AN INTEGER PROGRAMMING MODEL FOR DESIGNING CAUSAL NETWORKS

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF ENGINEERING AND SCIENCE
OF BILKENT UNIVERSITY
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR
THE DEGREE OF
MASTER OF SCIENCE
IN
INDUSTRIAL ENGINEERING

By
Ali İlhan Haliloğlu
August 2024

AN INTEGER PROGRAMMING MODEL FOR DESIGNING CAUSAL NETWORKS
By Ali İlhan Haliloğlu
August 2024

We certify that we have read this thesis and that in our opinion it is fully adequate, in scope and in quality, as a thesis for the degree of Master of Science.

Oya Karaşan(Advisor)

Özlem Karsu(Co-Advisor)

Nur Kaynar

Mustafa Kemal Tural

Approved for the Graduate School of Engineering and Science:

Orhan Arıkan
Director of the Graduate School

ABSTRACT

AN INTEGER PROGRAMMING MODEL FOR DESIGNING CAUSAL NETWORKS

Ali İlhan Haliloğlu
M.S. in Industrial Engineering
Advisor: Oya Karaşan
Co-Advisor: Özlem Karsu
August 2024

We propose a novel mixed integer programming formulation for the design of causal discovery networks. The model takes a set of rules that indicate statistical dependency relations between features of a given dataset, the so-called d-connection and d-separation relations, and aims to fit a casual network with minimum (weighted) violations. Allowing feedback cycles and latent confounders, our formulation stands out from most of the existing attempts in the literature. Although our model can work as an unsupervised machine learning model, it possesses the necessary flexibility for the decision-maker to enter known causal relations. The performance of our model is tested with several synthetic datasets.

Keywords: Causal Discovery, Integer Programming, Network Modeling.

ÖZET

NEDENSEL AĞLARIN TASARIMI İÇİN BİR TAMSAYILI PROGRAMLAMA MODELİ

Ali İlhan Haliloğlu
Endüstri Mühendisliği, Yüksek Lisans
Tez Danışmanı: Oya Karaşan
İkinci Tez Danışmanı: Özlem Karsu
August 2024

Nedensel keşif ağlarının tasarımı için yeni bir karma tamsayılı programlama formülasyonu öneriyoruz. Model, verilen bir veri kümesindeki özellikler arasındaki istatistiksel bağımlılık ilişkilerini belirten bir dizi kuralı, yani d-bağlantı ve d-ayrım ilişkilerini alır ve minimum (ağırlıklı) ihlal ile bir nedensel ağ oluşturmayı amaçlar. Geri bildirim döngülerine ve örtük karıştırıcılara izin vererek, formülasyonumuz literatürdeki mevcut yaklaşımların çoğundan ayrılmaktadır. Modelimiz denetimsiz bir makine öğrenme modeli olarak çalışabilse de, karar vericinin bilinen nedensel ilişkileri girmesine olanak tanıyan gerekli esnekliğe sahiptir. Modelimizin performansı, çeşitli sentetik veri setleriyle test edilmiştir.

Anahtar sözcükler: Nedensel Keşif, Tamsayılı Programlama, Ağ Modellemesi.

Acknowledgement

I always felt blessed with the people I encountered at Bilkent. There are so many people that I am thankful to have met along this journey.

First and foremost, I would like to express my immense gratitude to my advisors Prof. Dr. Oya Karaşan and Assoc Prof. Özlem Karsu. Oya Hoca has been my mentor since my sophomore year and taught me everything. Her understanding and commitment as my advisor have been invaluable to me and looking back insisting on being her TA and then RA was one of the best decisions I have made. I would like to also thank Özlem Hoca. Every time I visited her office, I left feeling as motivated as I was on Day 1, without even needing to mention my thoughts about quitting the program. She is truly one of those special people who can energize a room with positivity and motivation without trying. I believe Bilkent and academia are extremely lucky to have her.

I would like to also thank Prof. Bahar Yetiş for always supporting the graduate students. She has been a wall for me that can be relied upon.

I would like to thank the members of my thesis committee, Asst. Prof. Mustafa Kemal Tural and Asst. Prof. Nur Kaynar, for their constructive feedback.

I would like to acknowledge the financial support of The Scientific and Technological Research Council of Turkey (TUBITAK) for the 2210-A National Graduate Study Scholarship Program they awarded.

I would like to begin by thanking my friends Görkem Yeni, Melih Yılmaz, Onur Çoban, and Yusuf Saltan. I am super glad that we began this journey all the way back in dorm 63. I would like to also thank Baturay Konak and Gizem İkizler for being the best couple that one can spend his entire college journey with.

I would like to thank Zehranaz Varol, Deniz Akkaya, Göksu Ece Okur, Deniz Şahin, Bora Çetin, Kaan Çakıroğlu, and Tuna Arda Kınık for their company throughout my graduate studies.

I would like to explicitly thank my cohort, Mustafa Çolak, Ali Eren Demir, Elif Sena Işık, Doğuş Berk Koçak, and Gülin Yurter. IE MS program is an extremely difficult one. But with them every moment was joyful. I am honored to have shared the same classroom with you.

I would like to especially thank Elif Rana Yöner. When I am with her I feel alive. I am eager to continue our journey for the PhD.

Last but certainly not least I would like to thank my family. I thank my aunt Vildan Göker for always giving me sound advice in every aspect of life. I would like to thank my father Latif Haliloğlu for teaching me mathematics from when I was three to grad level. I would like to thank my sister Dilruba Haliloğlu for making Bilkent a home for me.

Finally, I dedicate this thesis to my mother Uğur Haliloğlu. I would like to thank her for keep believing in me even at moments when I stopped believing in myself. Thank you for your dedication and sacrifice all these years.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Most of the scientific hypotheses that are investigated by machine learning methods are causal questions such as "Does regular exercise cause improvements in mental health?" or "Does education level have a causal effect on lifetime earnings?". One can try to answer these questions through statistical analysis using methods such as hypothesis testing, estimation, and regression. Such methods rely on estimating the parameters of a distribution from samples drawn from that distribution and detect associations among variables of interest, using joint distributions of observed variables (1). On the other hand, causal discovery methods focus on relations that can not be inferred from only the associations of variables, as exemplified by the well-known phrase "correlation does not mean causation".

To see the difference between these two lines of approaches, consider the problem of disease prediction. The traditional way of statistical analysis predicts a disease using symptoms as observational variables and uses the same tools for prediction in the other way around, eventually detecting whether these two variables are dependent or not. Causal Discovery methods, on the other hand, utilize the fact that symptoms cannot be a predictor of disease, as a person with symptoms would already be infected (1). Referring back to our initial question of "Does regular exercise cause improvements in mental health?", traditional statistical analysis would indicate a correlation between improvement in mental health and

regular exercise, however, it would fail to detect the causal relationship between these two events. One of the following two statements or both of them could be valid: i) people who have good mental health conditions would exercise more; ii) people who do regular exercise have good mental health. Yet we cannot determine through correlation whether both are valid or, if only one is, which one is valid. The validity of such arguments can be detected through causal discovery, which would identify the cause and effect relations between various factors of interest.

The traditional form of statistical analysis interprets models from only the given (observed) data, disregarding external factors. This can be problematic for causal investigation. By its nature, causal discovery dives deeper into the discussion of unobserved common causes in the dataset. A well-known example of this concept is the firefighter bias. If the traditional methods of statistical learning are utilized, a result such as "Being a firefighter increases life expectancy" can be deduced. However, this interpretation disregards the fact that one must already be physically fit to become a firefighter. In this case, being physically fit is an unobserved common cause for both being a firefighter and having a longer life expectancy, which may be missed by the traditional approaches if health status is not considered in the dataset.

Identifying causal relationships is instrumental not only for better policy-making in public services but also in for-profit businesses. Consider a retail business that sells a range of products. The traditional methods help detecting the products that are perfect complements. However, if one can also identify which of these products triggers the purchase of the other, the marketing campaign can be greatly improved. An example for this would be printers and ink cartridges. These two products are perfect complements. Furthermore, the purchase of ink cartridges is triggered by the purchase of printers, so promoting printer sales would increase overall profit.

Unsupervised machine learning algorithms to identify causal relationships have been investigated since the late 1990s. One of the methods that is used for defining meaningful causal relationships has been creating Causal Graphs as they enable interpretability. Most of the efforts focused on constructing Directed Acyclic

Graphs (DAG) due to their simplicity.

Methods that estimate causal models from a given dataset are divided into two categories as constraint-based and score-based methods. In constraint-based methods, the focus is on finding a structure that satisfies certain dependency and direct causation conditions, whereas score-based methods involve assigning a score to each graph that measures the extent it fits with the data. The goal is to explore the space of graphs and identify one that optimizes the assigned score.

In this thesis, we propose a constraint-based algorithm that takes dependency and independency relationships from statistical tests and returns a network that has the minimum (weighted) violations of these relationships. In particular, we propose an exact approach based on integer programming that has an optimality guarantee.

Chapter 2

Definitions

Causal Discovery Graphs utilize d-separation and d-connection concepts that are used in probabilistic graphical models. Before explaining these concepts, we provide the following preliminary definitions.

