a and b in the sink graph of $G_\cap$ would not have been doubly oriented. Therefore, given that there is no directed paths between a and b in $D^*_\cap$, and given that a and b are marginally dependent since they are directly connected in $G_\cap$, then a and b must have a common ancestor, z ($z \neq a$ and $z \neq b$), in $D^*_\cap$. As $G_\cap$ is the marginal dependency graph of $D^*_\cap$, z must be connected to both a and b in $G_\cap$. Furthermore, the edges $z - a$ and $z - b$ must be oriented as $z \to a$ and $z \to b$ in the sink graph of $G_\cap$. Since z is an ancestor of a and b in $D^*_\cap$, we have $\Omega_{G_\cap}(z) \subseteq \Omega_{G_\cap}(a)$ and $\Omega_{G_\cap}(z) \subseteq \Omega_{G_\cap}(b)$. Next, since there exists a node, c, such that $c \in \Omega_{G_\cap}(a) \wedge c \notin \Omega_{G_\cap}(b)$, then $\Omega_{G_\cap}(z)$ must be a subset of the $\Omega_{G_\cap}(a)$. Therefore, the edge $z - a$ has to be oriented as $z \to a$ in the sink graph of $G_\cap$. Now, since $D^*_\cap$ is a subgraph of D^*_1 and D^*_2, z is also a common ancestor of a and b in D^*_1 and D^*_2. Therefore, z must also be connected to a and b in G_1 and G_2. Finally, the connections $z - a$ (and $z - b$) are either undirected or oriented as $z \to b$ in the sink graphs of G_1 and G_2: z is an ancestor of b in D^*_1 and D^*_2, thus the boundary of z in $G_1 = \Sigma(D^*_1)$ and $G_2 = \Sigma(D^*_1)$ must be a subset of or equal to that of the boundary of a. Finally, the orientations of the edges $z - a$ and $z - b$ in the sink graphs of $G_\cap$, G_1, and G_2 imply that the edges $z - a$ and $z - b$ are oriented as $a \leftarrow z \to b$ in the $\{G_1, G_2\}-$mirrored sink graph of $G_\cap$. This proves that a and b are directly connected in the marginal dependency graph of any DAG in the acyclic completion of the reduced $\{G_1, G_2\}-$mirrored sink graph of $G_\cap$.

2. The edge $a-b$ is oriented as $a \to b$ in the sink graph of G_1 (or G_2) and $a \leftarrow b$ in the sink graph of $G_\cap$:

The orientation $a \leftarrow b$ in the sink graph of G_1(or G_2) implies there is no directed path from a to b in D_1^* (or D_2^*). Given that $D_\cap^*$ is a subgraph of D_1^* (and D_2^*), there cannot be any directed paths from a to b in $D_\cap^*$ either. Therefore, and since the edge $a-b$ is oriented as $a \to b$ in the sink graph of $G_\cap$, there are no directed paths between a and b, whether from a to b or from b to a, in $D_\cap^*$. As a and b are connected in the $G_\cap$, then they must have had a common ancestor, z ($z \neq a$ and $z \neq b$), in $D_\cap^*$. Since $G_\cap$ is the marginal dependency graph of $D_\cap^*$, z must be directly connected to both a and b in $G_\cap$; as $G_\cap$ is a subgraph of G_1 and G_2, z must also connect to both a and b in G_1 and G_2. Now, the orientation $a \leftarrow b$ in the sink graph of $G_\cap$ implies there exists a node, c, in the boundary of a in $G_\cap$ that is not in the boundary of b. This node c cannot be in the boundary of z in $G_\cap$ either: otherwise, since z is an ancestor of b in $D_\cap^*$, b would have been connected to c in $G_\cap$ also. Then, since z is an ancestor of a in $D_\cap^*$ and given that $c \in \Omega_{G_\cap}(a)$ and c is not in the boundary of z in $G_\cap$, we have $\Omega_{G_\cap}(z) \subset \Omega_{G_\cap}(a)$, and the edge $z-a$ must be oriented as $z \to a$ in the sink graph of $G_\cap$. The same edge must be either undirected or oriented as $z \to a$ in the sink graphs of G_1 and G_2: z is an ancestor of a in both D_1^* and D_2^*, and therefore $\Omega_{\Sigma(D_1^*)}(z)$ must be a subset of or equal to $\Omega_{\Sigma(D_1^*)}(a)$ and $\Omega_{\Sigma(D_2^*)}(z) \subseteq \Omega_{\Sigma(D_2^*)}(a)$. Then the edge $z-a$ must be oriented as $z \to a$ in the $\{G_1, G_2\}-$mirrored sink graph of $G_\cap$. Likewise, the edge $z-b$ must be oriented as $z \to b$ in $\{G_1, G_2\}-$mirrored sink graph of $G_\cap$. Therefore, $a-z-b$ is oriented as $a \leftarrow z \to b$ in the $\{G_1, G_2\}-$mirrored sink graph of $G_\cap$. This implies a and b are directly connected in the marginal dependency graph of any DAG in the acyclic completion of the reduced $\{G_1, G_2\}-$mirrored sink graph of $G_\cap$.

3. The edge $a-b$ is doubly oriented in the sink graph of G_1 (or G_2):

Since D_1^* generates G_1, and the edge $a - b$ is doubly oriented in the sink graph of G_1, there cannot be any directed paths from a to b or b to a in D_1^*. Now, since a and b are directly connected in $G_\cap$, and $D_\cap^*$ is a subgraph of D_1^* generating $G_\cap$, we conclude that a and b must have had a common ancestor, z ($z \neq a$ and $z \neq b$), in $D_\cap^*$: there cannot be any directed paths between a and b in D_1^*, and since $D_\cap^*$ is a subgraph of D_1^*, there is no directed path between a and b in $D_\cap^*$ also. As $D_\cap^*$ is a subgraph of D_1^* and D_2^*, z must also be a common ancestor of a and b in both D_1^* and D_2^*, and must be directly connected to both of them in both G_1 and G_2. Now, given that the edge $a - b$ is doubly oriented in the sink graph of G_1, and z is a common ancestor of a and b in D_1^*, then the edges $z - a$ and $z - b$ must be oriented as $z \to a$ and $z \to b$ in the sink graph of G_1. Furthermore, the same edges ($z - a$ and $z - b$) are either undirected or oriented as $z \to a(b)$ in the sink graph of $G_\cap$ and the sink graph of G_2, since z is an ancestor of a in both $D_\cap^*$ and D_2^*. Therefore, $a - z - b$ is oriented as $a \leftarrow z \to b$ in the $\{G_1, G_2\}-$mirrored sink graph of $G_\cap$. This implies a and b are directly connected in the marginal dependency of graph any DAG in the acyclic completion of the reduced $\{G_1, G_2\}-$mirrored sink graph of $G_\cap$.

4. The edge $a - b$ is oriented as $a \to b$ in the sink graph of G_1 and $a \leftarrow b$ in the sink graph of G_2:

On the one hand, the orientation $a \to b$ in the sink graph of G_1 implies there is no directed path from b to a in D_1^*. Given that $D_\cap^*$ is a subgraph of D_1^*, there cannot be any directed paths from b to a in $D_\cap^*$ either. On the other hand, the orientation $a \leftarrow b$ in the sink graph of G_2 implies there is no directed path from a to b in D_2^* . Given that $D_\cap^*$ is a subgraph of D_2^*, there cannot be any directed paths from b to a in $D_\cap^*$ either. Thus, we conclude that there are no directed paths between a and b, whether from a to b or b to a, in $D_\cap^*$. As a and b are connected in the $G_\cap$, then they must have had a common ancestor, z ($z \neq a$ and $z \neq b$), in $D_\cap^*$. Since $G_\cap$ is the

marginal dependency graph of $D^*_\cap$, z must be directly connected to both a and b in $G_\cap$; as $G_\cap$ is a subgraph of G_1 and G_2, z must also connect to both a and b in G_1 and G_2. Now, the orientation $a \leftarrow b$ in the sink graph of G_2 implies there exists a node, c, in the boundary of a in G_2 that is not in the boundary of b. This node c cannot be in the boundary of z in G_2 either: otherwise, since z is an ancestor of b in D^*_2, b would have been connected to c in G_2 also. Then, since z is an ancestor of a in D^*_2 and given that $c \in \Omega_{G_2}(a)$ and c is not in the boundary of z in G_2, then $\Omega_{G_2}(z) \subset \Omega_{G_2}(a)$, and the edge $z - a$ must be oriented as $z \rightarrow a$ in the sink graph of G_2. The same edge must be either undirected or oriented as $z \rightarrow a$ in the sink graph of G_1 and $G_\cap$: z is an ancestor of a in both D^*_1 and $D^*_\cap$, and therefore $\Omega_{\Sigma(D^*_1)}(z)$ must be a subset of or equal to $\Omega_{\Sigma(D^*_1)}(a)$ and $\Omega_{\Sigma(D^*_\cap)}(z) \subseteq \Omega_{\Sigma(D^*_\cap)}(a)$. Therefore the edge $z - a$ must be oriented as $z \rightarrow a$ in the $\{G_1, G_2\}-$mirrored sink graph of $G_\cap$. By the same logic, the edge $z - b$ must oriented as $z \rightarrow b$ in the $\{G_1, G_2\}-$mirrored sink graph of $G_\cap$. Therefore, $a - z - b$ is oriented as $a \leftarrow z \rightarrow b$ in the $\{G_1, G_2\}-$mirrored sink graph of $G_\cap$. This implies a and b are directly connected in the marginal dependency of any DAG in the acyclic completion of the reduced $\{G_1, G_2\}-$mirrored sink graph of $G_\cap$.

Part 2: if We show if there exists a DAG, $D_\cap$, in the completion of the reduced $\{G_1, G_2\}-$mirrored sink graph of $G_\cap$ that generates $G_\cap$, then there exists two intersection matched DAGs, D_1 and D_2, with $D_\cap \subseteq D_1$ and $D_\cap \subseteq D_2$, generating G_1 and G_2. We construct D_1 and D_2, using the sink graphs of G_1 and G_2 as the starting point, as follows. For the sake of brevity we only discuss how to construct D_1, construction of D_2 follows similarly.

First, we form a PDAG by removing the doubly oriented edges in the sink graph of G_1 and copying the orientations of the edges in $D_\cap$ to it: for any edge $a - b$ in the sink graph of G_1, we orient this edge as $a \rightarrow b$ if the edge $a - b$ is oriented as $a \rightarrow b$ in $D_\cap$. No edge is reversed during this process since $D_\cap$ is a DAG in the acyclic

completion of the reduced $\{G_1, G_2\}-$mirrored sink graph of $G_\cap$; $a \to b$ is in $D_\cap$ only if a and b are connected in the reduced $\{G_1, G_2\}-$mirrored sink graph of $G_\cap$ and the edge $a - b$ is undirected or is oriented as $a \to b$ in the sink graphs of G_1, G_2, and $G_\cap$. This constructions gives rise to a partially directed acyclic graph: since both $D_\cap$ and the sink graph of G_1 (after removing its doubly oriented edges) are acyclic, a cycle in this construction would imply a semi-directed cycle in the sink graph of G_1, which is impossible (see Lemma S.B.1). We finish constructing D_1 by orientating all the remaining undirected edges in any arbitrary manner so that the resulting graph is acyclic. D_1 is a DAG in the acyclic completion of the sink graph of G_1 and its marginal dependency graph is equal to G_1 by Lemma 1.1; by construction, we also have $D_\cap \subseteq D_1$.

□

C.2

Theorem 2.2. Given two undirected graphs, G_1 and G_2, with G_1 and G_2 both in Σ, there exists two intersection matched DAGs generating the given marginal dependency graphs, if and only if for every external clique of $G_\cap$, $G_\cap = G_1 \cap G_2$, there exists at least one node that is (1) external in $G_\cap$, and (2) $G_\cap$-outer external in G_1 and G_2.

Proof. We first prove the *only if* part of this theorem. Assume two DAGs D_1^* and D_2^*, are two intersection matched DAGs generating G_1 and G_2: $\Sigma(D_1^*) = G_1$, $\Sigma(D_2^*) = G_2$, and $\Sigma(D_1^* \cap D_2^*) = G_1 \cap G_2$. We show that the roots nodes (nodes who don't have any parents) of $D_\cap^* = D_1^* \cap D_2^*$ are external in $G_\cap$ and $G_\cap$-outer external in G_1 and G_2.

First we show any such root node, ξ_i, is external in $G_\cap$: $\Omega_{G_\cap}(\xi_i) \subseteq \Omega_{G_\cap}(x)$ for all x in the boundary of ξ_i in $G_\cap$. Any node x in the boundary of ξ_i in $G_\cap$ must by

definition have a common ancestor with ξ_i in $D^*_\cap$, and since ξ_i is a root node in $D^*_\cap$, we conclude that any such x must be a descendant of ξ_i in $D^*_\cap$. Therefore, ξ_i is an external node in $G_\cap$, and together with the collection of all its descendants, it forms an external clique in $G_\cap$. On the other hand, ξ_i's are $G_\cap$-outer external in G_1 and G_2. Since $D^*_\cap$ is a subgraph of D^*_1 and D^*_2, any x in the boundary of ξ_i in $G_\cap$, which by aforementioned argument is a descendant of ξ_i in $D^*_\cap$, remains a descendant of ξ_i in D^*_1 and D^*_2. Therefore, as G_1 and G_2 are the marginal dependency graphs of D^*_1 and D^*_2, then $\Omega_{G_1}(\xi_i) \subseteq \Omega_{G_1}(x)$ and $\Omega_{G_2}(\xi_i) \subseteq \Omega_{G_2}(x)$, and the ξ_i's $G_\cap$-outer external in G_1 and G_2.

To prove the if part of the theorem, we construct three DAGs $D_\cap$, D_1, and D_2, with $D_\cap \subseteq D_1$ and $D_\cap \subseteq D_2$, such that $\Sigma(D_\cap) = G_\cap$, $\Sigma(D_1) = G_1$, and $\Sigma(D_2) = G_2$. We first construct $D_\cap$. We do so by selecting *one* external node, $\xi^\cap_i$, per external clique of $G_\cap$, that is also $G_\cap$-outer external in G_1 and G_2. Then, for every $\xi^\cap_i$ in this selection, we connect $\xi^\cap_i$ to all nodes, x, in its boundary with the following orientation: $\xi^\cap_i \to x$. The resulting directed graph by construction is acyclic and generates $G_\cap$. This completes the construction of $D_\cap$.

We next construct D_1 and D_2. For brevity, we only discuss how to construct D_1, construction of D_2 follows similarly. We construct D_1 on top of the the sink graph of G_1 as we describe next. First, we form a PDAG by (1) removing the doubly oriented edges of the sink graph of G_1, and (2) copying the orientations of the edges in $D_\cap$ (as constructed in the previous step) to the edges in the sink graph of G_1: for any edge, $a - b$, in the sink graph of G_1, we orient this edge as $a \to b$ if a and b are connected in $D_\cap$ and the edge $a - b$ is oriented as $a \to b$ in $D_\cap$. No edge is reversed during this process since we had assumed that $\xi^\cap_i$'s are a set of nodes that $G_\cap$-outer external in G_1. For any edge $\xi^\cap_i \to x$ in $D_\cap$, $\xi^\cap_i$ and x must be connected in G_1 and the edge $\xi^\cap_i - x$ must be either undirected in the sink graph of G_1 or it must be

oriented as $\xi_i^{\cap} \to x$ since $\xi_i^{\cap}$ is $G_{\cap}$-outer external in G_1, and $\Omega_{G_{\cap}}(\xi_i^{\cap}) \subseteq \Omega_{G_{\cap}}(\xi_i^{\cap})(x)$. This constructions gives rise to a partially directed acyclic graph: since both $D_{\cap}$ and the reduced sink graph of G_1 are acyclic, a cycle in this construction would imply a semi-directed cycle in the sink graph of G_1, which is impossible (see Lemma S.B.1). We finish the construction of D_1 by orientating all the remaining undirected edges in any arbitrary manner so that the resulting graph is acyclic. D_1 is a DAG in the acyclic completion of the reduced sink graph of G_1 and its marginal dependency graph is equal to G by Lemma 1.1; by construction, we also have $D_{\cap} \subseteq D_1$. □

C.3

Theorem 2.3. Given two undirected graphs, G_1 and G_2, with G_1 and G_2 both in Σ, there exists two intersection matched DGs generating the given marginal dependency graphs, only if there exists two intersection matched DAGs generating G_1 and G_2.

Proof. To prove this result we use the Lemmas S.C.1 and S.C.2 below. By Lemma S.C.1 there exists two intersection matched DGs generating the given marginal dependency graphs, G_1 and G_2, with $G_{\cap} = G_1 \cap G_2$, only if *any* DG in the completion of the reduced $\{G_1, G_2\}-$mirrored sink graph of $G_{\cap}$ generates $G_{\cap}$. This lets us conclude that all the *DAGs* (if any) in the completion of the reduced $\{G_1, G_2\}-$mirrored sink graph of $G_{\cap}$ generate $G_{\cap}$ also. Now, by Lemma S.C.2. we know the reduced $\{G_1, G_2\}-$mirrored sink graph of $G_{\cap}$ is acyclic so that there exists at least one DAG in its completion. Therefore, Lemma S.C.1 together with Theorem 2.1., allow us to conclude whenever there exists a pair of intersection matched DGs generating G_1 and G_2, then there also always exists a pair of intersection matched DAGs generating G_1 and G_2. □

Lemma S.C.1. Given two undirected graphs, G_1 and G_2, with G_1 and G_2 both in Σ, there exists two intersection matched DGs generating the given marginal dependency graphs, only if any DG in the completion of the $\{G_1, G_2\}-$mirrored sink graph of $G_\cap$, $G_\cap = G_1 \cap G_2$, generates $G_\cap$.

Proof. Let's say the pair D_1^* and D_2^* are pair of intersection matched DGs generating G_1 and G_2: $\Sigma(D_1^*) = G_1$, $\Sigma(D_2^*) = G_2$, and $\Sigma(D_1^* \cap D_2^*) = G_1 \cap G_2$. Let us refer to the intersection of these two DGs with $D_\cap^*$. Now we show any DG in the completion of the reduced $\{G_1, G_2\}-$mirrored sink graph of $G_\cap$ generates $G_\cap$.

First, note any DG $D_\cap$ in the completion of the reduced $\{G_1, G_2\}-$mirrored sink graph of $G_\cap$ is a subgraph of some DG in the completion of the reduced sink graph of $G_\cap$: $a \to b \in D_\cap$ only if the edge connecting a and b in the sink graph of $G_\cap$ is either undirected or is oriented as $a \to b$. By Lemma S.A.1, for any DG in the completion of the sink graph of $G_\cap$, its marginal dependency graph is equal to G_1. Now, since $_\cap$ is a subgraph of some such DG, we conclude the marginal dependency graph of $D_\cap$ is also a subgraph of $G_\cap$.

Now, we wish two show if there exists two intersection matched DGs, D_1^* and D_2^*, $\Sigma(D_1^* \cap D_2^*) = \Sigma(D_1^*) \cap \Sigma(D_2^*)$, generating G_1 and G_2, then, for any DG, $D_\cap$, in the completion of the reduced $\{G_1, G_2\}-$mirrored sink graph of $G_\cap$, the marginal dependency graph of $D_\cap$ is a supergraph of $G_\cap$— in other words, we wish to prove any $D_\cap$ in the completion of the reduced $\{G_1, G_2\}-$mirrored sink graph of $G_\cap$ implies *all* the edges in $G_\cap$ whenever there exists two DGs satisfying the conditions of the theorem. If a and b are connected in $G_\cap$, they remain connected in all such DGs unless the edge $a - b$ becomes doubly oriented in the $\{G_1, G_2\}-$mirrored sink graph of $G_\cap$. An edge, $a - b$, in $G_\cap$ becomes doubly oriented in the $\{G_1, G_2\}-$mirrored sink graph of $G_\cap$ under four conditions: (1) the edge $a - b$ is doubly oriented in the sink

graph of $G_\cap$, (2) the edge $a - b$ is oriented in opposite directions in the sink graphs of $G_\cap$ and G_1(or G_2), (3) $a - b$ is doubly oriented in the sink graph of G_1 (or G_2), (4) the edge $a - b$ is oriented in opposite directions in the sink graphs of G_1 and G_2.

1. The edge $a - b$ is doubly oriented in the sink graph of $G_\cap$:

Since (1) $D^*_\cap = D^*_1 \cap D^*_2$ generates $G_\cap$, and (2) the edge connecting a and b is doubly oriented in the sink graph of $G_\cap$, there cannot be a directed path between a and b in $D^*_\cap$. If a was an ancestor of b in $D^*_\cap$, then the boundary of a in $G_\cap = \Sigma(D^*_\cap)$ would be a subset of or equal to the boundary of b in $G_\cap$. However, if that was the case the edge between a and b in the sink graph of $G_\cap$ would not have been doubly oriented. Therefore, given that there is no directed paths between a and b in $D^*_\cap$ and given that a and b are marginally dependent since they are directly connected in $G_\cap$, then a and b must have a common ancestor, z ($z \neq a$ and $z \neq b$), in $D^*_\cap$. As $G_\cap$ is the marginal dependency graph of $D^*_\cap$, z must be connected to both a and b in $G_\cap$. Furthermore, the edges $z - a$ and $z - b$ must be oriented as $z \to a$ and $z \to b$ in the sink graph of $G_\cap$. Since z is an ancestor of a and b in $D^*_\cap$, we have $\Omega_{G_\cap}(z) \subseteq \Omega_{G_\cap}(a)$ and $\Omega_{G_\cap}(z) \subseteq \Omega_{G_\cap}(b)$. Given that there exists a c such that $c \in \Omega_{G_\cap}(a) \wedge c \notin \Omega_{G_\cap}(b)$, then $\Omega_{G_\cap}(z)$ must be a subset of the $\Omega_{G_\cap}(a)$. Therefore, the edge $z - a$ has to be oriented as $z \to a$ in the sink graph of $G_\cap$. Now, since $D^*_\cap$ is a subgraph of D^*_1 and D^*_2, z is also a common ancestor of a and b in D^*_1 and D^*_2. Therefore, z must also be connected to a and b in $G_1 = \Sigma(D^*_1)$ and $G_2 = \Sigma(D^*_2)$. Finally, the edges $z - a$ and $z - b$ are either undirected or oriented as $z \to b$ and $z \to a$ in the sink graphs of G_1 and G_2: z is an ancestor of b in D^*_1 and D^*_2, and the boundary of z in $\Sigma(D^*_1)$ ($\Sigma(D^*_1)$) must be a subset of or equal to the boundary of a. Finally, the orientations of the edges $z - a$ and $z - b$ in the sink graphs of $G_\cap$, G_1, and G_2 imply that the edges $z - a$ and $z - b$ are oriented as $a \leftarrow z \to b$ in the $\{G_1, G_2\}-$mirrored sink graph of $G_\cap$.