In Causal Discovery Graphs variables of a dataset are represented with their respective nodes. The directed arcs between these nodes represent the causal relations between these variables. A directed arc from node i to node j represents a causal relationship between the variables where variable i is a cause of variable j. The variables that the experimenter can control are called conditioning nodes. These variables form the conditioning set and are vital for causal discovery.

Example: Consider three variables: the amount of sugar in a cake, the amount of fat in a cake, and the calories in this cake. The causal structure can be seen in Figure 2.1:

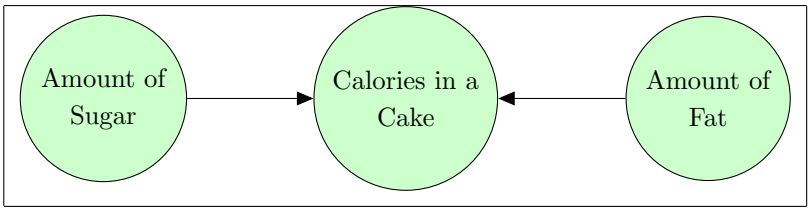


Figure 2.1: Sugar and Fat are Independent

As it stands the amount of sugar that is put into cake is independent of the amount of fat. However, if we condition this causal relationship on calories we will now observe that the amount of sugar and the amount of fat are dependent on each other as if one wants to keep the calories at the same level an increase in the amount of sugar must be compensated with a decrease in the amount of fat. Their new causal relationship can be seen in Figure 2.2

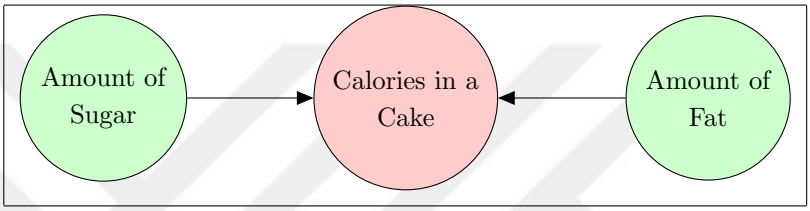


Figure 2.2: Sugar and Fat are Dependent

The variables that are in the conditioning set have an instrumental value in causal discovery graphs.

A *Directed Graph* or a *Network* is denoted by the pair $G = (V, E)$, where V is a set of nodes and E is a set of (directed) arcs, which are ordered pairs of nodes. An arc directed from $i \in V$ towards $j \in V \setminus \{i\}$ is denoted by the tuple $(i, j) \in E$ and depicted with an arrowed line $i \to j$ in G. In directed graphs self-loops are not allowed i.e. $(i, i) \notin E$. A *Directed Mixed Graph* (DMG) is a network where in addition to directed arcs, E contains bi-directed arcs, corresponding to unordered pairs $\{i, j\}$ and depicted as $i \leftrightarrow j$. A *path* is a consecutive sequence of nodes and arcs (both directed and bi-directed) where each node is visited once. If the path starts and ends at the same node, it is called a *cycle*. If all the arcs are directed towards the destination node on a path, it is called a *directed path*. A graph is called acyclic if it does not contain any cycles. Such graphs are called *Acyclic Directed Mixed Graphs (ADMGs)*.

Figure 2.3 depicts a directed path while Figure 2.4 shows a path between s and t.

For a node to be *descendant* of another node there should exist a directed path

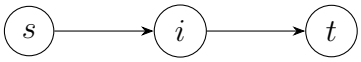


Figure 2.3: Directed path

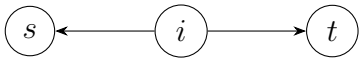


Figure 2.4: Path

from the latter to the former. In Figure 2.3, t is a descendant of s, while in Figure 2.4 it is not. A path visits a node as a *collider* if there are two incoming arcs into this node on the path and as a *non-collider* otherwise.

The following definition is due to (2).

Definition 1 *A path between nodes i and j is unblocked with respect to a set of nodes C called the conditioning set, if every node visited as a collider on the path is in C or has a descendant in C, and no node that is visited as a non-collider is in C. Otherwise, the path is blocked with respect to C.*

Consider Figure 2.5. If the starting and terminal nodes are taken as a and e respectively and the conditioning set is the empty set, $C = \{\emptyset\}$, path $a \to b \to d \to e$ is unblocked. However, if we take the conditioning set as $C = \{b\}$, the path becomes blocked as b is not visited as a collider on this path. On the other hand, if the starting node is taken as a and the terminal node is taken as c, then the path $a \to b \to c$ is blocked with respect to $C = \{\emptyset\}$ and unblocked with respect to $C = \{b\}$. Moreover, for the latter example, the path becomes unblocked if the conditioning set is taken as $C = \{e\}$ or $C = \{d\}$ since b has a descendant in the conditioning set.

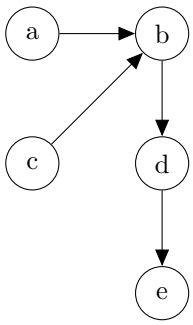


Figure 2.5: Bayesian Network Example

Definition 2 *Two nodes i and j are d-separated with respect to a conditioning set C if all paths between them are blocked. Otherwise, they are d-connected. That is, two nodes i and j are d-connected with respect to a conditioning set C if there is at least one unblocked path between them. (2)*

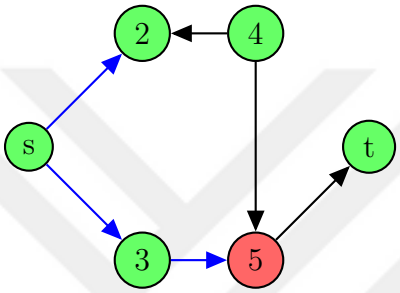


Figure 2.6: Example of d-separation

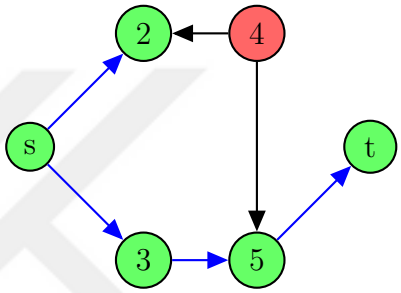


Figure 2.7: Example of D-Connection

In Figure 2.6 nodes s and t are d-separated since node 5, which is in C, cannot act as a collider. On the other hand, in Figure 2.7 nodes 3 and 5 are not in C and they do not act as colliders hence there exists a path between nodes s and t thus these nodes are d-connected. The path can be seen by tracing the blue arcs.

Definition 3 *If node i is d-separated from node j given conditioning set C in a graph $G = (V, E)$ with $i, j \in V$ and $C \subseteq V \setminus \{i, j\}$, then i is independent of j given C (2). This is called the Causal Markov condition.*

Definition 4 *If variable i is independent of variable j given conditioning set C, then in a graph $G = (V, E)$ the node representing variable i must be d-separated from the node representing variable j given C (2). This is called the Faithfulness condition.*

Definition 5 *Given a causal network $G = (V, E)$ and a conditioning set C, a route between nodes i and j is a path on which every node visited as a collider is in C and every node visited as a non-collider is not in C.*

(3) refers to routes as simple paths. We identify routes with unblocked paths and adopt this simplified definition of connectivity for d-connection and d-separation conditions in our model. Although our model uses this simplified definition of d-connectivity, the underlying mathematical modeling framework is flexible enough to be extended to the more general definition of connectivity.

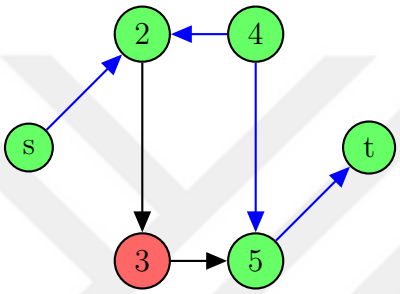


Figure 2.8: Traditional d-separation

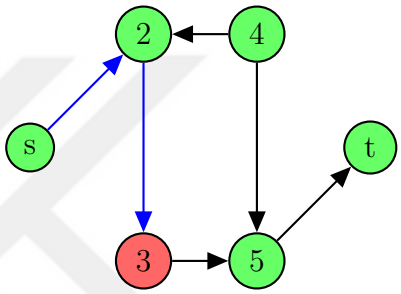


Figure 2.9: Routes for d-separation

In Figure 2.8 the traditional d-separation is assessed. If we use the definition from (2) nodes s and t are d-connected since node 2 can act as a collider as it has a descendant on C which is node 3. In Figure 2.9 our route definition is depicted. If we use the route definition node 2 can no longer be used as a collider hence nodes s and t are d-separated.

A *latent confounder* refers to an unobserved variable that affects both the cause variable and the affected variable in a causal relationship. Latent confounders can bias the estimation of causal effects and hinder discovering the true causal relationship.

Example: The effect of working as a firefighter (A) on the risk of death (Y) will be confounded if "being physically fit" (L) is a cause of both being an active firefighter and having a lower mortality risk. This bias, depicted in the causal diagram is often referred to as a healthy worker bias (4). In the absence of latent confounders in a model, one can derive the result that being a firefighter decreases the risk of mortality (see Figure 2.10). The correct way of modelling would acknowledge the possibility of such a latent confounder and of relations as in Figure 2.11. We allow latent confounders and use a bi-directed arcs to

represent all possible confounders between a pair of variables as in (5), see Figure 2.12.

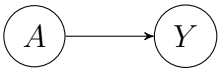


Figure 2.10: Without latent confounder

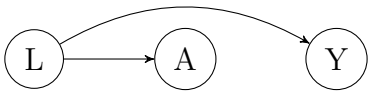


Figure 2.11: Actual causal relation

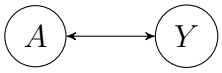


Figure 2.12: Representation

In Figure 2.13 we have a network where the actual causal relation can be seen. Our representation with bi-directed arcs can be seen right next to it in Figure 2.14. These graphs are equivalent.

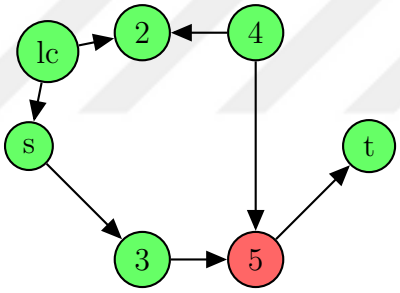


Figure 2.13: Network with Dummy Nodes

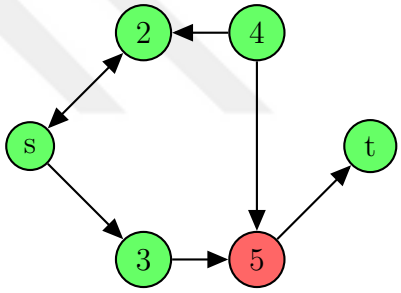


Figure 2.14: Network with Bi-Directed Arcs

Coping with feedback cycles is a notoriously difficult problem while constructing casual discovery networks. Hence, most of the discussions that represent causal structures have been revolving around DAGs (6). However, this structure fails to model feedback cycles thus, disregards the cases in which variables are not well defined in time (7). Allowing feedback cycles also allows causal structures to represent states that are in equilibrium. This paper proposes a model which yields a Directed Mixed Graph (DMG) as an output and hence differs from the majority of current attempts in the literature that focus on DAGs.

Chapter 3

Literature Review

Several algorithms have been proposed for learning the structure of a directed graphical model. These algorithms can be clustered into two main categories: score-based algorithms and constraint-based algorithms. Score-based approaches aim to find the graph that maximizes the likelihood of the data given the graph while constraint based algorithms try to find a graph that satisfies the conditional independencies or any other constraints inferred from the data (8). We discuss some of the existing works in these two categories. We refer the interested reader to (9), (10), and (11) for detailed explanations on some of the proposed algorithms.

Score-based algorithms search for the best network in terms of a score that indicates how well the network explains the data over the set of all possible networks. The score $S(\mathcal{G})$ of a graph is typically calculated based on the marginal likelihood of the data given the graph (7). This assessment can be done with several methods, the most common of which is a consistent approximation of the marginal likelihood, in the form of the Bayesian Information Criterion (BIC) score (12).

There are Dynamic Programming (DP) approaches that yield the optimal graph (13) as well as integer linear programming approaches for discovering causal

networks. A notable example of the latter is the GOBNILP approach (14), which uses parentship sets as variables and utilizes cutting planes to speed up branching to find the optimal graph with the assumption of acyclicity. This method is extended by (15), who propose a branch and price approach to speed up the algorithm. Other integer programming based approaches include the works of (16), (17), (18) and (19).

Exact methods guarantee optimality yet are computationally expensive since finding a graph that yields the optimum score is generally an NP-Hard problem (20). To find a compromise between accuracy and computational time, heuristic algorithms are developed. They are usually centered around greedy heuristics, providing a middle ground between the accuracy of the solution and computational easiness. A well-known algorithm that finds a local maximum score rather than a global maximum is the Greedy Equivalence Search (GES) algorithm of (12).

Note that allowing latent confounders is difficult in score based approaches, rendering them impractical for the problem setting we tackle in this work.

Constraint-based algorithms construct a network that represents features common to all directed graphs compatible with the statistical tests of conditional dependence. Such features common for all tests are taken as constraints in these approaches.

By their nature, constraint-based algorithms can be written as optimization problems that minimizes the (weighted) sum of violated d-separation and d-connection conditions. The logical conditions that are derived from statistical tests are used in boolean satisfiability solvers (21), answer set solvers (22), or with specialized branch and bound techniques (23) to find the optimal graph.

The most widely known constraint-based algorithm is the PC-Algorithm. (24). The PC-Algorithm begins with a fully connected graph, in which directions of the edges are not present. The algorithm then deletes the edges between the nodes that are independent. Once this step is complete, the directions for the edges are

assigned according to the dependency/ independency conditions derived from the statistical test. Although the PC-algorithm laid the foundations of the constraint-based algorithms it does not cover the possibility of latent confounders, which is addressed in an extension called the Fast Causal Inference (FCI) method (25).

A major advantage of the constraint-based algorithms is that they allow complete separation of the statistical tests that are used for dependence and independence conditions and the combinatorial optimization algorithm for designing the network (26) (27). Moreover, their structure allows extension for latent confounding.

The method proposed in this thesis follows this class of algorithms as it is based on an integer optimization model that takes logical conditions from a statistical test, enforces these conditions as constraints, and yields a DMG as an output. It differs from other causal discovery methods using integer optimization ((16), (14), (17), (19) and (18)), as they are score-based while our method takes statistical conditions as constraints. Specifically, they focus on assessing graphs based on the likelihood of the nodes' parentship sets, whereas our method focuses on minimizing the weighted d-separation and d-connection violations.

Furthermore, our approach allows feedback cycles and latent confounders so as to provide a more comprehensive result. The most similar approach to our work is the EdgeGen algorithm proposed in (3), which is based on iteratively enlarging a candidate edge set and using potential paths that can be obtained from them within an integer programming model. We propose an alternative integer programming model that is not iterative and guarantees optimality even when the optimal solution has nonzero violations: any solution with zero violations should be an optimal one while this may not be true for a solution with nonzero violations, making it more difficult to make a conclusive statement on optimality. The structure of our mathematical model also enables customisation in the sense that the users can easily modify the formulation when any information on the network exists such as a known causal relation between two features.

Chapter 4

Model

Before we proceed with the mathematical model that designs a causal network minimizing violations, we provide some preliminaries.

Let V be the set of nodes. A *rule* k is identified by a quadruple (s_k,t_k,C_k,R_k) where s_k is the starting node, t_k is the terminal node, $C_k \subseteq V \setminus \{s_k, t_k\}$ is a given conditioning set, and $R_k = D$ or I, gives the required dependency/independency relationship of pair (s_k,t_k) with respect to conditioning set C_k, respectively. We let $N_k = V \setminus C_k$. We are provided a list of such rules, say set K, perhaps based on a BIC test conducted apriori and would like to design a causal network that overlaps with the given rules as much as possible. Let $K_D = \{k \in K : R_k = D\}$ denote the set of dependent rules and $K_I = \{k \in K : R_k = I\}$ denote the set of independent rules. We say that network $G = (V, E)$ is consistent with rule k, if s_k and t_k are d-connected (d-separated) with respect to C_k in G when $R_k = D$ ($R_k = I$) . Else, we say that rule is violated in G.

To deem G consistent with rule k, our model tries to construct a route (if $k \in K_D$) or to show that no route exists (if $k \in K_I$) in G. Routes are constructed using the model design arcs, i.e., set E by sending flows between source-destination pairs visiting nodes in sets C_k and N_k as well as arcs appropriately. Binary variable y^k indicates the existence of such a path for rule $k \in K_D$. On the other hand, for G

to be consistent with a rule $k \in K_I$, there should be no route between s_k and t_k in G. To ensure that such a path does not exist, our model spans out from root node s_k by constructing a directed tree including all nodes that are reachable from the root node via routes using arcs in E and considering C_k as the conditioning set. Every node in the tree is labeled. We use three types of labels: out, in, and duplicate labels. A node i is out-labeled means there exists a route from s_k to i in the network designed by our model and on this route node i is visited through an outgoing arc, i.e., from its parent node j where $(i, j) \in E$. Similarly, a node i is in-labeled means there exists a route from s_k to i in the network designed by our model, and on this path node i is visited through an incoming arc, i.e., from its parent node j where $(j, i) \in E$. In routes, nodes in C_k can only be visited via incoming arcs and thus they can only have in-labels. However, a node in N_k can be both in and out label. Since extending routes through out-labeled nodes offers more options, we shall favor out-labels in our model. Note that our ultimate aim is to label t_k whenever possible. However, due to cycles, some nodes that can be out-labeled might be in-labeled by a proper choice of traversal orientation of the cycles. To address this issue, we give such nodes a second label, namely a duplicate label, and treat them as out-labeled nodes during further label extensions. If for rule $k \in K_I$, node t_k is not labeled, our designed network G will be declared as consistent with this rule and no penalty will be incurred in our objective. Else, a penalty will be incurred. Let $\omega_k \in \mathbb{R}^+$ be the penalty incurred when rule $k \in K$ is not consistent in G.