This proves that a and b are directly connected in the marginal dependency graph of any DG in the completion of the reduced $\{G_1, G_2\}-$mirrored sink graph of $G_\cap$.

2. The edge $a - b$ is oriented as $a \to b$ in the sink graph of G_1 and $a \leftarrow b$ in the sink graph of $G_\cap$:

The orientation $a \leftarrow b$ in the sink graph of G_1 implies there is no directed path from a to b in D_1^*. Given that $D_\cap^*$ is a subgraph of D_1^*, there cannot be any directed paths from a to b in $D_\cap^*$ either. Therefore, and since the edge $a - b$ is oriented as $a \to b$ in the sink graph of $G_\cap$, there are no directed paths between a and b, whether from a to b or from b to a, in $D_\cap^*$. As a and b are connected in the $G_\cap = \Sigma(D_\cap^*)$, then the nodes a and b must have had a common ancestor, z ($z \neq a$ and $z \neq b$), in $D_\cap^*$. Since $G_\cap$ is the marginal dependency graph of $D_\cap^*$, z must be directly connected to both a and b in $G_\cap$; as $G_\cap$ is a subgraph of G_1 and G_2, z must also connect to both a and b in G_1 and G_2. Now, the orientation $a \leftarrow b$ in the sink graph of $G_\cap$ implies there exists a node, c, in the boundary of a in $G_\cap$ that is not in the boundary of b. This node c cannot be in the boundary of z in $G_\cap$ either: otherwise, since z is an ancestor of b in $D_\cap^*$, b would have been connected to c in $G_\cap$ also. Then, since z is an ancestor of a in $D_\cap^*$, and given that $c \in \Omega_{G_\cap}(a)$ and c is not in the boundary of z in $G_\cap$, then $\Omega_{G_\cap}(z) \subset \Omega_{G_\cap}(a)$, and the edge $z - a$ must be oriented as $z \to a$ in the sink graph of $G_\cap$. The same edge must be either undirected or oriented as $z \to a$ in the sink graphs of G_1 and G_2: z is an ancestor of a in both D_1^* and D_2^*, and therefore $\Omega_{\Sigma(D_1^*)}(z)$ must be a subset of or equal to $\Omega_{\Sigma(D_1^*)}(a)$ and $\Omega_{\Sigma(D_2^*)}(z) \subseteq \Omega_{\Sigma(D_2^*)}(a)$. Then the edge $z - a$ must be oriented as $z \to a$ in the $\{G_1, G_2\}-$mirrored sink graph of $G_\cap$. Likewise, the edge $z - b$ must be oriented as $z \to b$ in the $\{G_1, G_2\}-$mirrored sink graph of $G_\cap$. Therefore, $a - z - b$ is oriented as $a \leftarrow z \to b$ in the $\{G_1, G_2\}-$mirrored sink graph of $G_\cap$. This implies a and b are directly connected in the marginal dependency graph of any DG in the completion of the reduced $\{G_1, G_2\}-$mirrored sink graph of $G_\cap$.

3. The edge $a - b$ is doubly oriented in the sink graph of G_1:

Since D_1^* generates G_1, and the edge $a - b$ is doubly oriented in the sink graph of G_1, there cannot be any directed paths from a to b or b to a in D_1^*. Now, since a and b are directly connected in $G_\cap$, and $D_\cap^*$ is a subgraph of D_1^* generating $G_\cap$, we conclude that a and b must have had a common ancestor, z ($z \neq a$ and $z \neq b$), in $D_\cap^*$: there cannot be any directed paths between a and b in D_1^*, and since $D_\cap^*$ is a subgraph of D_1^*, there is no directed path between a and b in $D_\cap^*$ also. As $D_\cap^*$ is a subgraph of D_1^* and D_2^*, z must also be a common ancestor of a and b in both D_1^* and D_2^*, and must be directly connected to both of them in both G_1 and G_2. Now, the edges $z - a$ must be oriented as $z \to a$ in the sink graph of G_1. Since z is an ancestor of a and b in D_1^*, we have $\Omega_{G_1}(z) \subseteq \Omega_{G_1}(a)$ and $\Omega_{G_1}(z) \subseteq \Omega_{G_1}(b)$. Next, since there exists a c such that $c \in \Omega_{G_1}(a) \wedge c \notin \Omega_{G_1}(b)$, then $\Omega_{G_1}(z)$ must be a subset of the $\Omega_{G_1}(a)$ and the edge $z - a$ has to be oriented as $z \to a$ in the sink graph of G_1. Likewise, the edge $z - b$ has to be oriented as $z \to b$ in the sink graph of G_1. Furthermore, the same edges, $z - a$ and $z - b$, are either undirected or oriented as $z \to a$ and $z \to b$ in the sink graph of $G_\cap$ and the sink graph of G_2 since z is an ancestor of a in $D_\cap^*$ and D_2^*. Therefore, $a - z - b$ is oriented as $a \leftarrow z \to b$ in the $\{G_1, G_2\}-$mirrored sink graph of $G_\cap$. This implies a and b are directly connected in the marginal dependency graph of any DG in the completion of the reduced $\{G_1, G_2\}-$mirrored sink graph of $G_\cap$.

4. The edge $a - b$ is oriented as $a \to b$ in the sink graph of G_1 and $a \leftarrow b$ in the sink graph of G_2:

On the one hand, the orientation $a \to b$ in the sink graph of G_1 implies there is no directed path from b to a in D_1^*. Given that $D_\cap^*$ is a subgraph of D_1^*, there cannot be any directed paths from b to a in $D_\cap^*$ either. On the other hand, the orientation $a \leftarrow b$ in the sink graph of G_2 implies there is no directed path from a to b in D_2^* . Given that $D_\cap^*$ is a subgraph of D_2^*, there cannot be any directed paths from b to

a in $D^*_\cap$ either. Thus, we conclude that there are no directed paths between a and b, whether from a to b or b to a, in $D^*_\cap$. As a and b are connected in the $G_\cap$, then they must have had a common ancestor, z ($z \neq a$ and $z \neq b$), in $D^*_\cap$. Since $G_\cap$ is the marginal dependency graph of $D^*_\cap$, z must be directly connected to both a and b in $G_\cap$. As $G_\cap$ is a subgraph of G_1 and G_2, z must also connect to both a and b in G_1 and G_2. Now, the orientation $a \leftarrow b$ in the sink graph of G_2 implies there exists a node, c, in the boundary of a in G_2 that is not in the boundary of b. This node c cannot be in the boundary of z in G_2 either: otherwise, since z is an ancestor of b in D^*_2, b would have been connected to c in G_2 also. Then, since z is an ancestor of a in D^*_2 and given that $c \in \Omega_{G_2}(a)$ and c is not in the boundary of z in G_2, we have $\Omega_{G_2}(z) \subset \Omega_{G_2}(a)$, and the edge $z - a$ must be oriented as $z \rightarrow a$ in the sink graph of G_2. The same edge must be either undirected or oriented as $z \rightarrow a$ in the sink graph of G_1 and $G_\cap$: z is an ancestor of a in both D^*_1 and $D^*_\cap$. Therefore, the edge $z - a$ must be oriented as $z \rightarrow a$ in the $\{G_1, G_2\}-$mirrored sink graph of $G_\cap$. By the same logic, the edge $z - b$ must oriented as $z \rightarrow b$ in the $\{G_1, G_2\}-$mirrored sink graph of $G_\cap$. Therefore, $a - z - b$ is oriented as $a \leftarrow z \rightarrow b$ in the $\{G_1, G_2\}-$mirrored sink graph of $G_\cap$. This implies a and b are directly connected in the marginal dependency graph of any DG in the completion of the reduced $\{G_1, G_2\}-$mirrored sink graph of $G_\cap$. □

Lemma S.C.2. Given any two undirected graphs G_1 and G_2, with $G_\cap = G_1 \cap G_2$, the reduced $\{G_1, G_2\}-$mirrored sink graph of $G_\cap$ is acyclic.

Proof. Lets assume there is a cycle, $C = \{x_1, x_2, ..., x_m, x_1\}$, in the reduced $\{G_1, G_2\}-$mirrored sink graph of $G_\cap$. We immediately conclude that every two consecutive nodes in the collection $\{x_1, x_2, ..., x_m, x_1\}$ must be connected by an edge in G_1, G_2, and $G_\cap$. Consider the edge $x_1 \rightarrow x_2$ in this cycle. Since the edge connecting x_1 and x_2 is

oriented as $x_1 \to x_2$ in the reduced $\{G_1, G_2\}-$mirrored sink graph of $G_\cap$, we conclude that the edge connecting these same two nodes must be oriented as $x_1 \to x_2$ in either the reduced sink graph of G_1, the reduced sink graph of G_2, or the reduced sink graph of $G_\cap$. Let's say that the edge $x_1 - x_2$ is oriented as $x_1 \to x_2$ in the reduced sink graph of G_1. Every other edge $x_i - x_{i+1}$ must either be undirected or must be oriented as $x_i \to x_{i+1}$ in the reduced sink graph of G_1. Thus the reduced sink graph of G_1 must have had a semi-directed cycle. However, according to Lemma S.B.1, the reduced sink graph of any UG, G, cannot contain a semi-directed cycle. □

C.4

Theorem 2.4. If there exists two intersection matched DAGs generating undirected graphs G_1 and G_2, then there exists two weakly acyclic DAGs generating G_1 and G_2.

Proof. Let's say D_1 and D_2 are two intersection matched DAGs generating G_1 and G_2. We will us refer to the intersection of these two DAGs with $D_\cap$.

Consider the root nodes of D_1: these nodes are also root nodes of $D_\cap$ and must be external in G_1 and $G_\cap$. Similarly the root nodes of D_2 must be external in G_2 and $G_\cap$, and they must be root nodes of $D_\cap$ also. From the given intersection matched DAGs we construct two DAGs, R_1 and R_2, such that R_1 and R_2 are weakly acyclic. We construct R_1 by directly connecting every root node in D_1, ξ_i^1, to every node, x, that is its descendant in D_1: $\forall x \in \Omega_{G_1}(\xi_i) : \xi_i^1 \to x$. We construct R_2 similarly and by connecting every root node of D_2 to all their descendants. We now show R_1 and R_2 are weakly acyclic.

Let's assume the contrary and say that there exists two nodes x and y, such that x is an ancestor of y in R_1, and y is an ancestor of x in R_2. Given the construction of

R_1 and R_2, then x must have been a root node of D_1 and y must have been a root node of D_2. Since D_1 and D_2 are intersection matched, and since x and y must be connected in both G_1 and G_2 (otherwise $x \to y$ would not have been in R_1 and $x \leftarrow y$ would not have been in R_2), we conclude that x and y must have had a common ancestor, z, in $D_\cap$. Since x is a root node of D_1, then the only possible common ancestor of x and y in $D_\cap$ would be x itself. Thus x is an ancestor of y in $D_\cap$. However, this is not possible since y is a root node of $D_\cap$. Therefore, R_1 and R_2 must be weakly acyclic.

□

D.1

Theorem 3.1 Given two marginal dependency graphs G_1 and G_2, there exists two ancestrally faithful, marginally faithful, intersection matched DAGs generating G_1 and G_2 if and only if the following three conditions hold:

(1) For every external clique of G_1 , there exists at least one node that is (1.1) external in G_1, (1.2) G_1-outer external in $G = G_1 \cup G_2$, (1.3) external in $G_\cap = G_1 \cap G_2$, (1.4) $G_\cap$-outer external in G_2, and (1.5) ***weakly*** transitive in $\{G_1, G_2, G_\cap\}-$mirrored sink graph of G, the merged sink graph of G_1 and G_2.

(2) For every external clique of G_2 , there exists at least one node that is (2.1) external in G_2, (2.2) G_2-outer external in G , (2.3) external in $G_\cap$, (2.4) $G_\cap$-outer external in G_1, and (2.5) ***weakly*** transitive in the merged sink graph of G_1 and G_2.

(3) For every external clique of $G_\cap$, there exists at least one node that is (3.1) external in $G_\cap$, (3.2) $G_\cap$-outer external in G_1, G_2, and G, and (3.3) ***weakly*** transitive in the merged sink graph of G_1 and G_2.

Proof. We use the following notation in our proof. P_1 and P_2 are used to denote the sink graphs of G_1 and G_2. We use G to represent the union of the two marginal dependency graphs, $G = G_1 \cup G_2$, and P to denote its sink graph. We use $G_\cap$ for the intersection of the two marginal dependency graphs, $G_\cap = G_1 \cap G_2$, and $P_\cap$ to denote its sink graph. We use P_1^* and P_2^* for the $G-$mirrored sink graphs of G_1 and G_2, $P_\cap^*$ to denote $\{G, G_1, G_2\}-$mirrored sink graph of $G_\cap$, and finally P^{**} for the merged sink graph of G_1 and G_2.

Only If

Lets say the two DAGs D_1 and D_2 are ancestrally faithful, marginally faithful, intersection matched DAGs generating G_1 and G_2. Let us represent the union and intersection of these two DAGs, respectively, by D and $D_\cap$.

First we show that the root nodes of D_1 and D_2 satisfy conditions (1.1)-(1.5) and (2.1)-(2.5).

(1.1, 2.1) Root nodes of D_1 are external nodes of G_1. Consider a root node of D_1, ϕ_i^1. Any node v connected to ϕ_i^1 in G_1 must also be connected to every other node in the boundary of ϕ_i^1 in G_1. If a node is connected to ϕ_i^1 in G_1, then this node must be a descendant of ϕ_i^1 in D_1, and therefore, it must be connected to every other descendant of ϕ_i^1 in the marginal dependency graph of D_1 as ϕ_i^1 is a common ancestor of all of its descendants. Furthermore, every edge in G_1 covered by an external clique containing one of these root nodes. Consider an edge $x - y \in G_1$. Since $\Sigma(D_1) = G_1$, then there must exist a z such that $z \geq x \wedge z \geq y$ in D_1. Now, either z is a root node, or it is a descendant of a root node π_z. Then the edge $x - y$ is contained in the external clique of the root node π_z in G_1.

(1.2, 2.2) The root nodes of D_1 are G_1-outer external in G and the root nodes of G_2 are G_2-outer external in G. Consider a root node of D_1, ϕ_i^1. Since D_1 is a subgraph of D, then all the nodes in $\Omega_{G_1}(\phi_i^1)$ remain as descendants of ϕ_i^1 in D:

$\forall v \in \Omega_{G_1}(\phi_i^1) : \phi_i^1 \geq v \in D$. Therefore, for all v in $\Omega_{G_1}(\phi_i^1)$, the boundary of ϕ_i^1 in G is a subset of or equal to the boundary of v in G, and ϕ_i^1 is G_1-outer external in G.

(1.3, 2.3) These roots also remain external in $G_\cap = G_1 \cap G_2$ since they are also the root nodes of $D_\cap$ as $D_\cap$ is a subgraph of D_1 and D_2.

(1.4, 2.4) Finally, since they remain as roots node in $D_\cap$, and since $D_\cap$ is a subgraph of D_1 and D_2, then these root nodes are $G_\cap$-outer external in G_1 and G_2.

We now show that any such root, ϕ, is weakly transitive in P^{**}. That is, (i) whenever there is a directed path from ϕ to node y through node x in P^{**}, i.e. $\phi \to x \to y$ or $\phi - x \to y$ or $\phi \to x - y$, then ϕ and y are also connected inP^{**} and the edge $\phi - y$ is either undirected in P^{**} or is oriented as $\phi \to y$ in P^{**}, and (ii) whenever there is an undirected path from ϕ to node y through node x, i.e. $\phi - x - y$, then ϕ and y are also connected in P^{**} and the edge $\phi - y$ is either undirected in P^{**} or is oriented as $\phi \to y$. We consider each of these four cases, namely $\phi \to x \to y$, $\phi - x \to y$, $\phi \to x - y$, and $\phi - x - y$ separately.

Part 1

Case 1. *Lets say the edges $\phi - x$ and $x - y$ are oriented as $\phi - x - y$ in P^{**}. We show that the nodes ϕ and y must be connected in $P^{*}*$ and the edge $\phi - y$ is either undirected in P^{**} or is oriented as $\phi \to y$.*

Here we only utilize the fact that ϕ is a node external in $G_\cap$ and $G_\cap$-outer external in G_1, G_2, and G. Let us then emphasize here that this case of weak transitivity holds for any pair of undirected graphs G_1 and G_2, and for any node ϕ who is a node external in $G_\cap$ and $G_\cap$-outer external in G_1, G_2, and $G = G_1 \cup G_2$.

First we note that the nodes ϕ and y must be connected in G. Otherwise, the triplet $\phi - x - y$ forms a v-structure in G, forcing the edge $x - y$ to be oriented as $x \leftarrow y$ in P, the sink graph of G. If the edge $x - y$ was oriented as $x \leftarrow y$ in P, however, the edge $x - y$ would not have been left undirected in the merged sink graph

of G_1 and G_2, P^{**}. Therefore, the nodes ϕ and y must be connected in G, and for that reason they must also be connected in G_1 or G_2.

(I) Now lets assume that the nodes ϕ and x are only connected in G_1. Since the edge $\phi - x$ is undirected in P^{**}, we conclude that $\phi - x$ is undirected in P_1^*.

(I.I) Now, if x and y are also connected in G_1, then the edge $x - y$ must also be undirected in P_1^* since this edge is undirected in P^{**}. Since both edges $x - y$ and $\phi - x$ are undirected in P_1^*, then ϕ and y must have also been connected in P_1^* through an undirected edge given that P_1^* is strongly transitive. Now, if ϕ and y are disconnected in G_2, then we conclude that the edge $\phi - y$ is undirected in P^{**} as well, since this edge only resides in G_1 and is undirected in P_1^*. On the other hand, if ϕ and y are also connected in G_2, then the edge $\phi - y$ must be either undirected or oriented as $\phi \to y$ in the sink graphs of G, G_2, and $G_\cap$: ϕ is a node external in $G_\cap$ and $G_\cap$-outer external in G_1, G_2, and G, and y is a node in the boundary of ϕ in $G_\cap$. Then the edge $\phi - y$ is either undirected in P_{**} or is oriented as $\phi \to y$.

(I.II) If x and y are not connected in G_1, then ϕ and y cannot be connected in G_1 either, since otherwise the triplet $y - \phi - x$ would have formed a v-structure in G_1 and the edge $\phi - x$ would have been oriented as $\phi \leftarrow x$ in P_1. Then the edge $\phi - x$ would not have been undirected in P^{**}. Therefore, ϕ and y are disconnected in G_1 and must be connected only in G_2. On the other hand, if x and y are not connected in G_1, then x and y must be connected in G_2 and the edge $x - y$ must be undirected in P_2^*. However, since the triplet $x - y - \phi$ forms a v-structure in G_2 (we had assumed that ϕ and x are only connected in G_1), the edge $x - y$ cannot be undirected in P_2^*.

(II) Now lets assume that the nodes ϕ and x are connected both in G_1 and G_2. Under this condition, $\phi - x$ can only be undirected in P_1^* and P_2^*.

(II.I) Now, if x and y are *only* connected in G_1, then the edge $x - y$ must be undirected in P_1^* since this edge is undirected in P^{**}. Furthermore, since x and y are

assumed to be disconnected in G_2, and the edge $\phi - x$ is undirected in P^{**}, then the nodes ϕ and y must be disconnected in G_2; otherwise, the v-structure $y - \phi - x$ would have forced the edge $\phi - x$ to take the orientation $\phi \leftarrow x$ in P_2 and this edge would not have been oriented as undirected in P^{**}. Therefore, ϕ and y must be connected in G_1 and only in G_1. Now, since P_1^* is strongly transitive and we have the chain $\phi - x - y$ in P_1^*, then the edge $\phi - y$ must be undirected in P_1^*. Since, the edge $\phi - y$ only resides in G_1, we conclude that ϕ and y must be connected through an undirected edge, $\phi - y$, in P^{**}.

(II.II) Finally, lets assume that x and y are also connected in both G_1 and G_2. We immediately note that ϕ and y must also be connected in both G_1 and G_2, otherwise, the triplet $\phi - x - y$ would form a v-structure in G_1 or G_2, preventing the formation of an undirected edge between x and y in P^{**}. This lets us conclude that x, y, and ϕ form a connected component in $G_\cap$. Now, since the edges $x - y$ and $\phi - x$ are undirected in P^{**}, then these two edges must also be undirected in $P_\cap^*$, the $\{G, G_1, G_2\}$-mirrored sink graph of $G_\cap$. Finally, since $P_\cap^*$ is strongly transitive, then $\phi - y$ must also be undirected in $P_\cap^*$, and therefore, it must also be undirected in P^{**}.