The main purpose of our model is to design a network as consistent with our set of rules as possible. This is a challenging task and our model entails a huge number of variables and constraints. We will try and give a preview of our model conceptually before we proceed with the formulation. We have binary arc design variables (making up E) corresponding to directed (z variables) as well as bi-directed (b variables) arcs. For each independent rule and for each node, our model has three types of binary label variables corresponding to in-label (I variables), out-label (O variables), and duplicate-label (D variables) of variables. To construct the rooted trees at source nodes, we have binary tree arc (x variables) as well as binary parentship relationships (p variables) to be able

to detect cycles. For each dependent rule, y variables indicate whether a route connects the source and destination of this rule using arcs in E. To construct such a path, an arc of E can be traversed in the forward orientation (indicated with binary u variables) or the backward orientation (indicated with binary w variables). Similarly, the path can be constructed by using a bi-directed arc of E (indicated with binary v variables). Table 4.1 depicts our variables.

$$
\begin{aligned}
z_{ij} &= \begin{cases} 1, & \text{If there is an arc from } i \in V \text{ to node } j \in V \setminus i \\ 0, & \text{otherwise} \end{cases} \\
b_{ij} &= \begin{cases} 1, & \text{If there is a bi-directed arc between } i \in V \text{ and } j \in V \setminus i \\ 0, & \text{otherwise} \end{cases} \\
I_i^k &= \begin{cases} 1, & \text{If } i \in V \text{ is in-labeled for rule } k \in K_I \\ 0, & \text{otherwise} \end{cases} \\
O_i^k &= \begin{cases} 1, & \text{If } i \in N_k \text{ is out-labeled for rule } k \in K_I \\ 0, & \text{otherwise} \end{cases} \\
D_i^k &= \begin{cases} 1, & \text{If } i \in N_k \text{ is duplicate-labeled for rule } k \in K_I \\ 0, & \text{otherwise} \end{cases} \\
x_{ij}^k &= \begin{cases} 1, & \text{If } (i,j) \text{ belongs to the tree rooted at } s_k \text{ for rule } k \in K_I \\ 0, & \text{otherwise} \end{cases} \\
p_{ij}^k &= \begin{cases} 1, & \text{If } i \in V \text{ is a parent of } j \in V \text{ on the tree routed at } s_k \text{ for rule } k \in K_I \\ 0, & \text{otherwise} \end{cases} \\
y^k &= \begin{cases} 1, & \text{If a route is constructed for rule } k \in K_D \\ 0, & \text{otherwise} \end{cases} \\
u_{ij}^k &= \begin{cases} 1, & \text{If arc } (i,j) \text{ is traversed in a forward manner on the route for rule } k \in K_D \\ 0, & \text{otherwise} \end{cases} \\
w_{ij}^k &= \begin{cases} 1, & \text{If arc } (i,j) \text{ is traversed in a backward manner on the route for rule } k \in K_D \\ 0, & \text{otherwise} \end{cases} \\
v_{ij}^k &= \begin{cases} 1, & \text{If bi-directed arc } (i,j) \text{ is traversed on the route for rule } k \in K_D \\ 0, & \text{otherwise} \end{cases}
\end{aligned}
$$

Table 4.1: Decision Variables

Our causal network design model is the following.

$$\min \qquad \sum_{k\in\mathcal{I}} \omega_k (I^k_{t_k} + O^k_{t_k}) + \sum_{k\in\mathcal{D}} \omega_k (1 - y^k) \tag{4.1}$$

Constraints (4.2)-(4.39) for each independent rule $k \in K_I$:

$$I^k_j + x^k_{ij} \geq 2(O^k_i + I^k_i - p^k_{ji} - \sum_{n\in V\setminus\{i,j\}} x^k_{nj} + z_{ij} - 1) \qquad i\in N_k\setminus t_k, j\in C_k \tag{4.2}$$

$$I^k_j + x^k_{ij} \geq 2(O^k_i + D^k_i - p^k_{ji} - \sum_{n\in V\setminus\{i,j\}} x^k_{nj} + b_{ij} - 1) \qquad i\in N_k\setminus t_k, j\in C_k \tag{4.3}$$

$$O^k_j + x^k_{ij} \geq 2(I^k_i - p^k_{ji} - \sum_{n\in V\setminus\{i,j\}} x^k_{nj} + z_{ji} - 1) \qquad i\in C_k, j\in N_k\setminus s_k \tag{4.4}$$

$$O^k_j + I^k_j + x^k_{ij} \geq 2(I^k_i - p^k_{ji} - \sum_{n\in V\setminus\{i,j\}} x^k_{nj} + b_{ji} - 1) \qquad i\in C_k, j\in N_k\setminus s_k \tag{4.5}$$

$$O^k_j + x^k_{ij} \geq 2(O^k_i + D^k_i - p^k_{ji} - \sum_{n\in V\setminus\{i,j\}} x^k_{nj} + z_{ji} - 1) \qquad i\in N_k\setminus t_k, j\in N_k\setminus\{s_k, i\} \tag{4.6}$$

$$O^k_j + I^k_j + x^k_{ij} \geq 2(O^k_i + I^k_i - p^k_{ji} - \sum_{n\in V\setminus\{i,j\}} x^k_{nj} + z_{ij} - 1) \qquad i\in N_k\setminus t_k, j\in N_k\setminus\{i, s_k\} \tag{4.7}$$

$$O^k_j + I^k_j + x^k_{ij} \geq 2(O^k_i + D^k_i - p^k_{ji} - \sum_{n\in V\setminus\{i,j\}} x^k_{nj} + b_{ij} - 1) \qquad i\in N_k\setminus t_k, j\in N_k\setminus\{i, s_k\} \tag{4.8}$$

$$I^k_j + x^k_{ij} \geq 2(I^k_i - p^k_{ji} - \sum_{n\in V\setminus\{i,j\}} x^k_{nj} + b_{ij} - 1) \qquad i\in C_k, j\in C_k\setminus\{i\} \tag{4.9}$$

$$x^k_{ij} \leq z_{ij} + z_{ji} + b_{ij} \qquad i\in V, j\in V\setminus\{i, s_k\} \tag{4.10}$$

$$I^k_i + O^k_i \leq 1 \qquad i\in N_k \tag{4.11}$$

$$O^k_{s_k} = 1 \tag{4.12}$$

$$O^k_i = 0 \qquad i\in C_k \tag{4.13}$$

$$\sum_{j\in V\setminus\{i\}} p^k_{ji} \leq |V|(O^k_i + I^k_i) \qquad i\in V \tag{4.14}$$

$$\sum_{j\in V\setminus\{i\}} p^k_{ij} \leq |V|(O^k_i + I^k_i) \qquad i\in V \tag{4.15}$$

$$p^k_{jm} + p^k_{ij} - 1 \leq p^k_{im} \qquad i\in V, j\in V\setminus i, m\in V\setminus\{i,j\} \tag{4.16}$$

$$p^k_{ij} + p^k_{ji} \leq 1 \qquad i\in V, j\in V\setminus i \tag{4.17}$$

$$x^k_{ij} \leq p^k_{ij} \qquad i\in V, j\in V\setminus\{i, s_k\} \tag{4.18}$$

$$p_{ij}^k + x_{mj}^k \leq 1 + p_{im}^k \qquad i \in V, j \in V \setminus \{i, s_k\}, m \in V \setminus \{i, j\} \quad (4.19)$$

$$O_i^k + I_i^k = \sum_{j \in V \setminus \{i\}} x_{ji}^k \qquad i \in N_k \setminus s_k \quad (4.20)$$

$$I_i^k = \sum_{j \in V \setminus \{i\}} x_{ji}^k \qquad i \in C_k \quad (4.21)$$

$$I_i^k + O_j^k + z_{ij} - p_{ij}^k \leq 2 \qquad i \in N_k \setminus t_k, j \in N_k \setminus i \quad (4.22)$$

$$I_i^k + I_j^k + z_{ij} - p_{ij}^k \leq 2 \qquad i \in N_k \setminus t_k, j \in C_k \quad (4.23)$$

$$I_i^k + D_j^k + z_{ij} - p_{ij}^k \leq 2 \qquad i \in N_k \setminus t_k, j \in N_k \setminus \{i, s_k\} \quad (4.24)$$

$$D_i^k \geq p_{il}^k + I_i^k + I_m^k + z_{ml} - p_{im}^k + O_l^k + D_l^k - 4 \qquad i \in N_k \setminus s_k, l \in N_k \setminus \{i, t_k\}, m \in N_k \setminus \{i, l, t_k\} \quad (4.25)$$

$$D_i^k \geq p_{il}^k + I_i^k + O_m^k + D_m^k + z_{ml} - p_{im}^k + O_l^k + D_l^k - 4 \qquad i \in N_k \setminus s_k, m \in N_k \setminus \{i, t_k\}, l \in N_k \setminus \{i, m, t_k\} \quad (4.26)$$

$$D_i^k \geq p_{il}^k + I_i^k + O_m^k + D_m^k + z_{lm} - p_{im}^k + O_l^k + D_l^k - 4 \qquad i \in N_k \setminus s_k, m \in N_k \setminus \{i, t_k\}, l \in N_k \setminus \{i, m, t_k\} \quad (4.27)$$

$$D_i^k \geq p_{il}^k + I_i^k + O_m^k + D_m^k + b_{lm} - p_{im}^k + O_l^k + D_l^k - 4 \qquad i \in N_k \setminus s_k, m \in N_k \setminus \{i, t_k\}, l \in N_k \setminus \{i, m, t_k\} \quad (4.28)$$