Case 2. *Lets say the edges $\phi - x$ and $x - y$ are oriented as $\phi \rightarrow x - y$ in P^{**}. We show that the edge $\phi - y$ must be in P^{**} and it must be either undirected or oriented as $\phi \rightarrow y$ in P^{**}.*

Here we only utilize the fact that ϕ is a node external in $G_\cap$ and $G_\cap$-outer external in G_1, G_2, and G. Let us then emphasize here that this case of weak transitivity holds for any pair of undirected graphs G_1 and G_2, and for any node ϕ who is a node external in $G_\cap$ and $G_\cap$-outer external in G_1, G_2, and $G = G_1 \cup G_2$.

First we note that the nodes ϕ and y must be connected in G. Otherwise, the triplet $\phi - x - y$ would have formed a v-structure in G, forcing the edge $x - y$ to be oriented as $x \leftarrow y$ in P, the sink graph of G. If this edge was oriented as $x \leftarrow y$ in

P, however, it would not be undirected in the merged sink graph of G_1 and G_2, P^{**}. Therefore, the nodes ϕ and y must be connected in G, and for that reason they must be also connected in G_1 or G_2.

(I) Now lets assume that the nodes ϕ and x are **only** connected in G_1. Since the edge $\phi - x$ is oriented as $\phi \to x$ in P^{**}, we conclude that the edge $\phi - x$ must have been oriented as $\phi \to x$ in P_1^*.

(I.I) Now, if the nodes x and y are also connected in G_1, then the edge $x - y$ must be undirected in P_1^* since this edge is undirected in P^{**}. Given that P_1^* is strongly transitive, then the nodes ϕ and y must be connected in P_1^* and the edge $\phi - y$ must be oriented as $\phi \to y$ in P_1^*. Now, if ϕ and y are disconnected in G_2, we can then immediately conclude that the edge $\phi - y$ will be oriented as $\phi \to y$ in P^{**}. On the other hand, if ϕ and y are connected in G_2 also, then the edge $\phi - y$ must either be undirected in P_2^* or it must be oriented as $\phi \to y$. To see this, note that if ϕ and y are connected in G_2 also, then y must be a node in the boundary of ϕ in $G_\cap$, and given that ϕ is a node $G_\cap$-outer external in G_2 and G, then the boundary of ϕ in G_2 and G must a subset of or equal to the boundary of y. Furthermore, as ϕ external in $G_\cap$, then the boundary of ϕ in $G_\cap$ is also a subset of or equal to the boundary of y. Therefore, the edge $\phi - y$ must be either undirected in $P_\cap$ or it must be oriented as $\phi \to y$ in $P_\cap$. Given the possible orientations of the edge $\phi - y$ in P_1^*, P_2^*, and $P - \cap$, we conclude that the edge $\phi - y$ must be oriented as $\phi \to y$ in P^{**}.

(I.II) If the nodes x and y are not connected in G_1, then the nodes ϕ and y cannot be connected in G_1 either. Otherwise, the triplet $y - \phi - x$ would have formed a v-structure in G_1 and the edge $\phi - x$ would have been oriented as $\phi \leftarrow x$ in P_1. Then, the edge $\phi - x$ would not have been oriented as $\phi \to x$ in P^{**}. Therefore, ϕ and y cannot be connected in G_1 and must be connected only in G_2. On the other hand, if x and y are not connected in G_1, then x and y must be connected in G_2 and

the edge $x - y$ must be undirected in P_2^*. However, since the triplet $x - y - \phi$ forms a v-structure in G_2 (we had assumed that ϕ and x are only connected in G_1), the edge $x - y$ cannot not be undirected in P_2^*.

(II) Now lets assume that ϕ and x are connected both in G_1 and G_2. Under this condition, given that the edge $\phi - x$ is oriented as $\phi \to x$ in P^{**}, the edge $\phi - x$ can only be either undirected or oriented as $\phi \to x$ in P_1^* and P_2^*.

(II.I) Now, if x and y are **only** connected in G_1, then the edge $x - y$ must be undirected in P_1^* since this edge is undirected in P^{**}. Furthermore, since x and y are disconnected in G_2, and the edge $\phi - x$ is oriented as $\phi \to x$ in P^{**}, then the nodes ϕ and y must be disconnected in G_2; otherwise, the v-structure $y - \phi - x$ would force the edge $\phi - x$ to take the orientation $\phi \leftarrow x$ in P_2 and would prevent the orientation $\phi \to x$ from forming in P^{**}. Therefore, ϕ and y must be connected in G_1 and only in G_1. Now, since P_1^* is strongly transitive and we either have the chain $\phi - x - y$ or $\phi \to x - y$ in P_1^*, then the edge $\phi - y$ must be oriented either as $\phi \to y$ or it must be left undirected in P_1^*. Since, the edge $\phi - y$ only resides in G_1, we conclude that ϕ and y must be connected either through an undirected edge, $\phi - y$, or with the orientation $\phi \to y$ in P^{**}.

(II.II) Finally, lets assume that the nodes x and y are also connected in both G_1 and G_2. We immediately note that ϕ and y must also be connected in both G_1 and G_2, otherwise, the triplet $\phi - x - y$ would form a v-structure in G_1 or G_2, preventing the edge $x - y$ to be left undirected in P^{**}. This lets us conclude that x, y, and ϕ form a connected component in $G_\cap$. Now, since the edges $x - y$ and ϕx are oriented as $\phi \to x - y$ in P^{**}, then they must be oriented with same exact orientation, i.e. $x - y$ and $\phi \to x$, in $P_\cap^*$, the $\{G, G_1, G_2\}$-mirrored sink graph of $G_\cap$. Finally, since $P_\cap^*$ is strongly transitive, then ϕ and y must be connected in $P_\cap^*$ and the edge $\phi - x$ must

be oriented as $\phi \to y$ in $P^*_\cap$, and hence, the nodes ϕ and y must be connected with the same orientation, $\phi \to y$, in P^{**}.

Case 3. *Lets say the edges $\phi - x$ and $x - y$ are oriented as $\phi \to x \to y$ in P^{**}. We show that the edge $\phi - y$ must be in P^{**} and it must be oriented as $\phi \to y$ in P^{**}.*

Here we show that if G_1 and G_2 are generated by two ancestrally faithful, marginally faithful, intersection matched DAGs or *DGs*, D_1 and D_2, then the nodes in D_1 whose boundary in G_1 and $G_\cap$ comprises of only their descendants in D_1 and $D_\cap$ satisfy this case of weak transitivity in P^{**}. The same applies to the nodes in D_2 whose boundary in G_2 and $G_\cap$ comprises of only their descendants in D_1 and $D_\cap$.

First we note that the nodes ϕ and y must be connected in G. Otherwise, the triplet $\phi - x - y$ forms a v-structure in G, forcing the edge $x - y$ to be oriented as $x \leftarrow y$ in P, the sink graph of G. If $x \leftarrow y$ was in P, however, the edge $x - y$ would not have been oriented as $x \to y$ in the merged sink graph of G_1 and G_2, P^{**}. Therefore, the nodes ϕ and y must be connected in G, and for that reason they must also be connected in either G_1 or G_2.

(I) Now lets assume that the nodes ϕ and x are **only** connected in G_1. Since the edge $\phi - x$ is oriented as $\phi \to x$ in P^{**}, we conclude that the edge $\phi - x$ must be oriented as $\phi \to x$ in P^*_1.

(I.I) Now, if x and y are also connected in G_1, then the edge $x - y$ must either be undirected in P^*_1 or it must be oriented as $x \to y$ in P^*_1 since it is oriented as $x \to y$ in P^{**}. Then, since P^*_1 is strongly transitive, and given the chain $\phi \to x - y$ in P^*_1, we conclude that the edge $\phi - y$ must be oriented as $\phi \to y$ in P^*_1. Now, if ϕ and y are disconnected in G_2, we can then immediately conclude that the edge $\phi - y$ must be oriented as $\phi \to y$ in P^{**}. On the other hand, if ϕ and y are connected in G_2, then ϕ and y are also connected in $G_\cap$, and we conclude that y must be a descendant of ϕ in $D_\cap$, and on that account, y must also be a descendant of ϕ in D_1 and D_2. This

implies that the the boundary of the node ϕ is a subset of or equal to that of the boundary of the node y in graphs G_1, G_2, $G_\cap$, and G. Then, the edge $\phi - y$ must either be undirected or it must be oriented as $\phi \to y$ in the sink graphs of G, G_1, G_2, and $G_\cap$. Given that we already established that the edge $\phi - y$ must be oriented as $\phi \to y$ in P_1^*, we conclude that the edge $\phi - y$ must be oriented as $\phi \to y$ in P^{**}.

(I.II) If the nodes x and y are not connected in G_1, then the nodes ϕ and y cannot be connected in G_1 either. Otherwise, the triplet $y - \phi - x$ would have formed a v-structure in G_1 and the edge $\phi - x$ would have been oriented as $\phi \leftarrow x$ in P_1. Then the edge $\phi - x$ would not have not been oriented as $\phi \to x$ in P^{**}. Therefore, ϕ and y cannot be connected in G_1 and must be connected only in G_2. Furthermore, if x and y are not connected in G_1, then x and y must be connected in G_2 and the edge $x - y$ must be oriented as $x \to y$ in P_2^*. Then, the nodes x, y, and ϕ form a v-structure $x - y - \phi$ in G_2, and the edge $y - \phi$ can only be either doubly oriented, $y \leftrightarrow \phi$, in P_2^* or it must be oriented as $y \leftarrow \phi$ in P_2^*. If $y - \phi$ is oriented as $y \leftarrow \phi$ in P_2^*, since y and ϕ are disconnected in G_1, we can immediately conclude that the edge $y - \phi$ is also oriented as $y \leftarrow \phi$ in P^{**}. We now show that $y - \phi$ cannot be doubly oriented in P_2^* under the conditions we have considered so far.

First we note that the edge $y - \phi$ must be undirected in P or it must be oriented as $y \leftarrow \phi$ (since P is strongly transitive, and $\phi \to x \to y$ is in P^{**}). Now, If $y - \phi$ is doubly oriented in P_2^*, then it must have been either doubly oriented in P_2 or it must have been oriented as $y \to \phi$. Since we had assumed that ϕ and x were disconnected in G_2, however, the edge between ϕ and y cannot be oriented as $\phi \leftarrow y$ in the sink graph of G_2: x is in the boundary of y in G_2 while it is not in the boundary of ϕ and $\Omega_{G_2}(y) \not\subseteq \Omega_{G_2}(\phi)$. Then, the only possibility is for the edge $\phi - y$ to be doubly oriented in P_2. Thus, ϕ has the node y in its boundary in G_2 who is not its descendant in D_2, and therefore ϕ must have been a node whose boundary in G_1 and $G_\cap$ comprises

of only its descendants in D_1 and $D_\cap$. Furthermore, since the edge $y - \phi$ is doubly oriented in P_2, we conclude that ϕ must have a node, ϕ_2, as its ancestor in D_2 where ϕ_2 is not connected to y in G_2. On the other hand, since ϕ and x are connected in G_1, and the boundary of ϕ in G_1 is comprised of only its descendants in D_1, we conclude ϕ must be an ancestor of x in D_1. Now, since ϕ is an ancestor of x in D_1, and ϕ_2 is an ancestor of ϕ in D_2, and that D_1 and D_2 are ancestrally faithful, then ϕ_2 must be an ancestor of x in either D_1 or D_2. However, ϕ_2 cannot be an ancestor of x in D_2, since that would imply ϕ_2 and y were connected in G_2. The boundary of x is a subset of or equal to that of y in G_2 according to the orientation of the edge $x - y$ in P_2^*, and if ϕ_2 were connected to x in G_2, then ϕ_2 and y must have been connected in G_2 as well. Therefore, ϕ_2 must be an ancestor of x in D_1. Finally, we note that under the conditions specified ϕ_2 and y must be connected in G_1. Otherwise, since ϕ_2 and y are not connected in G_2, then $\phi_2 - x - y$ would have formed a v-structure in G, and the edge $x - y$ would have not been oriented as $x \to y$ in P^{**}. Now, since ϕ_2 is an ancestor of x in D_1, and ϕ_2 and y are connected in G_1, we conclude that x and y must also be connected in G_1, which is against the conditions we have specified.

(II) Now lets assume that the nodes ϕ and x are connected in **both** G_1 and G_2.

(II.I) Now, if x and y are **only** connected in G_1, then the edge $x - y$ must be oriented as $x \to y$ in P_1^* since it is oriented as $x \to y$ in P^{**}. Therefore, since P_1^* is strongly transitive, and we have either the chain $\phi \to x \to y$ or the chain $\phi - x \to y$ in P_1^*, then the edge $\phi - y$ must be oriented as $\phi \to y$ in P_1^*. Now, if ϕ and y are disconnected in G_2, we can then immediately conclude that the edge $\phi - y$ must be oriented as $\phi \to y$ in P^{**} also. In fact, ϕ and y cannot be connected in G_2. Otherwise, the triplet $x - \phi - y$ would have formed a v-structure in G_2, and the edge $x - \phi$ would have been oriented as $x \to \phi$ in P_2, and this edge would not have been oriented as $\phi \to x$ in P^{**}.

(II.II) Finally, lets assume that the nodes x and y are also connected in **both** G_1 and G_2. We immediately note that ϕ and y must also be connected in both G_1 and G_2. Otherwise, the triplet $\phi - x - y$ would have formed a v-structure in G_1 or G_2, preventing the formation of $x \to y$ in P^{**}. This lets us conclude that x, y, and ϕ form a connected component in $G_\cap$. Now, since the edges $\phi - x$ and $x - y$ are oriented as $\phi \to x \to y$ in P^{**}, then the same edges must be oriented similarly, $x \to y$ and $\phi \to x$, in $P^*_\cap$, the $\{G, G_1, G_2\}$-mirrored sink graph of $G_\cap$. Finally, since $P^*_\cap$ is strongly transitive, then the edge $\phi - y$ must be oriented as $\phi \to y$ in $P^*_\cap$, and therefore, the edge $\phi - y$, must be oriented as $\phi \to y$ in P^{**}.

Case 4. *Lets say the edges $\phi - x$ and $x - y$ are oriented as $\phi - x \to y$ in P^{**}. We show that the nodes ϕ and y must be connected in P^{**} and the edge $\phi - y$ must either be undirected in P^{**} or it must be oriented as $\phi \to y$.*

Here we show that if G_1 and G_2 are generated by two ancestrally faithful, marginally faithful, intersection matched DAGs or *DGs*, D_1 and D_2, then the nodes in D_1 whose boundary in G_1 and $G_\cap$ comprises of only their descendants in D_1 and $D_\cap$ satisfy this case of weak transitivity in P^{**}. The same applies to the nodes in D_2 whose boundary in G_2 and $G_\cap$ comprises of only their descendants in D_1 and $D_\cap$.

First we note that the nodes ϕ and y must be connected in G. Otherwise, the triplet $\phi - x - y$ forms a v-structure in G, forcing the edge $x - y$ to be oriented as $x \leftarrow y$ in P, the sink graph of G. If the edge $x - y$ was oriented as $x \leftarrow y$ in P, however, this edge would not have been oriented as $x \to y$ in the merged sink graph of G_1 and G_2, P^{**}. Therefore, the nodes ϕ and y must be connected in G, and for that reason they must also be connected in G_1 or G_2.

(I) Now lets assume that nodes x and y are **only** connected in G_1. Since $x \to y$ is in P^{**}, we conclude that the edge $x - y$ must have been oriented as $x \to y$ in P^*_1.

(I.I) Now, if the nodes ϕ and x are also connected in G_1, then the edge $\phi - x$ must be undirected in P_1^* since this edge is undirected P^{**}. Since P_1^* is strongly transitive, then the edge $\phi - y$ must be oriented as $\phi \to y$ in P_1^*. Now, if ϕ and y are disconnected in G_2, we can then immediately conclude that the edge $\phi - y$ must be oriented as $\phi \to y$ in P^{**}. On the other hand, if ϕ and y are connected in G_2, then ϕ and y are also connected in $G_\cap$, and we conclude that y must be a descendant of ϕ in $D_\cap$, and on that account, y must also be a descendant of ϕ in D_1 and D_2. This implies that the the boundary of the node ϕ is a subset of or equal to that of the boundary of the node y in graphs G_1, G_2, $G_\cap$, and G. Then, the edge $\phi - y$ must either be undirected or it must be oriented as $\phi \to y$ in the sink graphs of G, G_1, G_2, and $G_\cap$. Given that we already established that ϕ and y must be connected in G_1, and the edge $\phi - y$ must be oriented as $\phi \to y$ in P_1^*, we conclude that the edge $\phi - y$ must be oriented as $\phi \to y$ in P^{**}.

(I.II) If ϕ and x are not connected in G_1, then, on the one hand, ϕ and x must be connected in G_2, and on the other hand, ϕ and y cannot be connected in G_2. Otherwise, since x and y were assumed to be disconnected in G_2, the triplet $y - \phi - x$ would have formed a v-structure in G_2 and the edge $\phi - x$ would have been oriented as $\phi \leftarrow x$ in P_2 . Then the edge $\phi - x$ would not have been undirected in P^{**}. Therefore, ϕ and y are disconnected in G_2 and must be connected only in G_1. Then x, y, and ϕ form a v-structure $x - y - \phi$ in G_1, and the edge $y - \phi$ can only be either doubly oriented $y \leftrightarrow \phi$ in P_1^* or it must oriented as $y \leftarrow \phi$ in P_1^*. If the edge $y - \phi$ is oriented as $y \leftarrow \phi$ in P_1^*, since y and ϕ are disconnected in G_1, we can immediately conclude that this edge must also oriented as $y \leftarrow \phi$ in P^{**}. We now show that $y - \phi$ cannot be doubly oriented in P_1^* under the conditions we have considered so far.

First we note that the edge $y - \phi$ must be undirected in P or it must be oriented as $y \leftarrow \phi$ (since P is strongly transitive, and $\phi \to x \to y$ is in P^{**}). Now, If $y - \phi$ is

doubly oriented in P_1^*, then it must have been either doubly oriented in P_1 or it must have been oriented as $y \to \phi$. Since we had assumed that ϕ and x were disconnected in G_1, however, the edge between ϕ and y cannot be oriented as $\phi \leftarrow y$ in the sink graph of G_2: x is in the boundary of y in G_1 while it is not in the boundary of ϕ and $\Omega_{G_1}(y) \not\subseteq \Omega_{G_1}(\phi)$. Then, the only possibility is for the edge $\phi - y$ to be doubly oriented in P_1. Thus, ϕ has node, y, in its boundary in G_1 who is not its descendant in D_1, therefore ϕ must have been a node whose boundary in G_2 and $G_\cap$ comprises of only its descendants in D_2 and $D_\cap$. Furthermore, since the edge $y - \phi$ is doubly oriented in P_1, we conclude that ϕ must have a node, ϕ_1, as its ancestor in D_1 where ϕ_1 is not connected to y in G_1. On the other hand, since ϕ and x are connected in G_2, and the boundary of ϕ in G_2 is comprised of only its descendants in D_2, we conclude ϕ must be an ancestor of x in D_2. Now, since ϕ is an ancestor of x in D_2, and ϕ_1 is an ancestor of ϕ in D_1, and that D_1 and D_2 are ancestrally faithful, then ϕ_1 must be an ancestor of x in either D_1 or D_2. However, ϕ_1 cannot be an ancestor of x in D_1, since that would imply ϕ_1 and y were connected in G_1. The boundary of x is a subset of or equal to that of y in G_1 according to the orientation of the edge $x - y$ in P_1^* and if ϕ_1 were connected to x in G_1, then ϕ_1 and y must have been connected in G_1 as well. Therefore, ϕ_1 must be an ancestor of x in D_2. Finally, we note that under the conditions specified ϕ_1 and y must be connected in G_2. Otherwise, since ϕ_1 and y are not connected in G_1, then $\phi_1 - x - y$ would form a v-structure in G, and the edge $x - y$ would not be oriented as $x \to y$ in P^{**}. Now, since ϕ_1 is an ancestor of x in D_2 and ϕ_1 and y are connected in G_2, we conclude that x and y must also be connected in G_2, which is against the conditions (condition **(I)**) we have specified.

(II) Now lets assume that x and y are connected both in G_1 and G_2.

(II.I) Now, if ϕ and x are **only** connected in G_1, then the edge $\phi - x$ must be undirected in P_1^* since this edge is undirected in P^{**}. On the other hand, since the

edge $x - y$ is oriented as $x \to y$ is in P^{**}, then this edge must either be undirected in P_1^* or it must be oriented as $x \to y$ in P_1^*. Given that both ϕ and x, and x and y are assumed to be connected in G_1, then ϕ and y must also be connected in G_1: otherwise, the triplet $\phi - x - y$ would form a v-structure in G_1, preventing the formation of $x \to y$ or $\phi - x$ in P^{**}. Now, since $\phi - x$ is undirected in P_1^*, and the edge $x - y$ must either be undirected in P_1^* or it must be oriented as $x \to y$ in P_1^*, and given that P_1^* is strongly transitive, then the edge $\phi - y$ either remains undirected in P_1^* or it must be oriented as $\phi \to y$. Now, if ϕ and y are disconnected in G_2, we can then immediately conclude that the edge $\phi - y$ either remains undirected in P^{**} or it must be oriented as $\phi \to y$ in P^{**}. On the other hand, if ϕ and y were connected in G_2, then ϕ and y must have also been connected in $G_\cap$. Since ϕ is a node whose boundary in $G_\cap$ comprises of only its descendants in $D_\cap$, then y must have been a descendant of ϕ in $D_\cap$, and the edge $\phi - y$ must have been either undirected or oriented as $\phi \to y$ in the sink graphs of G_1, G_2, G, and $G_\cap$. Then the edge $\phi - y$ must be either undirected or oriented as $\phi \to y$ in P^{**}.

(II.II) Finally, lets assume that x and y are also connected in both G_1 and G_2. We immediately note that the nodes ϕ and y must also be connected in both G_1 and G_2, otherwise, the triplet $\phi - x - y$ would form a v-structure in G_1 or G_2, and the edge $x - y$ would not be oriented as $x \to y$ in P^{**}. This lets us conclude that x, y, and ϕ form a connected component in $G_\cap$. Now, since the edges $x - y$ and $\phi - x$ are oriented as $\phi - x \to y$ in P^{**}, then these edges must be oriented similarly, $x \to y$ and $\phi - x$, in $P_\cap^*$, the $\{G_1, G_2, G\}$-mirrored sink graph of $G_\cap$. Finally, since $P_\cap^*$ is strongly transitive, then the edge $\phi - y$ must be oriented as $\phi \to y$ in $P_\cap^*$, and therefore, as $\phi \to y$ in P^{**}.

Part 2

Now consider the root nodes of $D_\cap$. These roots are external nodes of $G_\cap$, and for every external clique of $G_\cap$, there must exists a root node of $D_\cap$ that resides in this clique (2.1). Any such root node, $\phi_k^\cap$, is $G_\cap$-outer external in G_1, G_2, and G (G_2-outer external in G), since $\phi_k^\cap$ is an ancestor of all the nodes in $\Omega_{G_\cap}(\phi_k^\cap)$ in $D_\cap$, and remains an ancestor of all such nodes in D_1, D_2, and $D = D_1 \cup D_2$ also (2.2).

We now show that any such root, ϕ, is weakly transitive in P^{**}. That is, (i) whenever there is a directed path from ϕ to node y through node x in P^{**}, i.e. $\phi \to x \to y$ or $\phi - x \to y$ or $\phi \to x - y$, then ϕ and y are connected in P^{**} and the edge $\phi - y$ is either oriented as $\phi \to y$ or is undirected in P^{**}, and (ii) whenever there is an undirected path from ϕ to node y through node x, i.e. $\phi - x - y$, then ϕ and y are connected in P^{**} and the edge $\phi - y$ is undirected in P^{**} . We consider each of these four cases, namely $\phi \to x \to y$, $\phi - x \to y$, $\phi \to x - y$, and $\phi - x - y$ separately. Out of these four cases, we only need to reconsider the cases relating to the structures $\phi \to x \to y$ and $\phi - x \to y$.

Case 3*. Lets say the edges $\phi - x$ and $x - y$ are oriented as $\phi \to x \to y$ in P^{**}. We show that the edge $\phi - y$ must be in P^{**} and it must be oriented as $\phi \to y$ in P^{**}.*

Here we show that if G_1 and G_2 are generated by two ancestrally faithful, marginally faithful, intersection matched DAGs or *DGs*, D_1 and D_2, then the nodes in $D_\cap$ whose boundary in $G_\cap$ comprises of only their descendants in $D_\cap$ satisfy this case of weak transitive in P^{**}.

First we note that the nodes ϕ and y must be connected in G. Otherwise, the triplet $\phi - x - y$ would have formed a v-structure in G, forcing the edge $x - y$ to be oriented as $x \leftarrow y$ in P, the sink graph of G. If the edge $x - y$ was oriented as $x \leftarrow y$ in P, however, the edge $x - y$ would not have been oriented as $x \to y$ in the merged sink graph of G_1 and G_2, P^{**}. Therefore, the nodes ϕ and y must be connected in G, and for that reason they must also be connected in G_1 or G_2.

(I) Now lets assume that the nodes ϕ and x are **only** connected in G_1. Since the edge $\phi - x$ is oriented as $\phi \rightarrow x$ in P^{**}, we conclude that the edge $\phi - x$ must be oriented as $\phi \rightarrow x$ in P_1^*.

(I.I) Now, if x and y are also connected in G_1, then the edge $x - y$ must either be undirected in P_1^* or it must be oriented as $x \rightarrow y$ in P_1^* since it is oriented as $x \rightarrow y$ in P^{**}. Then, since P_1^* is strongly transitive, and given the chain $\phi \rightarrow x - y$ in P_1^*, we conclude that ϕ and y must also be connected in G_1 and the edge $\phi - y$ must be oriented as $\phi \rightarrow y$ in P_1^*. Now, if ϕ and y are disconnected in G_2, we can then immediately conclude that the edge $\phi - y$ must be oriented as $\phi \rightarrow y$ in P^{**}. On the other hand, if ϕ and y are connected in G_2, then y must have been a node in the boundary of ϕ in $G_\cap$, and given that ϕ is an ancestor of all the nodes in its boundary in $G_\cap$, ϕ must have been an ancestor of y in $D_\cap$, D_1 and D_2. Then the edge $\phi - y$ must have been either undirected or oriented as $\phi \rightarrow y$ in the sink graphs of $G_\cap$ and G_2. Given that we already established that the edge $\phi - y$ is oriented as $\phi \rightarrow y$ in P_1^*, we conclude that the edge $\phi - y$ must be oriented as $\phi \rightarrow y$ in P^{**}.

(I.II) If the nodes x and y are not connected in G_1, then the nodes ϕ and y cannot be connected in G_1 either. Otherwise, the triplet $y - \phi - x$ would have formed a v-structure in G_1 and the edge $\phi - x$ would have been oriented as $\phi \leftarrow x$ in P_1. Then, the edge $\phi - x$ would not have not been oriented as $\phi \rightarrow x$ in P^{**}. Therefore, ϕ and y cannot be connected in G_1 and must be connected in G_2 and only in G_2. Furthermore, if x and y are not connected in G_1, then x and y must be connected in G_2 and the edge $x - y$ must be oriented as $x \rightarrow y$ in P_2^*. Then, the nodes x, y, and ϕ form a v-structure $x - y - \phi$ in G_2, and the edge $y - \phi$ can only be either doubly oriented, $y \leftrightarrow \phi$, in P_2^* or it must be oriented as $y \leftarrow \phi$ in P_2^*. If $y - \phi$ is oriented as $y \leftarrow \phi$ in P_2^*, since y and ϕ are disconnected in G_1, we can immediately conclude that

the edge $y - \phi$ is also oriented as $y \leftarrow \phi$ in P^{**}. We now show that $y - \phi$ cannot be doubly oriented in P_2^* under the conditions we have considered so far.

First we note that the edge $y - \phi$ must be undirected in P or it must be oriented as $y \leftarrow \phi$ (since P is strongly transitive, and $\phi \rightarrow x \rightarrow y$ is in P^{**}). Now, If $y - \phi$ is doubly oriented in P_2^*, then it must have been either doubly oriented in P_2 or it must have been oriented as $y \rightarrow \phi$. Since we had assumed that ϕ and x were disconnected in G_2, however, the edge between ϕ and y cannot be oriented as $\phi \leftarrow y$ in the sink graph of G_2: x is in the boundary of y in G_2 while it is not in the boundary of ϕ and $\Omega_{G_2}(y) \not\subseteq \Omega_{G_2}(\phi)$. Then, the only possibility is for the edge $\phi - y$ to be doubly oriented in P_2. Now, since the edge $y - \phi$ is doubly oriented in P_2, we conclude that ϕ must have some ancestor in D_2, ϕ_2, that is a root or root* node of D_2 and whom y is not a descendant of: $\phi_2 \geq_{D_2} \phi \wedge \phi_2 \not\geq_{D_2} y$. Then ϕ must be connected to ϕ_2 in P_2^* with an edge oriented as $\phi_2 \rightarrow \phi$: ϕ_2 is an ancestor of ϕ in D_2 and there is a node, namely y, in the boundary of ϕ that is not in the boundary of ϕ_2. Now, given that ϕ is a node whose boundary in $G_\cap$ comprises of only its descendants in $D_\cap$, we conclude that ϕ_2 cannot be connected to ϕ in G_1. Otherwise, if ϕ_2 was connected to ϕ in G_1, then ϕ_2 would have been in the boundary of ϕ in $G_\cap$, and since any node in the boundary of ϕ in $G_\cap$ is by our assumption a descendant of ϕ in $D_\cap$, ϕ_2 would have been a descendant of ϕ in $D_\cap$, and on that account a descendant of ϕ in D_2 as well. However, this is impossible since we have already established that the node y, a neighbour of ϕ in G_2, is not in the boundary of ϕ_2 in G_2. Since ϕ_2 and ϕ must be disconnected in G_1, and since $\phi_2 \rightarrow \phi$ is in P_2^*, we conclude that the edge $\phi_2 - \phi$ must be oriented as $\phi_2 \rightarrow \phi$ in P^{**}. Now, Given that ϕ_2 is a root or a root* node of D_2, and having shown that nodes such as ϕ_2 whose boundary in G_2 and $G_\cap$ comprise only of their descendants in D_2 and $D_\cap$ are weakly transitive in P^{**}, the chain $\phi_2 \rightarrow \phi \rightarrow x$ in P^{**} tells us that ϕ_2 and x must be connected in P^{**} either with an undirected edge or as $\phi_2 \rightarrow x$. Given

that ϕ_2 cannot be in the boundary of x in G_2, as otherwise, since the boundary of x is a subset of or equal to that of the boundary y in G_2, we would have y also be in the boundary of ϕ_2 in G_2, we conclude that ϕ_2 and x must be connected in G_1 and G_1 only. Furthermore, since the edge $\phi_2 - x$ is either undirected in P^{**} or oriented as $\phi_2 \rightarrow x$, then this edge must also be either undirected in P_1^* or it must be oriented as $\phi_2 \rightarrow x$, implying that the boundary of ϕ_2 in G_1 is a subset of or equal to that of the boundary of x. Finally, we note that under the conditions specified ϕ_2 and y must be connected in G_1. Otherwise, since ϕ_2 and y are not connected in G_2, then the triplet $\phi_2 - x - y$ would have formed a v-structure in G, and the edge $x - y$ would have not been oriented as $x \rightarrow y$ in P^{**}. Now, since the boundary of ϕ_2 in G_1 must be a subset of or equal to that of the boundary of x, and ϕ_2 and y must be connected in G_1, we conclude that x and y must also be connected in G_1, which is against the conditions we have specified.

(II) Now lets assume that the nodes ϕ and x are connected in **both** G_1 and G_2.

(II.I) Now, if x and y are **only** connected in G_1, then the edge $x - y$ must be oriented as $x \rightarrow y$ in P_1^* since it is oriented as $x \rightarrow y$ in P^{**}. Therefore, since P_1^* is strongly transitive, and we have either the chain $\phi \rightarrow x \rightarrow y$ or the chain $\phi - x \rightarrow y$ in P_1^*, then the edge $\phi - y$ must be oriented as $\phi \rightarrow y$ in P_1^*. Now, if ϕ and y are disconnected in G_2, we can then immediately conclude that the edge $\phi - y$ must be oriented as $\phi \rightarrow y$ in P^{**} also. In fact, ϕ and y cannot be connected in G_2. Otherwise, the triplet $x - \phi - y$ would have formed a v-structure in G_2, and the edge $x - \phi$ would have been oriented as $x \rightarrow \phi$ in P_2, and this edge would not have been oriented as $\phi \rightarrow x$ in P^{**}.

(II.II) Finally, lets assume that the nodes x and y are also connected in **both** G_1 and G_2. We immediately note that ϕ and y must also be connected in both G_1 and G_2. Otherwise, the triplet $\phi - x - y$ would have formed a v-structure in G_1 or G_2,

preventing the formation of $x \to y$ in P^{**}. This lets us conclude that x, y, and ϕ form a connected component in $G_\cap$. Now, since the edges $\phi - x$ and $x - y$ are oriented as $\phi \to x \to y$ in P^{**}, then the same edges must be oriented similarly, $x \to y$ and $\phi \to x$, in $P^*_\cap$, the $\{G, G_1, G_2\}$-mirrored sink graph of $G_\cap$. Finally, since $P^*_\cap$ is strongly transitive, then the edge $\phi - y$ must be oriented as $\phi \to y$ in $P^*_\cap$, and therefore, the edge $\phi - y$, must be oriented as $\phi \to y$ in P^{**}.

Case 4*. Lets say the edges $\phi - x$ and $x - y$ are oriented as $\phi - x \to y$ in P^{**}. We show that the nodes ϕ and y must be connected in P^{**} and the edge $\phi - y$ must either be undirected in P^{**} or it must be oriented as $\phi \to y$.*

Here we show that if G_1 and G_2 are generated by two ancestrally faithful, marginally faithful, intersection matched DAGs or *DGs*, D_1 and D_2, then the nodes in $D_\cap$ whose boundary in $D_\cap$ comprises of only their descendants in $D_\cap$ satisfy this case of weak transitive in P^{**}.

First we note that the nodes ϕ and y must be connected in G. Otherwise, the triplet $\phi - x - y$ forms a v-structure in G, forcing the edge $x - y$ to be oriented as $x \leftarrow y$ in P, the sink graph of G. If the edge $x - y$ was oriented as $x \leftarrow y$ in P, however, this edge would not have been oriented as $x \to y$ in the merged sink graph of G_1 and G_2, P^{**}. Therefore, the nodes ϕ and y must be connected in G, and for that reason they must also be connected in G_1 or G_2.

(I) Now lets assume that nodes x and y are **only** connected in G_1. Since $x \to y$ is in P^{**}, we conclude that the edge $x - y$ must have been oriented as $x \to y$ in P^*_1.

(I.I) Now, if the nodes ϕ and x are also connected in G_1, then the edge $\phi - x$ must be undirected in P^*_1 since this edge is undirected P^{**}. Since P^*_1 is strongly transitive, then the edge $\phi - y$ must be oriented as $\phi \to y$ in P^*_1. Now, if ϕ and y are disconnected in G_2, we can then immediately conclude that the edge $\phi - y$ must be oriented as $\phi \to y$ in P^{**}. On the other hand, if ϕ and y are connected in G_2,

then the edge $\phi - y$ must either be undirected in P_2^* or it must be oriented as $\phi \to y$: $y \in \Omega_{G_\cap}(\phi)$ and ϕ is $G_\cap$-outer external in G_2 and G. Furthermore, ϕ must be external in $G_\cap$ and the edge $\phi - y$ must be either undirected in $P_\cap$ or it must be oriented as $\phi \to y$ in $P_\cap$. This lets us conclude that $\phi - y$ must be oriented as $\phi \to y$ in P^{**}.

(I.II) If ϕ and x are not connected in G_1, then, on the one hand, ϕ and x must be connected in G_2. On the other hand, ϕ and y cannot be connected in G_2. Otherwise, since x and y were assumed to be disconnected in G_2, the triplet $y - \phi - x$ would have formed a v-structure in G_2 and the edge $\phi - x$ would have been oriented as $\phi \leftarrow x$ in P_2. Then the edge $\phi - x$ would not have been undirected in P^{**}. Therefore, ϕ and y are disconnected in G_2 and must be connected in G_1 and only in G_1. Then x, y, and ϕ form a v-structure $x - y - \phi$ in G_1, and the edge $y - \phi$ can only be either doubly oriented $y \leftrightarrow \phi$ in P_1^* or it must oriented as $y \leftarrow \phi$ in P_1^*. If the edge $y - \phi$ is oriented as $y \leftarrow \phi$ in P_1^*, since y and ϕ are disconnected in G_1, we can immediately conclude that this edge must also be oriented as $y \leftarrow \phi$ in P^{**}. We now show that $y - \phi$ cannot be doubly oriented in P_1^* under the conditions we have considered so far.

First we note that the edge $y - \phi$ must be undirected in P or it must be oriented as $y \leftarrow \phi$ (since P is strongly transitive, and $\phi \to x \to y$ is in P^{**}). Now, If $y - \phi$ is doubly oriented in P_1^*, then it must have been either doubly oriented in P_1 or it must have been oriented as $y \to \phi$. Since we had assumed that ϕ and x were disconnected in G_1, however, the edge between ϕ and y cannot be oriented as $\phi \leftarrow y$ in the sink graph of G_1: x is in the boundary of y in G_1 while it is not in the boundary of ϕ and $\Omega_{G_1}(y) \not\subseteq \Omega_{G_1}(\phi)$. Then, the only possibility is for the edge $\phi - y$ to be doubly oriented in P_1. Now, since the edge $y - \phi$ is doubly oriented in P_1, we conclude that ϕ must have some ancestor in D_1, ϕ_1, that is a root or root* node of D_1 and whom y is not a descendant of: $\phi_1 \geq_{D_1} \phi \wedge \phi_1 \not\geq_{D_1} y$. Then ϕ must be connected to ϕ_1 in P_1^* with an edge oriented as $\phi_1 \to \phi$: ϕ_1 is an ancestor of ϕ in D_2 and there is a

node, namely y, in the boundary of ϕ that is not in the boundary of ϕ_1. Now, given that ϕ is a node whose boundary in $G_\cap$ comprises of only its descendants in $D_\cap$, we conclude that ϕ_1 cannot be connected to ϕ in G_2. Otherwise, if ϕ_1 was connected to ϕ in G_2, then ϕ_1 would have been in the boundary of ϕ in $G_\cap$, and since any node in the boundary of ϕ in $G_\cap$ is by our assumption a descendant of ϕ in $D_\cap$, ϕ_1 would have been a descendant of ϕ in $D_\cap$, and on that account a descendant of ϕ in D_1 as well. However, this is impossible since we have already established that the node y, a neighbour of ϕ in G_1, is not in the boundary of ϕ_1 in G_1. Since ϕ_1 and ϕ must be disconnected in G_2, and since $\phi_1 \to \phi$ is in P_1^*, we conclude that the edge $\phi_1 - \phi$ must be oriented as $\phi_1 \to \phi$ in P^{**}. Now, Given that ϕ_1 is a root or a root* node of D_2, and having shown that nodes such as ϕ_1 whose boundary in G_1 and $G_\cap$ comprise only of their descendants in D_1 and $D_\cap$ are weakly transitive in P^{**}, the chain $\phi_1 \to \phi \to x$ in P^{**} tells us that ϕ_1 and x must be connected in P^{**} either with an undirected edge or an edge oriented as $\phi_1 \to x$. Given that ϕ_1 cannot be in the boundary of x in G_1, as otherwise, since the boundary of x is a subset of or equal to that of the boundary y in G_1, we would have y also be in the boundary of ϕ_1 in G_1, we conclude that ϕ_1 and x must be connected in G_2 and G_2 only. Furthermore, since the edge $\phi_1 - x$ is either undirected in P^{**} or oriented as $\phi_1 \to x$, then this edge must be also either undirected in P_2^* or it must be oriented as $\phi_1 \to x$, implying that the boundary of ϕ_1 in G_2 is a subset of or equal to that of the boundary of x. Finally, we note that under the conditions specified ϕ_1 and y must be connected in G_2. Otherwise, since ϕ_1 and y are not connected in G_1, then the triplet $\phi_1 - x - y$ would have formed a v-structure in G, and the edge $x - y$ would have not been oriented as $x \to y$ in P^{**}. Now, since the boundary of ϕ_1 in G_2 must be a subset of or equal to that of the boundary of x, and ϕ_1 and y must be connected in G_2, we conclude that x and y must also be connected in G_2, which is against the conditions we have specified.

(II) Now lets assume that x and y are connected both in G_1 and G_2.

(II.I) Now, if ϕ and x are **only** connected in G_1, then the edge $\phi - x$ must be undirected in P_1^* since this edge is undirected in P^{**}. On the other hand, since the edge $x - y$ is oriented as $x \to y$ is in P^{**}, then this edge must either be undirected in P_1^* or it must be oriented as $x \to y$ in P_1^*. Given that both ϕ and x, and x and y are assumed to be connected in G_1, then ϕ and y must also be connected in G_1: otherwise, the triplet $\phi - x - y$ would form a v-structure in G_1, preventing the formation of $x \to y$ or $\phi - x$ in P^{**}. Now, since $\phi - x$ is undirected in P_1^*, and the edge $x - y$ must either be undirected in P_1^* or it must be oriented as $x \to y$ in P_1^*, and given that P_1^* is strongly transitive, then the edge $\phi - y$ either remains undirected in P_1^* or it must be oriented as $\phi \to y$. Now, if ϕ and y are disconnected in G_2, we can then immediately conclude that the edge $\phi - y$ either remains undirected in P^{**} or it must be oriented as $\phi \to y$ in P^{**}. In fact, the nodes ϕ and y must be disconnected in G_2. Otherwise, the the triplet $x - y - \phi$ would have formed a v-structure in G_2, and the edge $x - y$ would have been oriented as $x \to y$ in P_2. If that was the case, however, the edge $x - y$ would not be undirected in P^{**}.

(II.II) Finally, lets assume that x and y are also connected in both G_1 and G_2. We immediately note that the nodes ϕ and y must also be connected in both G_1 and G_2, otherwise, the triplet $\phi - x - y$ would form a v-structure in G_1 or G_2, and the edge $x - y$ would not be oriented as $x \to y$ in P^{**}. This lets us conclude that x, y, and ϕ form a connected component in $G_\cap$. Now, since the edges $x - y$ and $\phi - x$ are oriented as $\phi - x \to y$ in P^{**}, then these edges must be oriented similarly, $x \to y$ and $\phi - x$, in $P_\cap^*$, the $\{G_1, G_2, G\}$-mirrored sink graph of $G_\cap$. Finally, since $P_\cap^*$ is strongly transitive, then the edge $\phi - y$ must be oriented as $\phi \to y$ in $P_\cap^*$, and therefore, as $\phi \to y$ in P^{**}.