$$D_i^k \geq p_{il}^k + I_i^k + O_m^k + D_m^k + z_{lm} - p_{im}^k + I_l^k - 4 \qquad i \in N_k \setminus s_k, m \in N_k \setminus \{i, t_k\}, l \in N_k \setminus \{i, m, t_k\} \quad (4.29)$$

$$D_i^k \geq p_{il}^k + I_i^k + I_m^k + z_{lm} - p_{im}^k + I_l^k - 4 \qquad i \in N_k \setminus s_k, l \in N_k \setminus \{i, t_k\}, m \in C_k \quad (4.30)$$

$$D_i^k \geq p_{il}^k + I_i^k + I_m^k + z_{lm} - p_{im}^k + O_l^k + D_l^k - 4 \qquad i \in N_k \setminus s_k, l \in N_k \setminus \{i, t_k\}, m \in C_k \quad (4.31)$$

$$D_i^k \geq p_{il}^k + I_i^k + I_m^k + b_{lm} - p_{im}^k + O_l^k + D_l^k - 4 \qquad i \in N_k \setminus s_k, l \in N_k \setminus \{i, t_k\}, m \in C_k \quad (4.32)$$

$$D_i^k \geq p_{il}^k + I_i^k + O_m^k + D_m^k + z_{ml} - p_{im}^k + I_l^k - 4 \qquad i \in N_k \setminus s_k, m \in N_k \setminus \{i, t_k\}, l \in C_k \quad (4.33)$$

$$D_i^k \geq p_{il}^k + I_i^k + O_m^k + D_m^k + b_{ml} - p_{im}^k + I_l^k - 4 \qquad i \in N_k \setminus s_k, m \in N_k \setminus \{i, t_k\}, l \in C_k \quad (4.34)$$

$$D_i^k \geq p_{il}^k + I_i^k + I_m^k + z_{ml} - p_{im}^k + I_l^k - 4 \qquad i \in N_k \setminus s_k, m \in N_k \setminus \{i, t_k\}, l \in C_k \quad (4.35)$$

$$D_i^k \geq p_{il}^k + I_i^k + I_m^k + b_{ml} - p_{im}^k + I_l^k - 4 \qquad i \in N_k \setminus s_k, m \in C_k, l \in C_k \setminus m \quad (4.36)$$

$$D_i^k \leq I_i^k \qquad i \in N_k \quad (4.37)$$

$$x_{is_k}^k + p_{is_k}^k + x_{t_kj}^k + p_{t_kj}^k = 0 \qquad i \in V \setminus \{s_k\}, j \in V \setminus \{t_k\} \quad (4.38)$$

$$I_i^k, O_i^k, x_{ij}^k, p_{ij}^k, D_i^k \in \{0, 1\} \qquad i \in V, j \in V \setminus i \quad (4.39)$$

Constraints (4.40)-(4.54) for each dependent rule $k \in K_D$:

$$\sum_{j \in V \setminus \{i\}} (u_{ij}^k + w_{ij}^k + v_{ij}^k) - \sum_{j \in V \setminus \{i\}} (u_{ji}^k + w_{ji}^k + v_{ji}^k) = y^k \qquad i = s_k \quad (4.40)$$

$$\sum_{j \in V \setminus \{i\}} (u_{ij}^k + w_{ij}^k + v_{ij}^k) - \sum_{j \in V \setminus \{i\}} (u_{ji}^k + w_{ji}^k + v_{ji}^k) = -y^k \qquad i = t_k \quad (4.41)$$

$$\sum_{j \in V \setminus \{i\}} (u_{ij}^k + w_{ij}^k + v_{ij}^k) - \sum_{j \in V \setminus \{i\}} (u_{ji}^k + w_{ji}^k + v_{ji}^k) = 0 \quad i \in V \setminus \{s_k, t_k\} \quad (4.42)$$

$$u_{ij}^k + w_{ij}^k + v_{ij}^k + u_{ji}^k + w_{ji}^k + v_{ji}^k \leq 1 \qquad i \in V, j \in V \setminus j \quad (4.43)$$

$$v_{is_k}^k + u_{is_k}^k + w_{is_k}^k = 0 \qquad i \in V \setminus s_k \quad (4.44)$$

$$v_{t_k i}^k + u_{t_k i}^k + w_{t_k i}^k = 0 \qquad i \in V \setminus t_k \quad (4.45)$$

$$u_{c_1 c_2}^k + w_{c_1 c_2}^k = 0 \qquad c_1 \in C_k, c_2 \in C_k \setminus c_1 \quad (4.46)$$

$$u_{ij}^k + w_{ji}^k = 0 \qquad i \in C_k, j \in V \setminus i \quad (4.47)$$

$$u_{ij}^k \leq z_{ij} \qquad i \in V, j \in V \setminus i \quad (4.48)$$

$$w_{ij}^k \leq z_{ji} \qquad i \in V, j \in V \setminus i \quad (4.49)$$

$$v_{ij}^k \leq b_{ij} \qquad i \in V, j \in V \setminus i \quad (4.50)$$

$$\sum_{j \in V \setminus \{i\}} u_{ji}^k + \sum_{j \in V \setminus \{i\}} w_{ij}^k + \sum_{j \in V \setminus \{i\}} v_{ji}^k + \sum_{j \in V \setminus \{i\}} v_{ij}^k \leq 1 \qquad i \in N_k \quad (4.51)$$

$$\sum_{j \in V \setminus \{i\}} u_{ij}^k + \sum_{j \in V \setminus \{i\}} w_{ji}^k \leq 0 \qquad i \in C_k \quad (4.52)$$

$$y^k \in \{0, 1\} \quad (4.53)$$

$$w_{ij}^k, u_{ij}^k, v_{ij}^k \in \{0, 1\} \qquad i \in V, j \in V \setminus i \quad (4.54)$$

$$b_{ij} = b_{ji} \qquad i \in V, j \in V \setminus i \quad (4.55)$$

$$z_{ij}, b_{ij} \in \{0, 1\} \qquad i \in V, j \in V \setminus i \quad (4.56)$$

4.1 Objective Function (4.1):

Our aim is to ensure that the destinations of independent rules are not reachable from their sources, while the destinations of dependent rules are accessible from their sources. A labeled destination is a violation of an independent rule and not constructing a path is a violation of a dependent rule. The objective function penalizes these violations using the given penalties.

4.2 Constraints for Independent Rules: (4.2)-(4.39)

We have a group of constraints reserved for independent rules.

Labeling Constraints (4.2)-(4.9):

Consider a rule $k \in K_I$. The x variables keep track of a directed tree rooted at s_k. Note that this tree is superimposed on our arc design variables in E and there exists consecutive edges from s_k to node i means the arcs in E contain a route from s_k to i. A node in the tree will have either an in-label or an out-label. Depending on whether this node belongs to the conditioning set for rule k or not, and depending on its label, there are several possibilities of expanding the tree from such a labeled node. The list of constraints force further labels whenever possible so that if t_k is not on the rooted tree, we can safely be sure that no route exists from s_k to t_k. If (i, j) is an arc of this tree ($x_{ij}^k = 1$) , we say that node i is the *immediate parent* of node j and node j is the *immediate child* of node i and j is labeled from node i, i.e., the route from s_k to i is extendable into a route from s_k to j. p variables keep track of parentship relationships in this tree for detecting cycles, one of the most challenging aspects of our model. If the following conditions are jointly satisfied, constraints (4.2)-(4.9) will force node j to be labeled from node i.

1. Node i is labeled

2. Node j is not already in the tree as the immediate child of a node other than node i

3. Node j is not a parent of node i

4. There exists an arc joining i and j that is consistent with the label of node i, such that a route leading to i can be extended through this arc to form a route to j

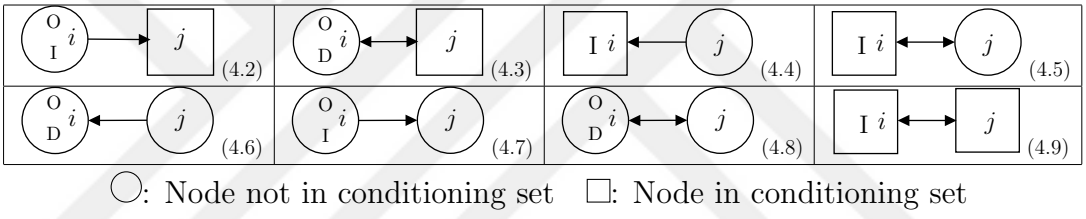


Figure 4.1: Labeling Possibilities

Figure 4.1 depicts a summary of labeling constraints (4.2)-(4.9). Constraints (4.2) and (4.3) force conditioning nodes to be in-labeled from other nodes. A node j in the conditioning set can only be labeled from node i with an incoming arc, i.e., if $z_{ij} = 1$ or $b_{ij} = 1$. However, in the latter case, i should be out-labeled. Note that duplicate labels are given to in-labeled nodes having an alternate possibility of becoming out-labeled (more on this later). With constraints (4.4) and (4.5) nodes in the conditioning set are forced to label other nodes. Note that this is only possible through incoming arcs into node i and directed arcs lead to out-labels while bi-directed arcs force in-labels (only if node j is not already out-labeled). Constraints (4.6), (4.7), and (4.8) cover the cases where a non-conditioning node labels a conditioning node. Different cases occur depending on the directions of the arcs and the labels of the immediate parents. Again, out-labels are favored and in-labels are assigned if the immediate child node j is not already out-labeled through other adjacent edges. Finally, constraints (4.9) force nodes in the conditioning set to in-label each other, a case possible only when they are joined with a bi-directed arc.