If

Lets assume for every external clique of G_1, G_2, and $G_\cap$, there exists at least one node, ψ_i^1, ψ_j^2, and $\psi_k^\cap$ that satisfies the conditions specified in the lemma. We construct two intersection matched, marginally faithful, ancestrally faithful DAGs R_1^* and R_2^* generating G_1 and G_2 as follows. **Note that the two DAGs we construct here are strongly acyclic**.

Step 1

We first construct three DAGs R_1, R_2, and $R_\cap$. We construct R_1 by selecting one node, ψ_i^1, for every external clique of G_1 that satisfies the conditions (1.1)-(1.5) specified in the lemma. We then connect this node to all other nodes in its neighbourhood in G_1 through a directed arrow as shown next: $\forall x \in \Omega_1(\psi_i^1) : \psi_i^1 \rightarrow x$. We construct R_2 and $R_\cap$ similarly, by first selecting one node per external clique of G_2 and $G_\cap$, ψ_j^2 and $\psi_k^\cap$, that satisfies the conditions, respectively (2.1-2.5) and (3.1-3.3), specified in the lemma, and then connecting it too all other nodes in their boundaries as described.

Step 2

We can always select the ψ's such that R_1, R_2 and $R_\cap$ are pairwise weakly acyclic. First, choose any arbitrary $R_\cap$. Now, lets say the chosen R_1 and $R_\cap$ are not weakly acyclic and they have K instance where the edges in R_1 and $R_\cap$ are reversed. Thus, there exists two nodes ψ_i^1 and $\psi_k^\cap$ such that $\psi_i^1 \rightarrow \psi_k^\cap$ is in R_1 and $\psi_i^1 \leftarrow \psi_k^\cap$ is in $R_\cap$. Since $\psi_k^\cap$ is $G_\cap$-outer external in G_1 and ψ_i^1 is external in G_1, then $\psi_k^\cap$ is also an external node of the same clique as that of ψ_i^1 in G_1. We construct a new R_1 by choosing $\psi_k^\cap$ in place of ψ_i^1 in the selection made in step 1. The new R_1 and $R_\cap$ then have at most $K-1$ instance where the edges in R_1 and $R_\cap$ are reversed.We can similarly construct an R_2 that is weakly acyclic w.r.t $R_\cap$.

Step 3

We note that if we have chosen R_1 and R_2 to both be weakly acyclic w.r.t $R_\cap$, then they are both weakly acyclic w.r.t one another. Assume the contrary and lets say

there exists two distinct nodes ψ_i^1 and ψ_j^2 such that $\psi_i^1 \rightarrow \psi_j^2$ is in R_1 and $\psi_i^1 \leftarrow \psi_j^2$ is in R_2. Then the nodes ψ_i^1 and ψ_j^2 must have been directly connected both in G_1 and G_2, and therefore, they must have also been connected in $G_\cap$. Since the edge $\psi_i^1 - \psi_j^2$ is in $G_\cap$, then, in the construction of $R_\cap$, we must have selected at least node, $\psi_k^\cap$, in an external clique covering this edge, and we must have connected this node to both ψ_i^1 and ψ_j^2 with the following orientation: $\psi_i^1 \leftarrow \psi_k^\cap \rightarrow \psi_j^2$. On the other hand, since the edge $\psi_i^1 - \psi_k^\cap$ must also be in G_1 (owing to the fact that $G_\cap \subseteq G_1$), then we must have had the edge $\psi_i^1 \rightarrow \psi_k^\cap$ in R_1. If $\psi_k^\cap$ and ψ_i^1 are distinct nodes, this would imply that $R_\cap$ and R_1 were not weakly acyclic. Thus, the nodes $\psi_k^\cap$ and ψ_i^1 must have been the same. Similarly, we can conclude that $\psi_k^\cap$ is the same node as ψ_j^2, which would imply that ψ_j^2 and ψ_i^1 are the same node. This is however impossible since we had assumed ψ_j^2 and ψ_i^1 are two distinct nodes.

Step 4

If the R_1, R_2, and $R_\cap$ are constructed as instructed above to be pairwise weakly acyclic, then their union $R = R_1 \cup R_2 \cup R_\cap$ must be acyclic.

Lets assume that there is a cycle. Either all edges in the cycle come from R_1 or R_2 or there exists at least one edge that is *only* present in $R_\cap$. Lets say one such edge exists and let us denote this edge with $\psi_k^\cap \rightarrow \overline{\psi}$. We show that R must then have a cycle without this edge as well.

To see this, consider the edge that comes right before this edge in the cycle: $\psi \rightarrow \psi_k^\cap \rightarrow \overline{\psi}$. The edge $\psi \rightarrow \psi_k^\cap$ cannot be in $R_\cap$. An edge in $R_\cap$ connects one external node per external clique of $G_\cap$ to a node in that external clique. If the edge $\psi \rightarrow \psi_k^\cap$ was in $R_\cap$, then we would conclude that ψ was an external node of some external clique, Γ, in $G_\cap$, and $\psi_k^\cap$ was some node in that external clique. The edge $\psi_k^\cap \rightarrow \overline{\psi}$ in $R_\cap$, on the other hand, implies that the node $\psi_k^\cap$ is an external node of some other external clique, $\hat{\Gamma}$, of $G_\cap$. This is not possible since we already established

that $\psi_k^{\cap}$ is in the external clique Γ and thus it cannot be an external node of $\hat{\Gamma}$. Then, the edge $\psi \rightarrow \psi_k^{\cap}$ must be in R_1 or in R_2.

Let's say the edge $\psi \rightarrow \psi_k^{\cap}$ is in R_1. The edge $\psi \rightarrow \psi_k^{\cap}$ in R_1 implies that ψ and $\psi_k^{\cap}$ are directly connected in G_1. Now, since $\psi_k^{\cap}$ is $G_{\cap}$-outer external in G_1, we have $\forall x \in \Omega_{G_{\cap}}(\psi_k^{\cap}) : \Omega_{G_1}(\psi_k^{\cap}) \subseteq \Omega_{G_1}(x)$. Given that $\overline{\psi}$ is in $\Omega_{G_{\cap}}(\psi_k^{\cap})$, then $\Omega_{G_1}(\psi_k^{\cap}) \subseteq \Omega_{G_1}(\overline{\psi})$, and since ψ is in $\Omega_{G_1}(\psi_k^{\cap})$, then ψ must also be in $\Omega_{G_1}(\overline{\psi})$ and the nodes ψ and $\overline{\psi}$ must be connected in G_1. As the nodes ψ and $\overline{\psi}$ must be connected in G_1, then we must have connected the nodes ψ and $\overline{\psi}$ through the edge $\psi \rightarrow \overline{\psi}$ in the construction of R_1 and we must have a cycle in R without the edge $\psi_k^{\cap} \rightarrow \overline{\psi}$.

We now show R cannot contain a cycle where all the edges in the cycle are in R_1 or R_2. We show this by proving that existence of such a cycle is only possible when R_1 and R_2 are not weakly acyclic.

Assume the contrary and lets say there is such a cycle $\{\psi_1, \psi_2, ...\psi_n, \psi_1\}$ in R. First we note that given the construction of R_1 and R_2, the longest directed path in R_1 and R_2 is one and there is no chain of length two in either of these two DAGs. Therefore, (1) no edge in the cycle can be present in both R_1 and R_2, and (2) no two successive edges in the cycle can be both present in R_1 and no two successive edges can be both present in R_2–in other words, if the edge $\psi_i \rightarrow \psi_{i+1}$ is in R_1, then the edge $\psi_{i+1} \rightarrow \psi_{i+2}$ must be in R_2. The other important detail about the structure of the graphs R_1, R_2 and $R_{\cap}$, is that for any edge $\psi \rightarrow x$ in these graphs, this edge must appear as either undirected in P^{**} or it must be present with the same orientation, $\psi \rightarrow x$. This can be easily verified by reminding ourselves of the conditions specified in the Lemma (in fact the conditions were engineered so as to guarantee this property). For instance, consider the edge $\psi \rightarrow x$ in R_1. Using the following properties, we can

immediately verify that ψ and x are connected in P^{**} and the edge $\psi - x$ is either undirected or is oriented as $\psi \to x$ in P^{**}

1 the nodes ψ and x must be connected in G_1 and hence in $G = G_1 \cup G_2$.

2 the edge $\psi - x$ must either be undirected or oriented as $\psi \to x$ in the sink graph of G_1 owing to the fact that ψ is an external node of G_1.

3 the edge $\psi - x$, if present in $G_\cap$, must either be undirected or oriented as $\psi \to x$ in the sink graph of $G_\cap$ since ψ is an external node in $G_\cap$.

4 the edge $\psi - x$ must either be undirected or oriented as $\psi \to x$ in the sink graph of G since ψ is G_1-external in G.

5 the edge $\psi - x$ if present in G_2 must either be undirected or oriented as $\psi \to x$ in the sink graph G_2 owing to ψ being a $G_\cap$ outer external node in G_2.

Now, to show that R cannot contain any cycles, first, consider the path $\psi_1 \to \psi_2 ... \to \psi_{n-1}$. Given that ψ_1 is weakly transitive in P^{**}, and that every edge in this sequence is either undirected in P^{**} or is oriented as $\psi_i \to \psi_{i+1}$, we can immediately conclude that the nodes ψ_1 and ψ_{n-1} are connected in P^{**} and the edge $\psi_1 - \psi_{n-1}$ is either undirected in P^{**} or is oriented as $\psi_1 \to \psi_{n-1}$. The complement of this directed path in the cycle, $\psi_{n-1} \to \psi_n \to \psi_1$, on the other hand, implies that the edge $\psi_{n-1} - \psi_1$ must be either undirected in P^{**} or it must be oriented as $\psi_{n-1} \to \psi_1$. Combining these two statements we conclude the edge $\psi_{n-1} - \psi_1$ must be undirected in P^{**}.

Now, let's say the edge $\psi_{n-1} \to \psi_n$ is in R_1. Then, the edge $\psi_n \to \psi_1$ must have been in R_2 and only in R_2. Finally, the edge $\psi_1 \to \psi_2$ must have been in R_1 and R_1 only. Then, the edge $\psi_1 - \psi_{n-1}$ cannot be in G_1 since ψ_1 and ψ_{n-1} are external nodes of two different external cliques in G_1. Therefore, since the nodes ψ_1 and ψ_{n-1} must be connected in P^{**}, and on that account they must be connected in $G = G_1 \cup G_2$, we conclude that ψ_1 and ψ_{n-1} must be connected in G_2. Now, since the edge $\psi_1 - \psi_{n-1}$ is

undirected in P^{**}, we conclude ψ_1 and ψ_{n-1} are also connected by an undirected edge in the sink graph of G_2. Since the edge $\psi_1 - \psi_{n-1}$ must have been undirected in the sink graph of G_2, then, every node in the boundary of ψ_1 in G_2 must have also been in the boundary of ψ_{n-1}. Since ψ_n is in the boundary of ψ_1 in G_2, we conclude that ψ_n must have also been in the boundary of ψ_{n-1} in G_2. Then, in the construction of R_2 we must have had connected ψ_n to ψ_{n-1} through the edge $\psi_n \to \psi_{n-1}$. However, this would imply that the DAGs R_1 and the R_2 were not weakly acyclic.

Step 5

Form the transitive closure of R, R_{TC}. Since every directed path in R begins at a node that is weakly transitive in P^{**}, ψ, and every edge along the path is either undirected in P^{**} or is oriented with the same orientation as in R, then every edge $\psi \to x$ in R_{TC} must appear in P^{**} as either $\psi \to x$ or $\psi - x$. Therefore, every such edge must also appear in P_1 or P_2 with the same orientation or as undirected. We can then add any such edges to either R_1 or R_2 without affecting their marginal dependency graph. Lets refer to these extended DAGs as R_1^* and R_2^*.

Step 6

The two DAGs R_1^* and R_2^* are ancestrally faithful, marginally faithful, intersection matched, and generate G_1 and G_2.

1 The marginal dependency graph of R_1^* is G_1 and the marginal dependency graph of R_2^* is G_2. First, we note that since the marginal dependency graph of R_1 is G_1, and R_1^* is a supergraph of R_1, then the marginal dependency graph of R_1^* is a *supergraph* of G_1. On the other hand, the marginal dependency graph of R_1^* is a *subgraph* of G_1 as R_1^* is a DAG who is a subgraph of some DAG in the completion of reduced the sink graph of G_1 (Lemma S.A.1). For every edge $\psi_i \to x$ in R_1^*, ψ_i and x must be connected in G_1, and the edge $\psi_i - x$ must

either be undirected or oriented as $\psi_i \to x$ in the sink graph of G_1 as discussed in Step 5.

2 The marginal dependency graph of the intersection of the two DAGs R_1^* and R_2^* is $G_\cap$ since $R_\cap$ is a subgraph of both R_1^* and R_2^*, and $\Sigma(R_\cap) = G_\cap$.

3 The marginal dependency graph of the union of the two DAGs R_1^* and R_2^* is G. Every edge in R_1^* or R_2^* must appear with the same orientation or as undirected in P^{**}. Then, R_1^* and R_2^* must be both subgraphs of DAGs in the completion of the reduced sink graph of G, since P^{**} is the $\{G_1, G_2, G_\cap\}$-mirrored sink graph of G. Then, by Lemma S.A.1, we conclude that the marginal dependency graph of the union of the two DAGs R_1^* and R_2^* is G.

4 The DAGs R_1^* and R_2^* are ancestrally faithful. To show this we need to prove that the transitive closure of $R_1^* \cup R_2^*$ is a subgraph of the union of the transitive closures of R_1^* and R_2^*. Owing to *Step 5*, the union of the two DAGs, $R_1^* \cup R_2^*$, is exactly equal to the DAG R_{TC}. Then, the transitive closure of $R_1^* \cup R_2^*$ is the transitive closure of R_{TC}, and since R_{TC} is a transitive DAG, then the transitive closure of $R_1^* \cup R_2^*$ is nothing but R_{TC} itself. Now, since the union of the two DAGs R_1^* and R_2^* is equal R_{TC}, then the union of their transitive closures must be a supergraph of R_{TC}, which by our previous argument is the transitive closure of the union of R_1^* and R_2^*.

□

E.1

Theorem 3.2. Given two marginal dependency graphs G_1 and G_2, there exists two ancestrally faithful, marginally faithful, intersection matched DGs generating G_1 and G_2 only if there exists two two ancestrally faithful, marginally faithful, intersection matched DAGs generating the given marginal dependency graphs.

Proof. We use the following notation in our proof. P_1 and P_2 are used to denote the sink graphs of G_1 and G_2. We use G to represent the union of the two marginal dependency graphs, $G = G_1 \cup G_2$, and P to denote its sink graph. We use $G_\cap$ for the intersection of the two marginal dependency graphs, $G_\cap = G_1 \cap G_2$, and $P_\cap$ to denote its sink graph. We use P_1^* and P_2^* for the $G-$mirrored sink graphs of G_1 and G_2, $P_\cap^*$ to denote $\{G, G_1, G_2\}-$mirrored sink graph of $G_\cap$, and finally P^{**} for the merged sink graph of G_1 and G_2.

Lets say the two DGs D_1 and D_2 are ancestrally faithful, marginally faithful, intersection matched DGs generating G_1 and G_2. Let us represent the union and intersection of these two DGs, respectively, by D and $D_\cap$. We find a set of nodes in D_1, D_2, and $D_\cap$ such that:

(1) For every external clique of G_1 , at least one node is (1.1) external in G_1, (1.2) G_1-outer external in $G = G_1 \cup G_2$, (1.3) external in $G_\cap = G_1 \cap G_2$, (1.4) $G_\cap$-outer external in G_2, and (1.5) ***weakly*** transitive in P^{**}, the merged sink graph of G_1 and G_2.

(2) For every external clique of G_2 , at least one node that is (2.1) external in G_2, (2.2) G_2-outer external in G , (2.3) external in $G_\cap$, (2.4) $G_\cap$-outer external in G_1, and (2.5) ***weakly*** transitive in P^{**}.

(3) For every external clique of $G_\cap = G_1 \cap G_2$, at least one node that is (3.1) external in $G_\cap$, (3.2) $G_\cap$-outer external in G_1, G_2, and G, and (3.3) ***weakly*** transitive in P^{**}.

Part 1: *Finding Appropriate Nodes In External Cliques of* $G_\cap$

For every edge $x - y$ in $G_\cap$ connecting the nodes x and y, we find a node in $D_\cap$, π, such that π is in the boundary of x and y in $G_\cap$ and π is (3.1) external in $G_\cap$, (3.2) $G_\cap$-outer external in G_1, G_2, and G, and (3.3) ***weakly*** transitive in P^{**}.

Since x and y are connected in $G_\cap$, and given the fact that $D_\cap$ has generated $G_\cap$, we can conclude that x and y must have had a common ancestor, π_1, in $D_\cap$ and $\pi_1 \in \Omega_{G_\cap}(x) \cap \Omega_{G_\cap}(y)$. Now either π_1 itself is a root* node in $D_\cap$ or it has a ancestor in $D_\cap$, π, that is a root* node in $D_\cap$. To see this, let's say neither π_1 nor any of its ancestors in $D_\cap$, $\Pi_\cap(\pi_1) = \{\pi_1, \pi_2..., \pi_m\}$, are a root* node in $D_\cap$. Since π_1 is not a root* node in $D_\cap$, then π_1 must have an ancestor in $D_\cap$ that is not its descendant. Lets call this ancestor π_1^*. Then, π_1^* is an ancestor of π_1 in $D_\cap$, and π_1^* must be in the set $\Pi_\cap(\pi_1)$. Let's assume that π_1^* is the node labeled as π_2 in this set. Now, since π_2 is not a root* node in $D_\cap$, then π_2 must have an ancestor in $D_\cap$ that is not its descendant. Lets call this ancestor π_2^*. Then π_2^* is an ancestor of π_2 in $D_\cap$, and since π_2 is an ancestor of π_1 in $D_\cap$, then π_2^* must have been an ancestor of π_1 in $D_\cap$ also. Therefore, π_2^* must be in the set $\Pi_\cap(\pi_1)$. Furthermore, the node π_2^* must be a different node than π_1. π_2^* cannot be π_1 since π_*^2 is a node that's not a descendant of π_2 in $D_\cap$. Thus π_2^* must be a node in the collection $\{\pi_3, \pi_4..., \pi_m\}$. Lets then assume that π_2^* is the node labeled as π_3 in this set. Again, since π_3 is not a root* node in $D_\cap$, then π_3 must have an ancestor in $D_\cap$ that is not its descendant. Using this logic we can then climb the ancestors of π_1 in $D_\cap$ one by one, iteratively, and as we make this climb, for every new node that we reach, since this node is not a root* node in $D_\cap$, we can always find a node that we have not visited previously and continue the climb indefinitely which should be impossible since the graphs we consider are defined over finite domains.

Therefore, we conclude for every edge $x - y$ in $G_\cap$, we can find a node π in $D_\cap$, such that π is in the boundary of x and y in $G_\cap$, and π is a root* node in $D_\cap$. Since π is a root* node in $D_\cap$, then π satisfies the following properties:

1 The node π is an external node in $G_\cap$. Since π is a root* node in $D_\cap$, then all its ancestors in $D_\cap$ must be its descendants also. Then any node x sharing a

common ancestor with π in $D_\cap$, must also be its descendant. Therefore, the boundary of π in $G_\cap$ is a subset of or equal to that boundary of every node, x, that its connected to π in $G_\cap$: $\forall x \in \Omega_{G_\cap}(\pi) : \Omega_{G_\cap}(\pi) \subseteq \Omega_{G_\cap}(x)$.

2 The node π $G_\cap$-outer external in G_1, G_2, and G. As before, any node x sharing a common ancestor with π in $D_\cap$, must also be its descendant. As $D_\cap$ is a subgraph of D_1, D_2, any such node, x, must also remain π's descendant in D_1, D_2, and $D = D_1 \cup D_2$. Then, the boundary of π in G_1, G_2, and G must be a subset of or equal to that boundary of every node, x, that its connected to π in $G_\cap$.

3 This node π must also be ***weakly*** transitive in P^{**}. This immediately results from our proof of weak transitivity of nodes in $D_\cap$ whose boundary in $D_\cap$ comprises of only their descendants in $D_\cap$ whenever G_1 and G_2 are generated by two ancestrally faithful, marginally faithful, intersection matched DGs (Supplementary D.1).

Part 2: *Finding Appropriate Nodes In External Cliques of G_1 and G_2*

We show that for every edge $x - y$ in G_1, we can find a node $\overline{\pi}$ in D_1, such that $\overline{\pi}$ is in the boundary of x and y in G_1, and $\overline{\pi}$ is a root* node in both D_1 and $D_\cap$.

We previously showed that for every edge $x - y$ in $G_\cap$, we can find a node, π_1, in $D_\cap$ such that π_1 is in the boundary of x and y in $G_\cap$, and π_1 is a root* node in $D_\cap$. The same applies to the graph G_1. That is, for every edge $x - y$ in G_1 , we can find a node, π_1, in D_1, such that π_1 is in the boundary of x and y in G_1, and π_1 is a root* node in D_1. Now, consider the ancestors of such a π_1 in D_1 and let us refer to this set with the notation $\Pi_1(\pi_1) = \{\pi_1, ..., \pi_m\}$. Note that all the nodes in $\Pi_1(\pi_1)$ are root* nodes of D_1. Assume the opposite and let's say π_i has an ancestor, η, in D_1 that is not its descendant. Then, η must be an ancestor of π_1, and since π_1 is a root* node of D_1, then π_1 must itself be an ancestor of η in D_1. Now, since π_i is an ancestor of π_1

in D_1, and given that π_1 is an ancestor of η in D_1, then π_i must be an ancestor of η in D_1 also. However, we had assumed that η is not a descendant of π_i.

We now show that at least one node in $\Pi_1(\pi_1)$ must be a root* node of $D_\cap$ as well. If $\Pi_1(\pi_1)$ only contains π_1, then π_1 is clearly a root* node of $D_\cap$ as well as that of D_1 (π_1 has no ancestors in neither D_1 nor D_2 besides itself and thus by default it is a root* node of both D_1 and $D_\cap$). Lets assume none of the nodes in $\{\pi_1, ..., \pi_m\}$ are a root* node in $D_\cap$. Since π_1 is not a root* node in $D_\cap$, then π_1 must have an ancestor in $D_\cap$ that is not its descendant in $D_\cap$. Lets call this ancestor π_1^*. Then π_1^* is an ancestor of π_1 in both $D_\cap$ and D_1, and π_1^* must be in the set $\Pi_1(\pi_1)$. Let's assume that π_1^* is the node labeled as π_2 in this set. Now, since π_2 is not a root* node in $D_\cap$, then π_2 must have an ancestor in $D_\cap$ that is not its descendant in $D_\cap$. Lets call this ancestor π_2^*. Then π_2^* is an ancestor of π_2 in $D_\cap$, and since π_2 is an ancestor of π_1 in $D_\cap$, then π_2^* must have been an ancestor of π_1 in $D_\cap$ also. Therefore, π_2^* must also be in the set $\Pi_1(\pi_1)$. Now, π_2^* cannot be π_1. π_2^* cannot be π_2 since π_*^2 is a node that's not a descendant of π_2 in $D_\cap$. Lets then assume that π_2^* is the node labeled as π_3 in this set. Again, since π_3 is not a root* node in $D_\cap$, then π_3 must have an ancestor in $D_\cap$ that is not its descendant. Using this logic we can then climb the ancestors of π_1 in $D_\cap$ one by one, iteratively, and as we make this climb, for every new node that we reach, since this node is not a root* node in $D_\cap$, we can always find a node that we have not visited previously and continue the climb indefinitely which should be impossible since the graphs we consider are defined over finite domains.

Therefore, we conclude for every edge $x - y$ in G_1, we can find a node $\overline{\pi}$ in D_1, such that $\overline{\pi}$ is in the boundary of x and y in G_1, and $\overline{\pi}$ is a root* node in both D_1 and $D_\cap$. Since $\overline{\pi}$ is a root* node in both D_1 and $D_\cap$, then $\overline{\pi}$ is (1.1) external in G_1, (1.2) G_1-outer external in $G = G_1 \cup G_2$, (1.3) external in $G_\cap = G_1 \cap G_2$, (1.4) $G_\cap$-outer external in G_2, and (1.5) ***weakly*** transitive in P^{**}, the merged sink graph of G_1 and

G_2. This immediately results from our proof of weak transitivity of nodes in D_1 (D_2) whose boundary in G_1 (G_2) and $G_\cap$ comprises of only their descendants in D_1 and $D_\cap$ whenever G_1 and G_2 are generated by two ancestrally faithful, marginally faithful, intersection matched DAGs or *DGs* (Supplementary D.1).

□

E.2

Theorem 3.3. If there exists an ancestrally faithful, marginally faithful, intersection matched pair of DGs generating G_1 and G_2, then there exists a strongly acyclic pair of DGs generating G_1 and G_2.

Proof. Theorem 3.2 tells us if there exists an ancestrally faithful, marginally faithful, intersection matched pair of DGs generating G_1 and G_2, then there exists an ancestrally faithful, marginally faithful, intersection matched pair of DAGs generating G_1 and G_2. The existence of two such DAGs implies the existence of a selection of nodes satisfying the conditions of Theorem 3.1, and we have already shown in Supplementary D.1 how to construct two strongly acyclic DAGs generating G_1 and G_2 using nodes that satisfy these conditions.

□

F.1

Lemma S.F.1. An undirected graph, G, is generated by the DG D, only if D is a subgraph of some DG in the completion of the reduced sink graph of G.

Proof. We show that $a \to b$ is in D, only if the edge $a - b$ is undirected or is oriented as $a \to b$ in the reduced sink graph of G. The connection $a \to b$ in D implies "a" and "b" are directly connected in G since D generates G and $\Sigma(D) = G$. Furthermore,

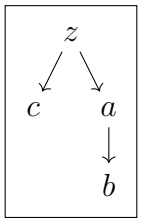


Figure 5.11: If a node, c, has a common ancestor with a, then it shares the same ancestor with b also.

as "a" is a parent of "b" in D, then $\Omega_G(a) \subseteq \Omega_G(b)$; any node sharing a common ancestor with "a" shares the same ancestor with "b" as well. In other words, any node in the boundary of "a" in G must also be in the boundary of "b" (see Figure 5.11). Therefore, the edge $a - b$ must be either undirected or is oriented as $a \to b$ in the reduced sink graph of G, completing our proof of this Lemma.

□

Lemma S.F.2. Two undirected graphs, G and G_1, with both in Σ, and $G_2 \subseteq G_1$ are generated by DGs D and D_1 with $D_1 \subseteq D$, only if D_1 is subgraph of some DG in the completion of the reduced $G-$mirrored sink graph of G_1.

Proof. We show that $a \to b$ is in D_1 only if the edge $a - b$ is undirected or is oriented as $a \to b$ in the reduced $G-$mirrored sink graph of G_1. Since D_1 is assumed to generate G_1, $a \to b$ is in D_1 only if "a" and "b" are directly connected in $G_1 = \Sigma(D_1)$. Furthermore, "a" being an ancestor of "b" in D_1 implies $\Omega_{G_1}(a) \subseteq \Omega_{G_1}(b)$: any node sharing a common ancestor with "a" shares the same ancestor with "b" as well. Since D_1 is a subgraph of D, then "a" remains an ancestor of "b" in D. Now, as D is assumed to generate G, we conclude that $\Omega_G(a) \subseteq \Omega_G(b)$ also. Therefore, since the boundary of "a" is a subset of or equal to the boundary of "b" in both G and G_1, and we conclude $a \to b$ is in D_1, only if the edge $a - b$ is undirected or is oriented as $a \to b$ in the reduced $G-$mirrored sink graph of G_1. □

Theorem 4.1. $\Lambda_\delta(\{G\}, G, G_1)$ is a tight upper bound for $\Delta(G, G_1)$. That is, for all δD in $\Delta(G, G_1)$, δD is a subgraph of $\Lambda_\delta(\{G\}, G, G_1)$: $a \to b \in \delta D$ only if $a \to b$ or $a - b$ are in $\Lambda_\delta(\{G\}, G, G_1)$.

Proof. We will use the following definition in proving Theorem 4.1.

Definition: Full Completion. A DG is said to be a full completion of a PDG if it can be obtained from the PDG by orienting any undirected edge in the PDG as $a \leftrightarrow b$.

δD is a subgraph of D, a directed graph generating G. By Lemma S.F.1, D, itself, must be a subgraph of some DG in the completion of reduced sink graph of G. Therefore, δD is a subgraph of some DG in the completion of the reduced sink graph of G, and a and b are connected by the edge $a \to b$ in δD only if a and b are connected in the sink graph of G and the edge $a - b$ is either undirected in the sink graph of G or is oriented as $a \to b$.

On the other hand, δD cannot have any edge $a \to b$ where "a" and "b" are connected in the $G-$mirrored sink graph of G_1 by an undirected edge or an edge oriented as $a \to b$. To see this, assume the contrary and let's say there exists two DGs, D and D_1, with $D_1 \subseteq D$ and $\Sigma(D) = G$ and $\Sigma(D_1) = G_1$, where $\delta D = D - D_1$ contains a set of edges, $E = \{a_i \to b_i | i = 1...m\}$, that are in the reduced $G-$mirrored sink graph of G_1 as either undirected or oriented as $a_i \to b_i$. If such a pair of DGs existed, then the pair D and $D_1 \cup E$ would also be a solution to the dependency loss problem. The marginal dependency graph of D_1 is G_1, and since $D_1 \cup E$ is a supergraph of D_1, then the marginal dependency graph of $D_1 \cup E$ contains all the edges in G_1. On the other hand, according to the Lemma S.F.2 and our assumption that the edges in E are in the the reduced $G-$mirrored sink graph of G_1, then every edge $x \to y$ in $D_1 \cup E$ is also present in the reduced $G-$mirrored sink graph of G_1 as

either an undirected edge or an edge oriented as $x \to y$. As the marginal dependency graph of any DG in the completion of the reduced $G-$mirrored sink graph of G_1 is equal to G_1 (Lemma S.B.3) and $D_1 \cup E$ is a subgraph of some such DG, then the marginal dependency graph $D_1 \cup E$ must be a subgraph of G_1. Since the marginal dependency graph of $D_1 \cup E$ must be both a subgraph and a supergraph of G_1, then the marginal dependency graph of $D_1 \cup E$ is equal to G_1. By construction, the pair $(D, D_1 \cup E)$ have a smaller Hamming distance to another compared to the original pair (D, D_1). Therefore, we conclude the original pair (D, D_1) must have been an instance of an undesired solution to the dependency loss problem, which should have had been removed.

To see $\Lambda_\delta(\{G\}, G, G_1)$ is a tight upper-bound , we construct a solution, D and D_1, where $D - D_1$ is a full completion of $\Lambda_\delta(\{G\}, G, G_1)$. Take D to be the full completion of the sink graph of G, and D_1 to be the full completion of $G-$mirrored sink graph of G_1. By lemma S.A.1, the marginal dependency graph of D is G. By Lemma S.B.3, the marginal dependency graph of D_1 is G_1. Furthermore, D_1 is a subgraph of D: an edge is doubly oriented in D_1 only if the edge is undirected in the reduced $G-$mirrored sink graph of G_1, which in turns requires the edge to be undirected in the sink graph of G as well, and by that account, this edge must also be doubly oriented in D. Similarly, we can argue an edge is oriented as $a \to b$ in D_1 only if the same edge is either doubly oriented in D or is oriented as $a \to b$. By construction, the difference between D and D_1, $\delta D = D_1 - D_2$, is the full completion of $\Lambda_\delta(\{G\}, G, G_1)$.

□

CHAPTER 6

CAUSAL DISCOVERY USING DIRECTED TOPOLOGICAL OVERLAP MATRIX

6.1 Introduction

There are now a variety of causal discovery tools capable of identifying causal relations from *purely observational* data [83, 22]. These causal discovery tools, at their core, must transform statistical relations into causal ones. This transformation is often made possible through the local causal Markov condition and the formalisms of functional causal models [84, 80]. Unfortunately, the theoretical assumptions that underlie the local causal Markov condition are often not met in practice [80, 22, 21, 49]. This disconnect between theory and practice is especially marked in genomics. In genomics, the unwanted presence of measurement errors, averaging effects, and feedback loops significantly undermine the legitimacy of the local causal Markov condition. Furthermore, even the most sample efficient causal discovery algorithms require sample sizes orders above what is available even for the largest genomics databases out there [50, 49, 21, 52, 45]. This paper aims to construct a causal discovery tool suitable for a genomics setting whose legitimacy and practicality are not undermined by limited sample sizes or the unwanted presence of measurement errors, averaging effects, and feedback loops.

Briefly, the reasoning engine of our proposed causal discovery tool operates via measuring what we refer to as the directed topological overlap matrix (DTOM) of the genes. Elements of the DTOM are calculated as a function of the neighborhoods of the genes in the gene co-expression network. They express the strength of evidence for causal relations between a pair of genes. The more the neighborhoods of two

genes in the gene co-expression network overlap, the larger their directed topological overlap becomes. Only the genes with high directed topological overlap are suspected of having a causal relation. This type of reasoning by which causal influences are expressed in terms of the neighborhoods of the genes in the co-expression network—or equivalently, the neighborhoods of the genes in the co-expression network are given a causal interpretation—is a direct consequence of Reichenbach's common cause principle as we will show later in this paper [60, 59, 54].

We are then extracting a type of causal knowledge that lies entirely within the gene co-expression network. We, therefore, expect our causal conclusions to be more robust w.r.t sample size effects as we avoid testing for higher-order conditional independence relations. Furthermore, and as we show in our paper, this type of causal knowledge is also robust against the presence of feedback loops, unobserved con-founders, averaging effects, and measurement noise. However, these benefits come at a cost. Through DTOM, we can only work out the causal relations up to an equivalence class. This equivalence class can often be larger than the one obtained assuming the local causal Markov condition.

The extent to which we can unravel the causal structure through DTOM depends on two factors. First is the accuracy by which we can estimate the gene co-expression network. Second is the manner in which the nodes are connected in the gene co-expression network. In general, the more *v-structures* the gene co-expression network contains, the better we can work out the causal forces. At one extreme, and when the gene co-expression network is a fully connected graph, DTOM does not impart any causal knowledge. At the other extreme, it is possible to have a causal resolution on par with the equivalence classes obtained through the local causal Markov condition without requiring strict assumptions. We study in more detail the extent to which we can expect to resolve causal forces through DTOM in a genomics setting using two

real gene expression data sets. We further evaluate the practicality and the usefulness of the causal information obtained through DTOM in different tasks.

First, we consider a large-scale gene deletion study in yeast (Saccharomyces cerevisiae) [85]. This data contains the genome-wide mRNA expression levels of 6,170 genes with a pool of 160 wild-type control samples and a total of 1,479 single-gene deletions *intervention* samples. Each gene deletion sample is labeled with the name of the deleted gene. In our experiments, we examine if the causal resolution offered by DTOM enables one to reverse engineer the sample labels. In other words, we examine if we can identify the deleted gene for each sample knowing only the set of differentially expressed genes. Thus, we establish the practicality and the utility of DTOM by studying if we can employ DTOM to make out the causal gene among a set containing its effects.

In the next experiment, we contrast the muscle transcriptomes of Piedmontese and Wagyu following the work in [86]. We set out to examine if we could apply DTOM to identify the genomic DNA mutation in the myostatin (GDF8) in the Piedmontese cattle. This study is interesting to us since the regulatory perturbation of myostatin in Piedmontese cattle is post-transcriptional. More specifically, GDF8 shows no differential expression across the two breeds. Then, to identify the causal mutation in myostatin, we must capture the *functional role* of the myostatin from the gene expression data since its expression level alone does not inform us of its post-transcriptional mutation. Following the work of Hudson, Reverter, and Dalrymple, we show how their proposed approach can be enhanced using DTOM such that one can immediately identify the genomic mutation in GDF8 in Piedmontese cattle.

6.2 Methods

Topological overlap is a measure capturing gene functional relations [87, 88, 89]. The topological overlap matrix (TOM) is obtained from the gene co-expression network based on the following equation:

$$o_{ij} = \frac{l_{ij}}{min(k_i.k_j)} \tag{6.1}$$

where the topological overlap between two genes, x_i and x_j, is calculated as ratio of number of their shared neighbors in the gene co-expression network, l_{ij}, to the minimum number of neighbors of each gene, k_i and k_j. This mathematical construct was first proposed by Ravasz et al. (see [89]) and has since been employed as a metric that can capture gene functional relations. Experimental results have supported the usefulness of TOM in numerous instances [90]. In fact, the strongest argument supporting the use of TOM to evaluate gene functional relations comes from its very success in practice; we found little theoretical justification that could explain *why* TOM has performed so well in practice[1].

Surprisingly, there exists a causal structure known as the sink graph that bears a striking similarity with TOM [53, 54, 59, 60]. First, let us show how we calculate the sink graph of the gene co-expression network and its relation to the topological overlap matrix. We can then discuss its causal interpretation after reviewing some graph-theoretic notions.

[1] If our causal interpretation of TOM based on the theory of sink graphs is valid, we can perhaps attribute its success in mapping genes with similar functionality to its causal connotations.

To construct the sink graph of the gene co-expression network, we start by first calculating what we refer to as the directed topological overlap matrix, O^*:

$$O^*_{ij} = \frac{l_{ij}}{k_i}, O^*_{ji} = \frac{l_{ij}}{k_j}. \tag{6.2}$$

Now, if we imagine the genes as nodes in a graph, then the sink graph of the gene co-expression network is the directed graph in which two genes are connected by an edge, $x_i \rightarrow x_j$, if and only if $o^*_{ij} == 1$. Using Reichenbach's common cause principle, we next show that the sink graph of the gene co-expression network estimates an upper bound on the causal forces between the genes. But, before that, we need to review certain graph-theoretic notions.

Consider a domain over m variables $X = \{x_1, ..., x_m\}$. For now, we assume that we are given the marginal dependency relations in the domain, for instance the co-expression information, in the form of an undirected graph, G, where an edge $x_i - x_j \in G$ connects the two nodes x_i and x_j in G if and only if the nodes are statistically dependent. We often refer to G as the domain's marginal dependency graph. We assume that underlying the domain is a causal system giving rise to the observed marginal dependency relations. We represent this causal structure with a directed graph D: $x_i \rightarrow x_j \in D$ if and only if x_i is a *direct cause* of x_j (x_i to directly control the regulation and the production of x_j) [81]. We say x_i is a parent of x_j in D if the edge $x_i \rightarrow x_j$ connects the two nodes in D. We say x_i is an ancestor of x_j if there exists a sequence of nodes $\{x_1, ..., x_m\}$ where for any two immediate nodes in the sequence, x_k and x_{k+1}, the edge $x_k \rightarrow x_{k+1}$ is in D, and both the edges $x_i \rightarrow x_1$ and $x_m \rightarrow x_j$ are also in D. We use the notation $Anc_D(x_i)$ to denote the collection of all the ancestors of the node x_i in the directed graph D (we sometimes may remove

the subscript D if doing so does not cause confusion). We assume the convention that a node is its own ancestor.

Through Reichenbach's common cause principle, we can directly calculate the marginal dependency graph from the causal structure by connecting pairs of nodes sharing at least one ancestor. We say a causal structure, D, **generates** a marginal dependency graph, G, if G can be obtained from the causal structure in this manner and write $\Sigma(D) = G$. While the marginal dependency graph can be directly calculated from the causal structure, the causal structure is not uniquely identifiable from the marginal dependency graph. However, the following key result tells us that a marginal dependency graph can be used to estimate an upper-bound on the set of causal relations that hold in the domain. In fact, this bound is nothing but the sink graph of the marginal dependency graph: [2]

Theorem 6.2.1. *An undirected graph, G, is generated by the directed graph D, only if, D is a subgraph of the sink graph of G.*

Proof. We show if a node, a, is parent of another, b, in D, then a and b must be connected through the edge $a \rightarrow b$ in the sink graph of G. To show this, we only need to prove that every node connected to a in G is also connected to b. Consider a node c that is connected to a in G. According to the Reichenbach's common cause principle, and since c and a are connected in G, then c and a must have had a common ancestor in D. Let us refer to this common ancestor as z. (see Figure:6.1) Now, since we had assumed that a is a parent and direct cause of b in D, then z is also a common ancestor of b and c in D. This lets us conclude that every node connected to a in G,

[2]The same result can be found in [53, 54, 59, 60]. However, they only consider cases wherein the generating causal structure is acyclic and there are no feedback loops. Here we show that this results remains valid whether the underlying causal system contains cycles or not.

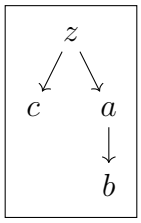


Figure 6.1: If a node, c, has a common ancestor with a, then it shares the same ancestor with b also.

must also be connected to b. Therefore, a and b must be connected through the edge $a \to b$ in the sink graph of G.

□

According to Theorem 1, the sink graph of the gene co-expression then estimates an upper bound on the causal forces between the genes. The extent to which this upper bound approximates the underlying causal system depends on the structure and the connections in the gene co-expression network itself. In general, the more *v-structures* [3] the gene co-expression network contains, the better we can work out the causal forces since the v-structures reduce the overlap between genes' neighbors [53].

Theorem 1 above holds only when we can have an exact estimate of the gene co-expression network. Given that both false positives and false negatives are expected in estimating the gene co-expression network, we, therefore, prefer to directly work with the DTOM matrix itself or construct the sink graph by applying a more liberal threshold: we connect two nodes in the sink graph through an edge $x_i \to x_j$ if the directed topological overlap from x_i to x_j exceeds a liberal threshold, $o^*_{ij} \geq \tau_2$, $0 < \tau_2 < 1$.

[3] We say the nodes a, b, and c form a **v-structure**, $a - b - c$, in a undirected graph, G, if the nodes a and b, and the nodes b and c are both connected in G, $a - b \wedge b - c \in G$, and the nodes a and c are disconnected in G.

Putting all the pieces together, to identify the causal forces among the genes through DTOM, first we establish their statistical dependence. In our paper, we use Pearson's correlation to verify statistical dependence. After constructing the correlation matrix, we apply a suitable threshold τ_1 and convert the matrix into an undirected graph representing the gene co-expression. We then calculate DTOM using eq. 6.2. Having computed the DTOM, we can either opt to work with the DTOM directly or obtain the sink graph of the gene co-expression network by applying a secondary threshold, τ_2, to the DTOM.

The threshold parameter τ_1 used in constructing the gene co-expression network is often decided through statistical means; given enough samples, this threshold could be as low as 0.1. By increasing this threshold parameter, we can further distinguish the strong statistical dependencies in the domain. As we increase τ_1, we successively construct more sparse gene co-expression networks wherein only the strongest statistical dependencies are depicted. The question here is whether calculating DTOM using a gene co-expression network that only displays the strongly correlated genes obtains an upper-bound on *strong* causal effects [4]. This would only hold if the following extension of Reichenbach's common cause principle were true[5]:

The Strong Common Cause Postulate: Two variables are strongly correlated if and only if they have a strong common cause.

The strong common cause postulate, if it were to be accurate, would transform the parameter τ_1 into a convenient dial that would help tune causal discovery through DTOM, offering a mechanism to trade recall for better precision. In our simulations,

[4]We say x_i is a strong cause of x_j if x_i is a cause of x_j and the Pearson correlation of x_i and $x - J$ is above τ_1.