Constraints (4.10)-(4.21)*:*

A labeling can only happen if there is an arc between two nodes and this is ensured with (4.10). Constraints (4.11) force nodes to have either in or out labels, if any. With (4.12), the starting nodes of all rooted trees are out-labeled. In routes, conditioning nodes can only be visited via incoming arcs so they can only be in-labeled, which is reflected with (4.13). We then have constraints that regulate the parenting (p) relationships. Ancestors as well as children should be labeled through (4.14)-(4.15). Parentship is a transitive relationship (4.16) and is uni-directional (4.17). Every immediate parent is also a parent via (4.18). As our formulation only allows a single entry, if a node (i) is the parent of a node (j) then it must also be the parent of its immediate parent (m), and this is reflected in (4.19). Being part of a rooted tree is equivalent with being labeled through a single incoming arc. Constraints (4.20) and (4.21) ensure this condition for non-conditioning and conditioning nodes, respectively.

Constraints (4.22)-(4.24)*:*

In the presence of cycles in E, it is possible to label some non-conditioning nodes as both in and out depending on how the cycle is traversed. Out-labeled non-conditioning nodes have greater potential to expand the rooted trees compared to in-labeled ones and in-labeling a node can even result in a failure to label a destination node accurately. For example in Figure 4.2 with all nodes as non-conditioning nodes, in-labeling node 2 via node 1 will block us from constructing the route $s \to 2 \to t$. Constraints (4.22)-(4.23), forbid a node (other than destination) that can be out-labeled to be in-labeled. In particular, if node i can be out-labeled through an out-labeled node $j \in N_k$ (or in-labeled node $j \in C_k$ or duplicate-labeled node $j \in N_k \setminus s_k$) via an outgoing arc then it cannot be in-labeled. Note that with these constraints, node 2 cannot be in-labeled in Figure 4.3.

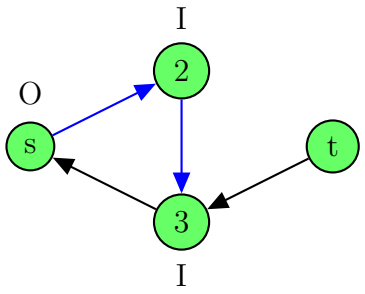


Figure 4.2: Without Constraints (4.22), (4.23), (4.24)

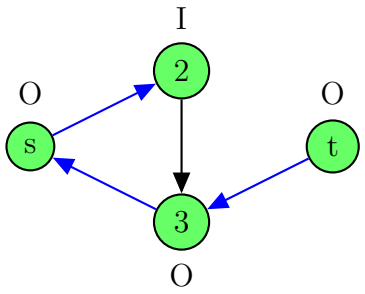
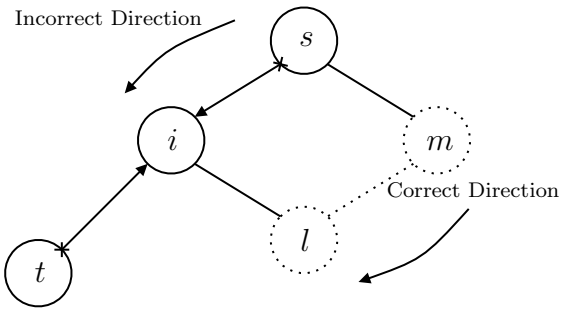


Figure 4.3: With Constraints (4.22), (4.23), (4.24)

Constraints (4.25)-(4.36)*:*

In the above example, we just had to switch the immediate parent of a node to make it out-labeled. However, some cycles might be more challenging to handle. In particular, consider the situation schematized in Figure 4.4. In our rooted tree, node i is in-labeled and has a child node l with a possibility of being labeled through node m such that traversing the cycle including nodes s, i, l, and m (and perhaps other nodes scattered among these four nodes appropriately) correctly will lead to an out-label of node i. So, we add an additional duplicate label to in-labeled nodes having the potential to be switched. so that the route which continues through node t can be found.



$\times$: Arrow may or may not exist

$\rightarrow$: Incoming Arc to Node i

. . . .: Nodes m and l and their joining arc must enable labeling

—: Path Exists

Figure 4.4: Formation of duplicate labels

Having a duplicate-label is equivalent with having an out-label in terms of

further labeling. A summary for our conditions for forcing duplicate labels are:

- Node i should be an in-labeled non-conditioning node
- Node i should be the parent of node l
- Node m should not be a child of node i
- Node m should be labeled and able to relabel node l in a way that the cycle can be traversed in the correct direction and reverse the label of node i

Constraints (4.25) to (4.36) ensure that all possible cases involving nodes i, m, l and the arc between m and l are enumerated. The possibilities and their respective constraint numbers are depicted in Figure 4.5. The constraints (4.25)-(4.36) have the following common structure:

$$D_i^k \geq p_{il}^k + I_i^k + \text{label-type(m)} + \text{arc-type(l,m)} - p_{im}^k + \text{label-type(l)} - 4$$

If the conditions forcing duplicate labels are all met, i.e., node i is in-labeled ($I_i^k = 1$), node i is a parent of l ($p_{il}^k = 1$, node m is not a child of i ($p_{im}^k = 0$), node m is labeled and can label l through arc between m and l (label-type(m), arc-type((l, m)) and label-type(l) are all equal to 1 and consistent with each other), then left hand sides become 1 and force duplicate labeling of node i. In all these conditions, nodes i, l, m and s_k are required to be distinct and $m \neq t_k$ (so that it can further label node l) and $l \neq t_k$ (for otherwise there is no need for duplicate labeling since the ultimate goal is forcing destination to be labeled). Figure 4.5 depicts all viable possibilities between nodes m and l with respective constraint numbers.

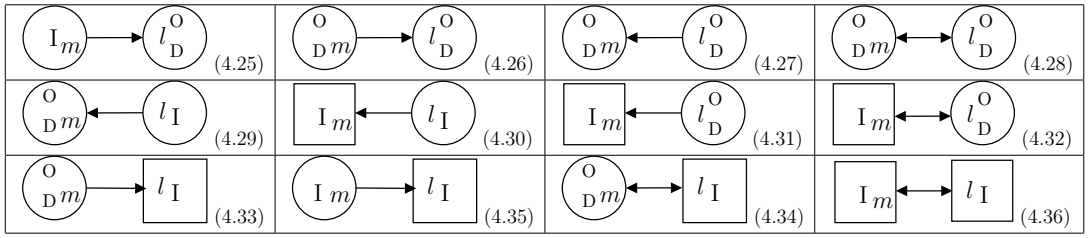


Figure 4.5: Duplicate Label Conditions

Constraints (4.37)-(4.39)*:*

Constraint (4.37) ensures that a node has to be in-labeled to have a duplicate label. (4.38) ensures that the starting node cannot be labeled from any node, the starting node cannot be the child of any node, the terminal node cannot label any node and the terminal cannot be the parent of any node. (4.39) are domain variables for constraints corresponding to independent rules.

4.3 Constraints for Dependent Rules: (4.40)-(4.54)

We have a group of constraints reserved for dependent rules. For a rule $k \in K_D$, we shall try and send a unit of flow along a simple path from s_k to t_k if exists in our designed network. Binary variable y^k will take value 1, favorable for our objective, in such a case. u (w) variables correspond to forward (backward) traversal of design arcs and v variables correspond to traversal of bi-directed arcs.

Flow Balance Constraints (4.40)-(4.42):

Constraints (4.40)-(4.42) are classical flow balance constraints ensuring that the net outgoing flow is y^k for the source node, $-y^k$ for the destination node and zero for any other node.

Constraints (4.43)-(4.54):

With (4.43) we make sure that the flow can only go in a single direction. Flows cannot arrive to our starting node (4.44) and leave our terminal node (4.45). Constraints (4.46) hinder two conditioning nodes to be consecutive through forward or backward traversal of arcs and (4.47) hinder leaving a conditioning node through a forward arc or entering through a backward arc. We also add constraints (4.48), (4.49), and (4.50) to ensure that flow traverses the existing arcs in a proper manner. Since non-conditioning nodes can only have a single incoming arc on a route we add (4.51). We also add (4.52) since conditioning nodes cannot be visited through outgoing arcs. Finally, (4.53)-(4.54) are domain restrictions for dependent variables.

4.4 Common design variables: (4.55)-(4.56)

Bi-directed arcs act like undirected edges through (4.55). Binary directed arc and bi-directed arc design variables are used throughout both parts of our model their domain restrictions are added at (4.56).

Chapter 5

Computational Results of The Model

In this section, we discuss the performance of our model. In our computational tests, we used four settings based on prioritization between the two types of rules and the structure of the ground truth: whether a conflict-free graph is guaranteed to exist (Conflict-Free versus Conflicted) and whether the rules involve all possible conditioning sets or a single one (AllCSets versus SingleCSet). In Figure 5.1 you can see the breakdown of these settings.

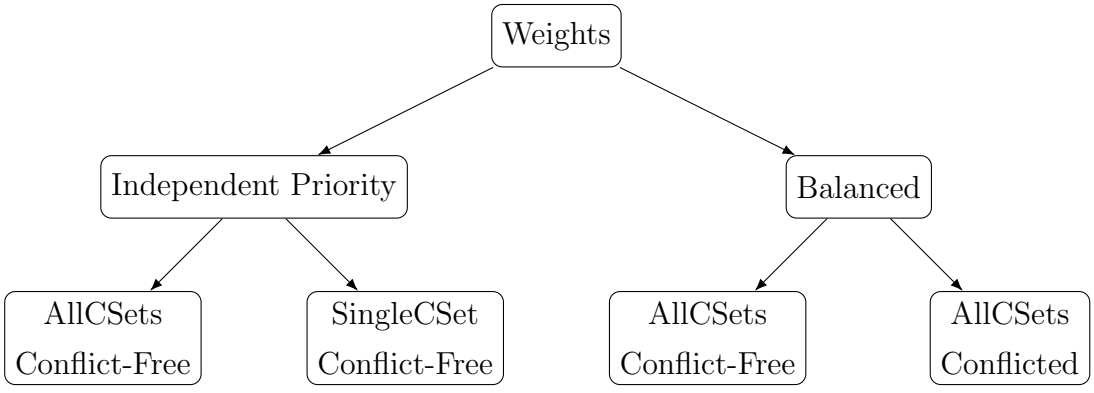


Figure 5.1: Different settings used in experiments

In the first two settings, we assume that independent rules are prioritized over

dependent ones, i.e, $\omega_k = M \quad \forall k \in K_I$ where M is a very large integer and $\omega_k = 1 \quad \forall k \in K_D$. This translates into a lexicographic optimization problem, in which the number of violations for rules in K_I is minimized in the first stage. Then, a second stage problem is solved which minimized total violations from set K_D while keeping independent violations at the minimum level. Note that this minimum level is always zero, as a graph with no edges has zero violation of independent rules. Hence we directly optimize the second stage model and minimize the number of dependency rule violations while enforcing zero independent rule violations. This observation greatly increased the scalability of our approach.

In the first setting, we consider all possible conditioning sets for each starting and terminal node pair. This means that for any arbitrary $s - t$ pair, we take each member of the power set of $V \setminus \{s, t\}$ in separate rules as a conditioning set. So, for a setting with n nodes, there exists $2^{n-2}\binom{n}{2}$ rules. In the second setting, we have a limited conditioning set with cardinality of a maximum one for each pair ($|C_k| = 1$ or $|C_k| = 0 \quad \forall k \in K$), resulting in $\binom{n}{2}$ rules. For these settings, we know that there exists a true data-generating graph that is conflict-free. Hence the optimal value should be zero.

For the remaining two settings, we do not make prioritization and set $\omega_k = 1 \quad \forall k \in K$. We use conflict-free and conflicted settings, respectively; and define rules with all possible conditioning sets for every start terminal pair.

To improve scalability and decrease run times we have used a number of warm-start heuristics. The main idea behind these warm start heuristics is that as the number of arcs increases (resp. decreases) in our graph we satisfy more of the dependent (resp. independent rules). For the cases where we apply independent rule prioritization, we only kept the constraints for independent rules (4.2)-(4.39) and changed the objective function to (5.1).

$$\max \quad \sum_{i \in V} \sum_{j \in V \setminus \{i\}} 2z_{ij} + b_{ij} \tag{5.1}$$

This model finds a graph where all the independent rules are satisfied with the

maximum number of arcs. The solution retrieved from this model must satisfy all of the independent rules and since it maximizes the number of arcs (as opposed to the trivial solution with no arcs), it can be an effective a heuristic solution to achieve fewer violations of dependent rules.

When the problem is not cast as a lexicographic optimization problem, i.e. both types of rules have equal priority, we use a similar warm start approach: This time, we enforce dependent rule satisfaction while minimizing the number of arcs. That is we change our objective function to (5.2) and we only include the constraints that are written for dependent rules (4.40)-(4.54). In both of these heuristics, we keep the constraints (4.55) and (4.56).

$$\min \quad \sum_{i \in V} \sum_{j \in V \setminus \{i\}} 2z_{ij} + b_{ij} \tag{5.2}$$

All computational experiments are conducted on a computer with 32 GB RAM and an Intel(R) Core(TM) i9-10980XE CPU @ 3.00GHz 3.00 GHz chip. In every run, we use a run time limit of 500 seconds. We prioritize branching of the variables that represent the arcs (z and b) since they are the variables that ultimately determine the values of other variables.

We assess the performance of our approaches using two metrics: the error values ϵ_I,ϵ_D,ϵ of the returned solutions, which are the percentage violations of independent, dependent, and all rules respectively; and solution times of the warm start heuristic and the model. We also report how many times the limit of 500 seconds is reached. In addition we provide information on the rule set by reporting the average cardinality of the rule sets K_I and K_D, as well as their ratio $|K_D|/|K_I| + |K_D|$ (Ratio column).

5.1 Independent Rule Prioritized Conflict Free Settings with Full Conditioning Sets

For this setting, we generate test instances of different sizes by adjusting the number of vertices and the maximum node degrees allowed in the graphs. For each $|V|$ and allowed node degree, 10 random graphs are generated. The statistical test indications of each graph are given as parameters to the model, ensuring the existence of a true data-generating graph that is conflict-free.

Table 5.1: Independent Prioritization with Full Conditioning Sets and Node Degree 2

$\|V\|$	$\|K_D\|$	$\|K_I\|$	Ratio	ϵ	ϵ_I	ϵ_D	WS Time	Time	Limits
7	289.9	382.1	0.43	0	0	0	1	3	0
8	907.6	884.4	0.51	0	0	0	20	43	0
9	1936.8	2671.2	0.42	0.1	0	0.24	204	306	5
10	4304.6	7215.4	0.37	0.33	0	0.88	470	455	9

Table 5.2: Independent Prioritization with Full Conditioning Sets and Node Degree 4

$\|V\|$	$\|K_D\|$	$\|K_I\|$	Ratio	ϵ	ϵ_I	ϵ_D	WS Time	Time	Limits
7	627.4	44.6	0.93	0	0	0	1	38	0
8	1654.4	137.6	0.92	0.35	0	0.38	8	450	9
9	4420	188	0.96	0.47	0	0.49	104	500	10
10	10633.8	886.2	0.92	0.78	0	0.85	471	500	10

Table 5.3: Independent Prioritization with Full Conditioning Sets and Node Degree 6

$\|V\|$	$\|K_D\|$	$\|K_I\|$	Ratio	ϵ	ϵ_I	ϵ_D	WS Time	Time	Limits
7	671.9	0.1	1	0	0	0	0	0	0
8	1784.1	7.9	1	0.03	0	0.03	0	151	3
9	4602.9	5.1	1	0.03	0	0.03	0	155	3
10	11503.5	16.5	1	0.02	0	0.02	10	334	6

Tables 5.1 - 5.3 show the results of our experiments. In this setting, our model can be used to solve problems with up to 10 nodes within the memory limit. Our model performs better if the ratio of the dependent rules is either very small or very close to 1. This is because in the independent prioritized cases, forcing the terminal node to be labelled for each rule allows the solver to do a comprehensive presolve, eventually less branching. Moreover, when the ratio of the dependent rules is almost 1, the solver starts branching from the trivial solution where all edges are present. Starting from a point so close to the optimal solution decreases the solution time. Since all independent rules are forced to be satisfied, all errors are from dependent rule violations.

5.2 Independent Rule Prioritized Conflict Free Settings with Limited Conditioning Sets

In this setting, for each $|V|$ and allowed node degree, 10 random graphs are generated, based on which parameters of the model are set. So, we once again know that there exists a true-data-generating graph. The model is able to scale up to more nodes as we only have a single rule for a starting and a terminal node pairing.