[5]For the strong common cause postulate to hold it is necessary that no long chains of cause and effect exists in the domain, $x_1 \rightarrow x_2 \rightarrow ... \rightarrow x_n, n >> 1$.

we study the utility of the strong common cause postulate. Before that, we next argue why causal discovery through DTOM can be more robust to the unwanted influences of unobserved confounding variables, measurement errors, and averaging effects.

6.2.1 Unobserved Con-founders

Even with unobserved confounding variables, the sink graph remains a valid upper bound for the causal forces in the domain. We can easily verify this by consulting the proof of Theorem 1. However, unwanted confounding variables can affect the extent to which the sink graph can resolve the causal relations in a domain. For example, a global confounding effect may cause a significant overlap between the neighborhoods of the genes, inducing large cliques in the gene co-expression network. As a result, the sink graph will not resolve the causal relations among the genes participating in these cliques. The strong common cause postulate offers a means that could help amend this issue. We could construct the gene co-expression network by applying a rather conservative threshold to the gene correlation matrix. By doing so, we can filter out weak confounding effects and only consider correlations that are stronger than those produced by the unwanted confounding variables. This comes at the cost of filtering out possible weak causal relations.

6.2.2 Measurement Errors

In general, conditional independencies of a set of variables $\{X_1, X_2, ..., X_n\}$ will not hold among the variables $\{X_1^m, X_2^m, ..., X_n^m\}$, where X_i^m's are noisy measurements of X_i's: $X_i^m = f_i(X_i, \delta_i)$, f is a deterministic function and δ_i's are jointly independent [21]. Measurement noise can then threaten the legitimacy of the local causal Markov condition. However, the Reichenbach's common cause principle still remains valid in presence of measurement errors: the noisy measurements of X_i^m and X_j^m are

statistically dependent, if and only if X_i and X_j have a common cause. In other words, the marginal dependency graph of the variables $\{X_1^m, X_2^m, ..., X_n^m\}$ is identical to that of the variables $\{X_1, X_2, ..., X_n\}$; i.e X_i^m and X_j^m are dependent if and only if X_i and X_j are dependent.

6.2.3 Averaging

Gene expression experiments usually measure the average concentration of a gene in a large collection of cells rather than in a single cell. Consider a population of cells $\{C_1, ..., C_M\}$ and let X_i^m represent the expression level of gene X_i in the cell C_m. Assume that the same causal mechanisms are at work in every cell so that if X_i^m and X_j^m are independent conditional X_k^m for a given cell, then a similar independency holds among the expression levels of these genes in other cells. The important point here is that the corresponding conditional independency will not hold among the average expression levels [21]. Then averaging can undermine the legitimacy of the local Causal Markov condition. On the other hand, the marginal dependency graph of the average expression levels is identical to the marginal dependency graph of the gene expression levels in each cell. Then, the sink graph of the gene co-expression network remains unaffected whether we measure the average concentrations of the genes in a collection of cells or whether our measurements come from a single cell.

6.3 Simulations

Our main goal in this section is to study the utility of the strong common cause principle. We examine here if the threshold parameter τ_1 allows us to tune the recall and the precision of our proposed causal discovery algorithm.

We set up our simulation environment through the following steps. We first create the adjacency matrix of our causal structure: A, where $A_{ij} == 1$ if i is a parent

of j. We consider a domain consisting of 500 variables, $X = \{x_1, x_2, ..., x_{500}\}$ and we construct our adjacency matrix such that each node on average is directly connected to four others in a directed acyclic graph. We generate this adjacency matrix similar to the process described in [45]: such a simulation setting is widely used in practice to probe the performance of causal learning algorithms. Afterward, we generate eighty samples, $\{x_i^1, ...x_i^{80}\}$, in a recursive manner, starting from the root nodes of the directed acyclic graph, and down to the leaves using the following equation:

$$x_i^k = \mu_i + \sum_{x_j \in \pi_i} b_{ij}(x_j^k - \mu_j) + \mathcal{N}(0, 1/\tau_i), \tag{6.3}$$

where π_i denotes the parents of the node x_i and $\mathcal{N}(0, 1/\tau_i)$ is a noise term sampled from a normal distribution with mean of zero and variance of τ_i. We sample the parameters μ_i and τ_i from the uniform distribution $\mathcal{U}(0, 1)$, and b_{ij}'s were chosen as independent realizations of $\mathcal{U}(-1, 1)$. We then feed these samples to our causal learning machine to calculate DTOM.

To identify the causal forces in the domain, first we establish their statistical dependence. In our simulations we use Pearson's correlation to verify statistical dependence. We then apply successively increasing thresholds $\tau_1 \in \{0.3, 0.5, 0.7, 0.9\}$ to the correlation matrix and obtain successively more sparse undirected graphs representing the domain's strongest marginal dependency relations. For each of the resulting marginal dependency graphs, we then calculate DTOM using eq. 6.2. Afterward, we apply a secondary threshold, $\tau_2 \in \{0.7, 0.9\}$, to the DTOM to obtain the sink graph. By comparing the resulting sink graph to the generating causal structure, we calculate our algorithm's precision, and also, we report the number of causal relation we were able to discover at thresholds τ_1 and τ_2. We then generate a new adjacency matrix and repeat this process all over. The results that we report

next are the averages obtained by sampling over 5000 randomly generated adjacency matrices.

Figure 6.2 plots the average precision and the average number of discovered causal relations as a function of the parameters τ_1 and τ_2. We see that increasing the thresholds τ_1 and τ_2 value has lead to uncovering a lower number of causal relations, however, the causal relations that are extracted are more precise. This verifies the utility of the strong common cause postulate and the usefulness of DTOM in resolving causal relation at least in the experimental setup we just described. The utility of DTOM in resolving causal in genomics setting remains to be seen.

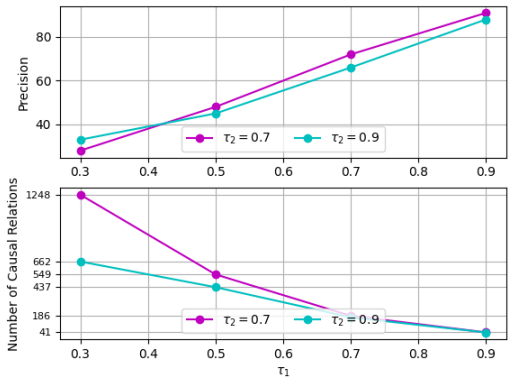


Figure 6.2: The average precision of DTOM and the average number of discovered causal relations as a function of the parameters τ_1 and τ_2 in a domain consisting of 500 nodes when given 80 samples.

6.4 A Large Scale Gene Deletion Study in Yeast

For our first experiment we consider a large scale gene deletion study in yeast (Saccharomyces cerevisiae). This data contains the genome wide mRNA expression

levels of 6,170 genes with a pool of 160 wild-type control samples and a total of 1,479 single gene deletion *intervention* samples.

Consider a single gene deletion sample wherein the gene X is deleted. We can use this sample to identify the set of genes, Y, that are differentially expressed following the deletion of the gene X and thus extract a set of true positive causal relations, $X \rightarrow Y$. We can then deploy these true positive causal relations as a measuring stick to validate the use of directed topological overlap matrix in causal discovery. However, these true positive causal relations can be very sensitive to the criteria by which the differential expression of a gene is established. To obtain a set of robust true positives, we employ a conservative criteria combining the notion of strong intervention effect of [91] and the restrictions on differential expression as specified in [85]. In summary, we establish a true positive causal influence from gene X to gene Y, $X \rightarrow Y$ if the following conditions hold.

1. The sample where gene X is deleted is must be a successful deletion. A sample where gene X is deleted is considered a successful deletion if the following conditions hold.
   (a) The sample where the gene X is deleted is labeled as a responsive mutation according to the criteria set by [2],
   (b) the expression level (the M-ratio or the log-2 fold change) of the gene X attains its lowest value in the sample where the gene X is deleted,
   (c) the expression level (the M-ratio or the log-2 fold change) of the gene X is below -1.7 in the sample where gene X is deleted,
   (d) the expression level of the gene X is statistically significantly different at p-value of $1e-6$ in the sample where gene X is deleted when compared to its normal expression levels in the wild-type control samples.

2. We extract a true positive causal relation in the sample where gene X is deleted if we are able to find any gene Y that satisfies the conditions below:
   (a) the expression level (the M-ratio or the log-2 fold change) of the gene Y attains its lowest value or its highest value in the sample where the gene X is deleted,
   (b) the expression level (the M-ratio or the log-2 fold change) of the gene Y is below -1.7 or above 1.7 in the sample where gene X is deleted,
   (c) the expression level of the gene Y is statistically significantly different at p-value of $1e-6$ in the sample where gene X is deleted when compared to its normal expression levels in the wild-type control samples.

Furthermore, for the sake of uniformity, we limited our analysis to those samples of BY4742 strain where the extraction protocol was labeled yeast HTP RNA isolation for robot v2.0 and were grown using the Tecan platereader setup.

Only 365 of the 1,479 single gene deletion samples met the criteria 1.a-1.d specified above. Out of the 365 successful deletions, in only 80 instances we were able to identify a true positive causal relation (criteria 2.a - 2.c were satisfied by at least one gene Y, $Y \neq X$). Per each mutation instance where at least one true positive causal relation was identified, we found on average just below 9 causal relations, giving rise to a total of 704 true positive causal relations.

After establishing the set of true positive causal relations, we estimated the gene co-expression network over the set of 784 genes that participated in the 704 true positive causal relations. To obtain an accurate estimate of the gene co-expression network, we concatenated the wild-type control samples with all the intervention samples, regardless of whether the intervention was deemed successful or not. Afterwards, we subtracted the first SVD component from the data matrix thus constructed to remove any slow-growth effects. We then calculated the Pearson correlation coefficient for

each gene pair. The 0.99 quantile of the co-expression values was 0.463. Permuting the expression levels for each gene, we then estimated a *null* coexpression distribution. The 1-1e-8 quantile of this distribution was 0.4571. Interestingly, both the 0.99 quantile of the co-expression values and the estimated 1-1e-8 quantile of the null co-expression values were significantly higher than the critical threshold one would obtain using Fisher's Z-transformation at p-value of 1e-8. The critical threshold obtained using Fisher's transformation was 0.17. Given our results in the Simulations section, we decided to construct the gene co-expression network by thresholding the co-expression matrix at the more conservative threshold of 0.4571.

Having the gene co-expression network, we then calculated its directed topological overlap matrix. For each deleted gene in the 80 successful gene deletion records, we extracted the set of genes, Y, where the directed topological overlap from the deleted gene to Y was above a predefined threshold, τ_2. Whenever the directed topological overlap matrix from the deleted gene to Y crossed the predefined threshold, we predicted the deleted gene to be a causal regulator of Y. Selecting a higher threshold value, τ_2, would lead to uncovering a lower number of causal relations. However, we except to have more confidence in the causal relations we extract. We lowered the threshold value τ_2 from 0.95 down to 0.70 in intervals. At each step, we recorded the count of causal relations we discovered, and also, the precision of these discoveries as validated against the true positive causal relations we estimated previously. Figure 6.3 shows the precision of DTOM in uncovering causal relations against the number of causal relations we extract at each level of τ_2.

The precision of the discovered causal relations as a function of both parameters τ_1 and τ_2 is shown in Figure 6.4. This results suggests that constructing the co-expression network using the conservative threshold $\tau_1 = 0.4571$ obtains a significantly more precise set of causal relations compared to the more liberal threshold obtained

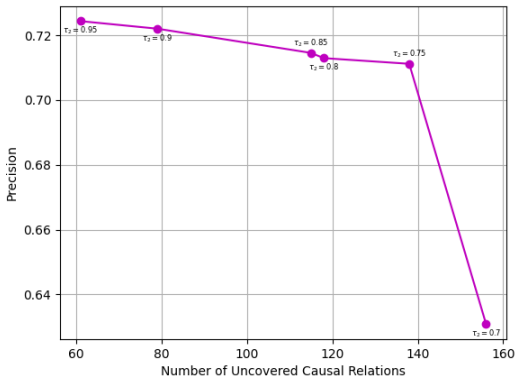


Figure 6.3: We lowered the threshold parameter τ_2 from 0.95 down to 0.70 in intervals. At each step, we recorded the count of causal relations we discovered (X-axis), and also, the precision of these discoveries as validated against the true positive causal relations we estimated previously (Y-axis). The point at the top left corner corresponds to $\tau_2 = 0.95$ and the point at the bottom right corner correspond to $\tau = 0.7$.

by using Fisher's Z-transformation. This confirms the utility of the strong common cause postulate in genomics settings. However, we see that increasing the threshold τ_1 beyond a certain value has diminishing returns, and can even lower the precision.

Our results suggest that DTOM, using conservative thresholds $\tau_1 = 0.4571$ and $\tau_2 = 0.7$, can obtain a fairly precise picture (precision of 0.7) of the causal relations, although a rather incomplete picture (138 causal relations identified distributed over 80 genes). The important question here is whether such a portrayal of the underlying causal system is of any use. To emphasize the practicality of DTOM, we consider whether causal knowledge imparted by DTOM allows us to reverse engineer the sample labels, and identify the deleted gene for each gene deletion sample. This is rather an elementary task if we were to know the fold changes of the genes in the deletion sample; the deleted gene is almost always the gene whose expression is the lowest in the deletion sample. To make this task more challenging, we hide the fold change

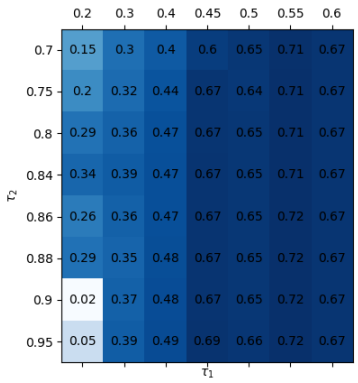


Figure 6.4: Precision of DTOM in resolving causal relations as a function of the threshold parameters τ_1 and τ_2.

numbers from our algorithm, and for each gene deletion sample we only inform our algorithm of the set of genes that were measured to be differentially expressed.

To obtain the set of differentially expressed genes in a gene deletion sample, we search for genes whose expression level (the M-ratio or the log-2 fold change) is statistically significantly different at p-value of 1e-6 when compared to their normal expression levels in the wild-type control samples. We further require the expression level (the M-ratio or the log-2 fold change) to be above a predefined threshold, $\tau_{FC} \in \{0.7, 0.9, 1.1, 1.3, 1.7\}$. Clearly, the lower the selected threshold on fold change is, the higher the count of differentially expressed genes grows, and the harder it becomes to pick out the deleted gene. Basically, in this experiment we examine if we can employ DTOM to distinguish the causal gene among its *effects*[6], i.e. the list of differentially expressed genes.

Like before, for the sake of uniformity, we limited our analysis to those samples of BY4742 strain where the extraction protocol was labeled yeast HTP RNA isolation for

[6]We emphasize here that since we are relaxing the conservative criteria 1.a-1.d and 2.a-2.b, the deleted gene is no longer guaranteed to be the causal regulator of the differentially expressed genes extracted thus.

robot v2.0 and were grown using the Tecan platereader setup. We further eliminated all mutations that are labeled as a nonresponsive mutation according to the criteria set by [2]. For each of the remaining deletion samples (a total of 367) we identified the set of differentially expressed genes: genes whose expression level (the M-ratio or the log-2 fold change) were statistically significantly different at p-value of 1e-6 when compared to their normal expression levels in the wild-type control samples, and whose log-2 fold change was be above a predefined threshold, $\tau_{FC} \in \{0.7, 0.9, 1.1, 1.3, 1.7\}$. At fold change level of 1.7, we found a total of 1073 genes that were differentially expressed in at least one of the gene deletion samples. At fold change level of 0.7, we found 3547 genes that were differentially expressed in at least one of the gene deletion samples.

Afterward, we concatenated the wild-type control samples with all the intervention samples, regardless of whether the intervention was deemed responsive or not. After subtracting the first SVD component, we calculated the Pearson correlation coefficient for each gene pair in the set of differentially expressed genes. We then constructed the co-expression network by thresholding the gene co-expression matrix, connecting only those genes whose correlation exceeded the 0.4571 threshold we had calculated earlier. Having the gene co-expression network, we then calculated the DTOM using eq. 6.2. This time, instead of thresholding the DTOM, we opted to work with the DTOM values directly.

Now, consider a specific gene deletion sample where the gene x_k was been deleted. Let us assume that that our selected fold change level was 1.7, and that the set of differentially expressed genes in this sample were the genes $DEG(x_k) = \{x_k, y_1, ..., y_m\}$. We represent this set of differentially expressed genes with a vector of size 1073; 1073 is the size of the set of all differentially expressed genes in all deletion samples at fold change of 1.7. We set this vector equal to zero everywhere except the indices corresponding to the genes in the set $DEG(x_k)$ where we set it equal to one. Let us

denote this vector with $\vec{D}(x_k)$. To pick out the deleted gene in the set $DEG(x_k)$, we assigned a score to each of the genes in the set that measures the overlap between a gene's directed topological overlap vector and the vector $\vec{D}(x_k)$. To be more specific, we assigned to each gene a score obtained using the following equation:

$$S(y_i) = \frac{< DTOM[y_i, :], \vec{D}(x_k) >}{||DTOM[y_i, :]||_1}, \tag{6.4}$$

where $< ., . >$ and $||.||_1$ denote inner product and L1 norm. $DTOM[y_i, :]$ denotes the row in the DTOM corresponding to the gene y_i. We then ranked the genes in $DEG(x_k)$ according to the score assigned to them by the formula above. Figure 6.5 shows the average rank of the deleted gene against the number of candidate genes as scored by DTOM.

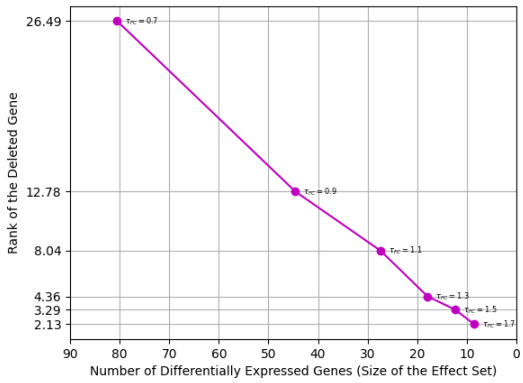


Figure 6.5: The average rank of the deleted gene in a gene deletion sample as scored by the metric in Eq. 6.4 against the average number of differentially expressed genes. Differentially expressed genes are those genes (1) whose expression levels are statistically significantly different at p-value of 1e-6 when compared to normal expression levels in the wild-type control samples, and (2) whose expression levels are above a predefined threshold. We have marked the thresholds we chose in the figure.

At fold change level of 1.7 we found on average 8.55 differentially expressed genes per each deletion sample. The average rank of the deleted gene among the set of the differentially expressed genes across the 367 deletions samples was 2.13.

6.5 Uncovering Post-transcriptional Myostatin Mutation in Piedmontese Cattle

A genomic DNA mutation in the myostatin (GDF8) mRNA transcript of the Piedmontese cattle accelerates muscle growth causing the double-muscling the Piedmotese cattles are known for. This regulatory perturbation of myostatin in the Piedmontese cattles is entirely post-transcriptional. Due to the post-transcriptional nature of this mutation, the myostatin shows no differential expression in the Piedmotese cattle when compared to the non-mutant Wagyu cross. Our goal in this experiment is to investigate if it is possible to identify the mutation in myostatin by contrasting the muscle transcriptomes of the Piedmontese and Wagyu crosses. For this, it is necessary that we somehow capture the *functional role* of the myostatin from gene expression data since its expression level alone does not inform of its post-transcriptional mutation.

To measure the extent to which the functional role of a regulator changes in the Piedmontese cattle, Hudson, Reverter, and Dalrymple propose the following metric [86]:

$$R_f = \sum_{i=1}^{n} \mu_i |\mu_i^P - \mu_i^W| (r_{if}^P - r_{if}^W)^2, \tag{6.5}$$

where the score assigned to regulator f, R_f, is calculated as a weighted average of the difference in the co-expression of the said regulator and the set of genes differentially expressed across the two breeds, $r_{if}^P - r_{if}^W$. The weights in this average correspond to

the expression level of the differentially expressed gene μ_i, further scaled by the size of differential expression $\mu_i^P - \mu_i^W$.

Using bovine oligonucleotide microarray, Hudson, Reverter, and Dalrymple obtained the genome wide expression levels of 11,057 genes in two crosses and across 10 developmental time points [86]. Out of the 11,057 genes, they then extracted a list of 920 candidate regulators. After identifying 85 differentially expressed genes, they rank the 920 candidate regulators using eq. 6.5. They find GDF8 to be the fourth most *positively differentially wired*[7] regulator. Figure 6.6 succinctly captures the positive differential wiring of GDF8. It is but natural to ask ourselves if we can improve upon their score using DTOM.

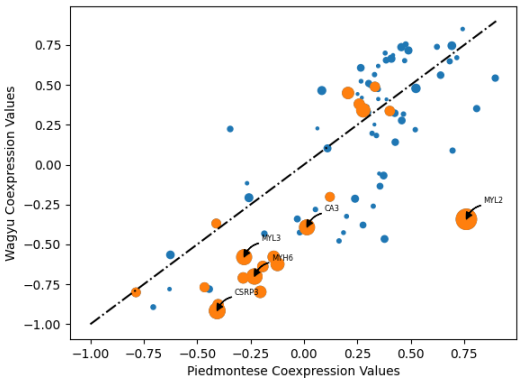


Figure 6.6: The 85 circles mark the co-expression values of GDF8 and the 85 differentially expressed genes; we used the Pearson correlation coefficient in measuring the co-expression of the genes. The size of a circle is drawn proportional to the expression level of its corresponding differentially expressed gene multiplied by the size of its differential expression. We have indicated the 20 largest circles with the orange color. In the Wagyu cattle, GDF8 often appears as a negative regulator of the differentially expressed genes. In the Piedmontese cattle, however, the GDF8's functional role as a negative regulator is significantly less pronounced.

[7]Hudson, Reverter, and Dalrymple refer to a regulator f as being positively differentially wired to some gene, i, if $r_{if}^W < r_{if}^P$.

To get started, first we opted to calculate DTOM directly using the co-expression values and without defining gene neighbors through significance testing: it is challenging to distinguish the statistically significant co-expressions given the limited number of samples. We decided to estimate DTOM using the following measure (eq 6.6):

$$o^*_{fi} = \frac{\langle |r_{f.}|, |r_{.i}| \rangle}{||r_{f.}||_1}, \tag{6.6}$$

where r_f is a vector delineating the co-expression values of the regulator f, the operator $||.||_1$ denotes the L1 norm, and the operator $|.|$ denotes the element-wise absolute value of a vector. The proposed DTOM estimator is comparable to the generalized TOM (GTOM) measure proposed by Zhang and Horvath. In fact, the only difference is that here we take the absolute value of the co-expressions whereas in GTOM the co-expression values are inserted as they are.

Figure 6.7 shows the directed topological overlap from GDF8 to the 85 differentially expressed genes. The 85 circles mark the 85 differentially expressed genes where they are drawn proportional to the expression level of the differentially expressed gene multiplied by the size of its differential expression. We have indicated the 20 largest circles with the orange color. We see a large concentration of circles to the top right corner of the figure. This endorses GDF8's causal role in regulating the expression of these genes. Furthermore, we see that GDF8's causal role is more pronounced in the Wagyu cross since most of the circles appear above the diagonal line drawn.

Now, if we were to take the same scoring metric as 6.4 and weigh each co-expression further by the DTOM of the regulator to the differentially expressed gene, see eq. 6.7 below, then GDF8 pops as the number one highest scoring gene.

$$R_f = \sum_{i=1}^{n} \mu_i(\mu_i^P - \mu_i^W)(r_{if}^P o_{if}^{*P} - r_{if}^W c_{if}^{*W}). \tag{6.7}$$

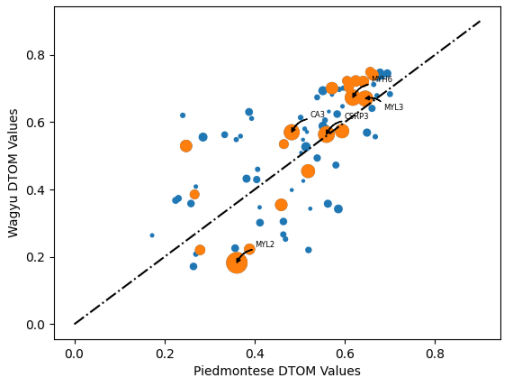


Figure 6.7: The directed topological overlap from GDF8 to the 85 differentially expressed genes in the Piedmontese cattle (X-axis) and the Wagyu cattle (Y-axis).

Hudson, Reverter, and Dalrymple were able to score gene GDF8 as the highest scoring gene only when they combined two different metrics. Here, however, using DTOM information we were able to immediately highlight GDF8 as the highest scoring gene. The top five ranking genes according to our proposed metric are GDF8, MYOD1 (master regulator of muscle cell differentiation), CSRP1 (development and cellular differentiation), SUV39H2 (represses promoter activity), and SMYD1 (transcription corepressor).

6.6 Conclusions

The theoretical assumptions that underlie the local causal Markov condition are often not met in practice. This disconnect between theory and practice is especially marked in genomics. In genomics, the unwanted presence of measurement errors, averaging effects, and feedback loops significantly undermine the legitimacy of the local causal Markov condition. In this paper, replacing the local causal Markov condition with Reichenbach's common cause principle, we presented DTOM, a flexible approach to causal discovery in genomics settings. Briefly, the elements of DTOM are calculated

as a function of the neighborhoods of the genes in the gene co-expression network. The more the neighborhoods of two genes overlap, the larger their directed topological overlap becomes. Only those pairs of genes with high directed topological overlap are then suspected of having a causal relation.

We studied the practicality and the utility of DTOM for discovering causal relations using two real gene expression data-sets. First, we considered a large-scale gene deletion study in yeast. We showed that through DTOM we are able to reverse engineer the sample labels with desirable precision, distinguishing the deleted gene in a sample knowing only the set of differentially expressed genes. In the next study, we contrasted the muscle transcriptomes of Piedmontese and Wagyu crosses. While the regulatory perturbation of myostatin in the Piedmontese cattle is entirely post-transcriptional, we were still able to immediately identify the causal mutation in Myostatin in the Piedmontese cattle using DTOM.

REFERENCES

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

[2] F. V. Jensen, *An introduction to Bayesian networks*. UCL press London, 1996, vol. 210.

[3] P. Spirtes, C. N. Glymour, and R. Scheines, *Causality from probability*. Carnegie-Mellon University, Laboratory for Computational Linguistics, 1989, vol. 112.

[4] N. Friedman, "Inferring cellular networks using probabilistic graphical models," *Science*, vol. 303, no. 5659, pp. 799–805, 2004.

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

[6] G. F. Cooper and E. Herskovits, "A bayesian method for the induction of probabilistic networks from data," *Machine learning*, vol. 9, no. 4, pp. 309–347, 1992.

[7] D. Heckerman, D. Geiger, and D. M. Chickering, "Learning bayesian networks: The combination of knowledge and statistical data," *Machine learning*, vol. 20, no. 3, pp. 197–243, 1995.

[8] D. M. Chickering and C. Meek, "Finding optimal bayesian networks," in *Proceedings of the Eighteenth conference on Uncertainty in artificial intelligence*. Morgan Kaufmann Publishers Inc., 2002, pp. 94–102.

[9] D. M. Chickering, "Optimal structure identification with greedy search," *Journal of machine learning research*, vol. 3, no. Nov, pp. 507–554, 2002.

[10] T. Silander and P. Myllymaki, "A simple approach for finding the globally optimal bayesian network structure," *arXiv preprint arXiv:1206.6875*, 2012.

[11] M. Teyssier and D. Koller, "Ordering-based search: A simple and effective algorithm for learning bayesian networks," *arXiv preprint arXiv:1207.1429*, 2012.

[12] J. M. Ortega, *Numerical analysis: a second course*. SIAM, 1990.

[13] A. Miller, *Subset selection in regression*. Chapman and Hall/CRC, 2002.

[14] D. Smith and J. Bremner, "All possible subset regressions using the qr decomposition," *Computational Statistics & Data Analysis*, vol. 7, no. 3, pp. 217–235, 1989.

[15] P. Yanev, P. Foschi, and E. J. Kontoghiorghes, "Algorithms for computing the qr decomposition of a set of matrices with common columns," *Algorithmica*, vol. 39, no. 1, pp. 83–93, 2004.

[16] C. Gatu, P. I. Yanev, and E. J. Kontoghiorghes, "A graph approach to generate all possible regression submodels," *Computational Statistics & Data Analysis*, vol. 52, no. 2, pp. 799–815, 2007.

[17] C. Gatu and E. J. Kontoghiorghes, "Parallel algorithms for computing all possible subset regression models using the qr decomposition," *Parallel Computing*, vol. 29, no. 4, pp. 505–521, 2003.

[18] M. Clarke, "Algorithm as 163: A givens algorithm for moving from one linear model to another without going back to the data," *Journal of the Royal Statistical Society. Series C (Applied Statistics)*, vol. 30, no. 2, pp. 198–203, 1981.

[19] J. Pearl, *Probabilistic reasoning in intelligent systems: networks of plausible inference*. Elsevier, 2014.

[20] R. G. Cowell, P. Dawid, S. L. Lauritzen, and D. J. Spiegelhalter, *Probabilistic networks and expert systems: Exact computational methods for Bayesian networks*. Springer Science & Business Media, 2006.

[21] P. Spirtes, C. Glymour, R. Scheines, S. Kauffman, V. Aimale, and F. Wimberly, "Constructing bayesian network models of gene expression networks from microarray data," 2000.

[22] P. Spirtes, C. N. Glymour, R. Scheines, D. Heckerman, C. Meek, G. Cooper, and T. Richardson, *Causation, prediction, and search*. MIT press, 2000.

[23] M. Scutari, "Learning bayesian networks with the bnlearn r package," *arXiv preprint arXiv:0908.3817*, 2009.

[24] T. Silander, T. Roos, P. Kontkanen, and P. Myllymäki, "Factorized normalized maximum likelihood criterion for learning bayesian network structures," in *Proceedings of the 4th European Workshop on Probabilistic Graphical Models*, 2008.

[25] R. R. Bouckaert, "Probabilistic network construction using the minimum description length principle," in *European conference on symbolic and quantitative approaches to reasoning and uncertainty*. Springer, 1993, pp. 41–48.

[26] R. Daly, Q. Shen, and S. Aitken, "Learning bayesian networks: approaches and issues," *The knowledge engineering review*, vol. 26, no. 2, pp. 99–157, 2011.

[27] N. Friedman, M. Goldszmidt, *et al.*, "Discretizing continuous attributes while learning bayesian networks," in *ICML*, 1996, pp. 157–165.

[28] H. Akaike, "Information theory and an extension of the maximum likelihood principle," in *Selected Papers of Hirotugu Akaike*. Springer, 1998, pp. 199–213.

[29] G. Schwarz *et al.*, "Estimating the dimension of a model," *The annals of statistics*, vol. 6, no. 2, pp. 461–464, 1978.

[30] D. Geiger and D. Heckerman, "Learning gaussian networks," in *Uncertainty Proceedings 1994*. Elsevier, 1994, pp. 235–243.

[31] A. Barron, J. Rissanen, and B. Yu, "The minimum description length principle in coding and modeling," *IEEE Transactions on Information Theory*, vol. 44, no. 6, pp. 2743–2760, 1998.

[32] M. H. Hansen and B. Yu, "Model selection and the principle of minimum description length," *Journal of the American Statistical Association*, vol. 96, no. 454, pp. 746–774, 2001.

[33] P. D. Grünwald, *The minimum description length principle*. MIT press, 2007.

[34] R. D. Shachter and C. R. Kenley, "Gaussian influence diagrams," *Management science*, vol. 35, no. 5, pp. 527–550, 1989.

[35] R. Jorma, *Stochastic complexity in statistical inquiry*. World scientific, 1998, vol. 15.

[36] J. Rissanen, "Stochastic complexity and modeling," *The annals of statistics*, pp. 1080–1100, 1986.

[37] W. Lam and F. Bacchus, "Learning bayesian belief networks: An approach based on the mdl principle," *Computational intelligence*, vol. 10, no. 3, pp. 269–293, 1994.

[38] J. Rissanen, "A universal prior for integers and estimation by minimum description length," *The Annals of statistics*, pp. 416–431, 1983.

[39] ——, "Mdl denoising," *IEEE Transactions on Information Theory*, vol. 46, no. 7, pp. 2537–2543, 2000.

[40] K. Miyaguchi, "Normalized maximum likelihood with luckiness for multivariate normal distributions," *arXiv preprint arXiv:1708.01861*, 2017.

[41] T. Roos, "Mdl regression and denoising," *Unpublished manuscript*, 2004.

[42] J. Kuipers, G. Moffa, D. Heckerman, *et al.*, "Addendum on the scoring of gaussian directed acyclic graphical models," *The Annals of Statistics*, vol. 42, no. 4, pp. 1689–1691, 2014.

[43] I. Tsamardinos, L. E. Brown, and C. F. Aliferis, "The max-min hill-climbing bayesian network structure learning algorithm," *Machine learning*, vol. 65, no. 1, pp. 31–78, 2006.

[44] M. de Jongh and M. J. Druzdzel, "A comparison of structural distance measures for causal bayesian network models," *Recent Advances in Intelligent Information Systems, Challenging Problems of Science, Computer Science series*, pp. 443–456, 2009.

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

[46] S. Shimizu, P. O. Hoyer, A. Hyvärinen, and A. Kerminen, "A linear non-gaussian acyclic model for causal discovery," *Journal of Machine Learning Research*, vol. 7, no. Oct, pp. 2003–2030, 2006.

[47] H. Dai, K. B. Korb, C. S. Wallace, and X. Wu, "A study of causal discovery, with weak links and small samples," in *IJCAI*. Citeseer, 1997, pp. 1304–1309.

[48] Z. Liu, B. Malone, and C. Yuan, "Empirical evaluation of scoring functions for bayesian network model selection," in *BMC bioinformatics*, vol. 13, no. S15. Springer, 2012, p. S14.

[49] N. Friedman, M. Linial, I. Nachman, and D. Pe'er, "Using bayesian networks to analyze expression data," *Journal of computational biology*, vol. 7, no. 3-4, pp. 601–620, 2000.

[50] L. Badea, "Inferring large gene networks from microarray data: a constraint-based approach," in *IJCAI-2003 Workshop on Learning Graphical Models for Computational Genomics*, vol. 72. Citeseer, 2003.

[51] B. Liu, J. Li, A. Tsykin, L. Liu, A. B. Gaur, and G. J. Goodall, "Exploring complex mirna-mrna interactions with bayesian networks by splitting-averaging strategy," *BMC bioinformatics*, vol. 10, no. 1, p. 408, 2009.

[52] D. Colombo and M. H. Maathuis, "Order-independent constraint-based causal structure learning," *The Journal of Machine Learning Research*, vol. 15, no. 1, pp. 3741–3782, 2014.

[53] J. Textor, A. Idelberger, and M. Liśkiewicz, "Learning from pairwise marginal independencies," *arXiv preprint arXiv:1508.00280*, 2015.

[54] J. Pearl and N. Wermuth, "When can association graphs admit a causal interpretation?" in *Selecting Models from Data*. Springer, 1994, pp. 205–214.

[55] M. Tsagris, Z. Papadovasilakis, K. Lakiotaki, and I. Tsamardinos, "Efficient feature selection on gene expression data: Which algorithm to use?" *BioRxiv*, p. 431734, 2018.

[56] S. Weichwald, T. Meyer, O. Özdenizci, B. Schölkopf, T. Ball, and M. Grosse-Wentrup, "Causal interpretation rules for encoding and decoding models in neuroimaging," *Neuroimage*, vol. 110, pp. 48–59, 2015.

[57] S. P. Pantazatos, Y. Huang, G. B. Rosoklija, A. J. Dwork, V. Arango, and J. J. Mann, "Whole-transcriptome brain expression and exon-usage profiling in major depression and suicide: evidence for altered glial, endothelial and atpase activity," *Molecular psychiatry*, vol. 22, no. 5, pp. 760–773, 2017.

[58] A. Lozano, G. Swirszcz, and N. Abe, "Group orthogonal matching pursuit for logistic regression," in *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, 2011, pp. 452–460.

[59] F. R. McMorris and G. Myers, "Some uniqueness results for upper bound," *Discrete Mathematics*, vol. 44, no. 3, pp. 321–323, 1983.

[60] F. McMorris and T. Zaslavsky, "Bound graphs of a partially ordered set," *J. Combin. Inform. System Sci*, vol. 7, no. 2, pp. 134–138, 1982.

[61] M. I. Love, W. Huber, and S. Anders, "Moderated estimation of fold change and dispersion for rna-seq data with deseq2," *Genome biology*, vol. 15, no. 12, p. 550, 2014.

[62] D. Kostka and R. Spang, "Finding disease specific alterations in the co-expression of genes," *Bioinformatics*, vol. 20, no. suppl_1, pp. i194–i199, 2004.

[63] M. Watson, "Coxpress: differential co-expression in gene expression data," *BMC bioinformatics*, vol. 7, no. 1, pp. 1–12, 2006.

[64] N. J. Hudson, B. P. Dalrymple, and A. Reverter, "Beyond differential expression: the quest for causal mutations and effector molecules," *BMC genomics*, vol. 13, no. 1, pp. 1–16, 2012.

[65] C. Gaiteri, Y. Ding, B. French, G. C. Tseng, and E. Sibille, "Beyond modules and hubs: the potential of gene coexpression networks for investigating molecular mechanisms of complex brain disorders," *Genes, brain and behavior*, vol. 13, no. 1, pp. 13–24, 2014.

[66] D. D. Bhuva, J. Cursons, G. K. Smyth, and M. J. Davis, "Differential co-expression-based detection of conditional relationships in transcriptional data: comparative analysis and application to breast cancer," *Genome biology*, vol. 20, no. 1, pp. 1–21, 2019.

[67] M. Bockmayr, F. Klauschen, B. Győrffy, C. Denkert, and J. Budczies, "New network topology approaches reveal differential correlation patterns in breast cancer," *BMC systems biology*, vol. 7, no. 1, pp. 1–14, 2013.

[68] D. Amar, H. Safer, and R. Shamir, "Dissection of regulatory networks that are altered in disease via differential co-expression," *PLoS Comput Biol*, vol. 9, no. 3, p. e1002955, 2013.

[69] A. de la Fuente, "From differential expressionto differential networking–identification of dysfunctional regulatory networks in diseases," *Trends in genetics*, vol. 26, no. 7, pp. 326–333, 2010.

[70] B. Shipley, *Cause and correlation in biology: a user's guide to path analysis, structural equations and causal inference with R*. Cambridge University Press, 2016.

[71] C. Gaiteri, J.-P. Guilloux, D. A. Lewis, and E. Sibille, "Altered gene synchrony suggests a combined hormone-mediated dysregulated state in major depression," *PloS one*, vol. 5, no. 4, p. e9970, 2010.

[72] L. D. Thomas, D. Vyshenska, N. Shulzhenko, A. Yambartsev, and A. Morgun, "Differentially correlated genes in co-expression networks control phenotype transitions," *F1000Research*, vol. 5, 2016.

[73] E. Gov and K. Y. Arga, "Differential co-expression analysis reveals a novel prognostic gene module in ovarian cancer," *Scientific reports*, vol. 7, no. 1, pp. 1–10, 2017.

[74] M. Farahbod and P. Pavlidis, "Differential coexpression in human tissues and the confounding effect of mean expression levels," *Bioinformatics*, vol. 35, no. 1, pp. 55–61, 2019.

[75] J. Ihmels, S. Bergmann, J. Berman, and N. Barkai, "Comparative gene expression analysis by a differential clustering approach: application to the candida albicans transcription program," *PLoS Genet*, vol. 1, no. 3, p. e39, 2005.

[76] K. D. Hoover, "The logic of causal inference: Econometrics and the conditional analysis of causation," *Economics & Philosophy*, vol. 6, no. 2, pp. 207–234, 1990.

[77] J. Tian and J. Pearl, "Causal discovery from changes," in *Proceedings of the Seventeenth conference on Uncertainty in artificial intelligence*, 2001, pp. 512–521.

[78] J. Peters, P. Bühlmann, and N. Meinshausen, "Causal inference by using invariant prediction: identification and confidence intervals," *Journal of the Royal Statistical Society. Series B (Statistical Methodology)*, pp. 947–1012, 2016.

[79] K. Zhang, B. Huang, J. Zhang, C. Glymour, and B. Schölkopf, "Causal discovery from nonstationary/heterogeneous data: Skeleton estimation and orientation determination," in *IJCAI: Proceedings of the Conference*, vol. 2017. NIH Public Access, 2017, p. 1347.

[80] J. Peters, D. Janzing, and B. Schölkopf, *Elements of causal inference: foundations and learning algorithms*. The MIT Press, 2017.

[81] P. Sprites, C. Glymour, and R. Scheines, "Causation, prediction and search (first, online ed.)," 1993.

[82] M. Wienöbst and M. Liskiewicz, "Recovering causal structures from low-order conditional independencies," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34, no. 06, 2020, pp. 10 302–10 309.

[83] C. Glymour, K. Zhang, and P. Spirtes, "Review of causal discovery methods based on graphical models," *Frontiers in genetics*, vol. 10, p. 524, 2019.

[84] J. Pearl and T. Verma, *The logic of representing dependencies by directed graphs*. University of California (Los Angeles). Computer Science Department, 1987.

[85] P. Kemmeren, K. Sameith, L. A. Van De Pasch, J. J. Benschop, T. L. Lenstra, T. Margaritis, E. ODuibhir, E. Apweiler, S. van Wageningen, C. W. Ko, *et al.*, "Large-scale genetic perturbations reveal regulatory networks and an abundance of gene-specific repressors," *Cell*, vol. 157, no. 3, pp. 740–752, 2014.

[86] N. J. Hudson, A. Reverter, and B. P. Dalrymple, "A differential wiring analysis of expression data correctly identifies the gene containing the causal mutation," *PLoS computational biology*, vol. 5, no. 5, p. e1000382, 2009.

[87] B. Zhang and S. Horvath, "A general framework for weighted gene co-expression network analysis," *Statistical applications in genetics and molecular biology*, vol. 4, no. 1, 2005.

[88] A. M. Yip and S. Horvath, "Gene network interconnectedness and the generalized topological overlap measure," *BMC bioinformatics*, vol. 8, no. 1, pp. 1–14, 2007.

[89] E. Ravasz, A. L. Somera, D. A. Mongru, Z. N. Oltvai, and A.-L. Barabási, "Hierarchical organization of modularity in metabolic networks," *science*, vol. 297, no. 5586, pp. 1551–1555, 2002.

[90] P. Langfelder and S. Horvath, "Wgcna: an r package for weighted correlation network analysis," *BMC bioinformatics*, vol. 9, no. 1, pp. 1–13, 2008.

[91] N. Meinshausen, A. Hauser, J. M. Mooij, J. Peters, P. Versteeg, and P. Bühlmann, "Methods for causal inference from gene perturbation experiments and validation," *Proceedings of the National Academy of Sciences*, vol. 113, no. 27, pp. 7361–7368, 2016.