Table 5.4: Independent Prioritization with Limited Conditioning Sets and Node Degree 2

$\|V\|$	$\|K_D\|$	$\|K_I\|$	Ratio	ϵ	ϵ_I	ϵ_D	WS Time	Time	Limits
10	17	28	0.38	0	0	0	1	2	0
12	19.9	46.1	0.3	0	0	0	9	13	0
14	22.5	68.5	0.25	0	0	0	39	95	0
16	27.8	92.2	0.23	0	0	0	103	157	1
18	31.7	121.3	0.21	0.01	0	0.05	196	306	4
20	38.9	151.1	0.2	0.10	0	0.49	436	477	9

Table 5.5: Independent Prioritization with Limited Conditioning Sets and Node Degree 4

$\|V\|$	$\|K_D\|$	$\|K_I\|$	Ratio	ϵ	ϵ_I	ϵ_D	WS Time	Time	Limits
10	42.8	2.2	0.95	0	0	0	0	0	0
12	63	3	0.95	0	0	0	2	9	0
14	84.9	6.1	0.93	0.02	0	0.02	15	187	3
16	101.8	18.2	0.85	0.22	0	0.26	281	400	8
18	135.2	17.8	0.88	0.20	0	0.23	266	450	9
20	-	-	-	-	-	-	-	-	-

Table 5.6: Independent Prioritization with Limited Conditioning Sets and Node Degree 6

$\|V\|$	$\|K_D\|$	$\|K_I\|$	Ratio	ϵ	ϵ_I	ϵ_D	WS Time	Time	Limits
10	45	0	1	0	0	0	0	0	0
12	65.2	0.8	0.99	0	0	0	0	1	0
14	90	1	0.99	0	0	0	0	2	0
16	119.3	0.7	0.99	0	0	0	12	1	0
18	151.1	1.9	0.99	0	0	0	40	68	1
20	189.3	0.7	1	0	0	0	48	55	1

Table 5.7: Independent Prioritization with Limited Conditioning Sets and Node Degree 8

	Node Degree 8								
$\|V\|$	$\|K_D\|$	$\|K_I\|$	Ratio	ϵ	ϵ_I	ϵ_D	WS Time	Time	Limits
10	45	0	1	0	0	0	0	0	0
12	65.9	0.1	1	0	0	0	0	0	0
14	91	0	1	0	0	0	0	0	0
16	120	0	1	0	0	0	0	0	0
18	153	0	1	0	0	0	0	0	0
20	190	0	1	0	0	0	0	1	0

As can be seen in the tables above, the model is able to scale up to 20 nodes with the only memory issue occurring in instances with $|V| = 20$ and a maximum node degree of 4. Similar to the previous setting, the model performs better if the dependent rule ratio is lower or approximates 1. Since we are satisfying all of the independent rules, the errors are due to violations in the dependent rule set.

5.3 Weighted Conflict Free Cases

As previously mentioned, for the remaining two cases we take the weights of each rule as equal. For this case, we solve the model for all conditioning sets. For each $|V|$ and allowed node degree, 10 random graphs are generated.

Table 5.8: Performance Metrics for Node Degree 2 (Weighted Conflict-Free)

$\|V\|$	$\|K_D\|$	$\|K_I\|$	Ratio	ϵ	ϵ_I	ϵ_D	WS Time	Time	Limits
5	53.3	26.7	0.67	0	0	0	1	0	0
6	114.1	125.9	0.48	0	0	0	12	7	0
7	346.3	325.7	0.52	0	0	0	209	137	0

Table 5.9: Performance Metrics for Node Degree 3 (Weighted Conflict-Free)

$\|V\|$	$\|K_D\|$	$\|K_I\|$	Ratio	ϵ	ϵ_I	ϵ_D	WS Time	Time	Limits
5	69.9	10.1	0.87	0	0	0	3	0	0
6	188.8	51.2	0.79	0	0	0	158	3	0
7	538.9	133.1	0.8	0.02	0.14	0	447	143	1

The results of our experiments are provided in Tables 5.8 and 5.9. Since these problems are more challenging, the scalability of our model decreases dramatically. However, we are still able to find optimal solutions for problems with up to seven nodes.

5.4 Weighted Conflicted Cases

Thus far, all of our cases were generated from a ground truth graph to check performance under a set of rules that have zero conflict. For the conflicted setting, we again create a graph to attain all rules however, we change 10 % of the rules to create potential conflicts. There is no longer guarantee that there exists a ground-truth graph but we have an upper bound on optimal ϵ as 0.1.

We create 10 graphs with different number of nodes and maximum allowed node degrees. In this setting, we have equal weights for rules and full conditioning sets.

Table 5.10: Performance Metrics for Node Degree 2 (Conflicted)

$\|V\|$	$\|K_D\|$	$\|K_I\|$	Ratio	ϵ	ϵ_I	ϵ_D	WS Time	Time	Limits
5	45.1	34.9	0.56	0.1	0.157	0.055	0	2	0
6	114.3	125.7	0.48	0.178	0.239	0.111	9	375	6

As expected, this setting is the hardest for our problem to scale in. When $|V| = 5$ we received the optimal solution within the time limit in all runs. For $|V| = 3$ we can observe that our error is smaller than our change rate of 0.1. This means that our model found graphs that yield less conflicts than the ones

Table 5.11: Performance Metrics for Node Degree 3 (Conflicted)

$\|V\|$	$\|K_D\|$	$\|K_I\|$	Ratio	ϵ	ϵ_I	ϵ_D	WS Time	Time	Limits
5	56.4	23.6	0.71	0.098	0.262	0.028	1	4	0
6	176.2	63.8	0.73	0.125	0.404	0.023	235	439	8

we have created. When $|V|$ is increased to six some of our runs reach the time limit, hence not yielding optimal graphs. This results in an average error that is higher than the upper bound of 0.1.

Chapter 6

Further Approaches to the Problem

In this chapter, we will go through ideas that was discussed before writing the model that is presented in Chapter 4. These approaches had the potential to lead us to a simpler mathematical model, however, their incorporation created several issues about the nature of the problem.

The arcs were given as parameters in our initial tests to check if the labeling logic worked correctly. To test whether a route existed in our graph we used the algorithms in Appendix A. While testing the model, we observed that the variables p and D can be written as continuous variables rather than binary. However, making this change did not improve the scalability so, in our final model we present an IP model where all variables are taken as binary.

6.1 Ideas for Eliminating C_k

As previously mentioned the conditioning sets play a vital role in designing causal discovery networks. We force the nodes that represent the variables in the conditioning set to act as a collider inside a route. If a variable is taken into the conditioning set the node representing this variable can block or unblock a route. Naturally, this generalization also holds true for our model as many of our constraints and decision variables are centered around the correct representation of the conditioning set. If a method could have been thought in which we can ease this requirement we may have ended up with a simpler model. In the below examples, the nodes that are painted red represent the variables that are on the conditioning set. The nodes that are painted green represent the variables that are not on the conditioning set.

To reach the goal of a simplified model we went through several ideas. The first one of these was about removing the nodes that represented the variables in the conditioning set. We had the following idea if a node in a C_k is approached with two incoming arcs we would delete these arcs and connect the neighbors of the node in C_k with directed arcs in each direction. The illustration of the idea can be seen in Figure 6.1.

(a) Initial Network (b) Proposed Network

Figure 6.1: Comparison of Initial and Proposed Networks Without Colliders

However, the point where this idea fails is quite easy to observe. In the network represented in Figure 6.1a there exists a route between nodes 1 and 5. However in the proposed network which can be seen in Figure 6.1b the route is blocked since none of the nodes can be passed as a collider. So this adjustment changes

the nature of the network which is something that should be avoided.

Another proposition that followed a similar idea was to keep the directed arcs that went to the node in the conditioning set and add new directed arcs in each direction between the nodes that these arcs originated from. This in theory solves the issue in the prior example as the rule would have still been consistent.

An example of this approach can be seen in Figure 6.2. In this network, there is a directed arc from nodes 2 and 4 to node 3. We add new directed arcs in each direction between nodes 2 and 4. We do not add new arcs between the neighbors of node 2 since both arcs are not incoming arcs for that node.

(a) Initial Network (b) Proposed Network

Figure 6.2: Comparison of Initial and Proposed Networks With Extra Edges

Once again the observer can easily recognize the problem. This time in the initial network which can be seen on Figure 6.2a a route does not exist. However, if we did the described adjustments we would end up with a network where a route does exist between nodes 1 and 5. This network is depicted in Figure 6.2b.

6.2 Ideas for Adding Shortcuts

We have also thought of the idea of adding shortcuts from the nodes that are in N_k and labeled and which we know there exists a directed path to a node in C_k. We hoped that this would lead to more practical routes and potentially lead to a model with fewer variables.

(a) Without Shortcut

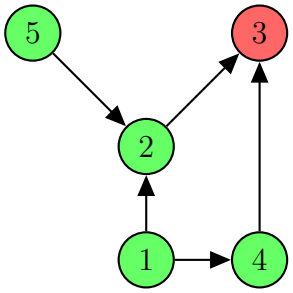
(b) With Shortcut

Figure 6.3: Comparison of Initial and Proposed Networks With Shortcuts to Nodes in C_k

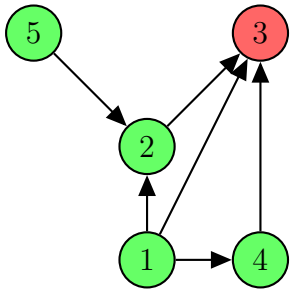
An example of the implementation of this idea can be seen in Figure 6.3 where node 1 is our starting node (s_k) and node 4 is our terminal node (t_k). Since node 1 is the s_k we know that it is going to be labeled. So, we add an arc from node 1 to node 3 which would act as a shortcut. However, observing the issue on this approach is trivial as in our original network which is depicted in Figure 6.3a a route does not exist between nodes 1 and 4. On the other hand in the network with shortcuts which can be seen in Figure 6.3b a route does exist between the nodes.

Shortcuts idea can be extended in the following manner; when the arc for the shortcut is added and used in the route, the arcs that it bypasses can be also seen as passed. This would solve the problem in the example prior as it would not create a new route since the arc from node 2 to node 3 cannot be used if the added arc is used for labeling.

However, this idea causes issues in dense graphs where there are multiple directed paths from a node in N_k to a node in C_k. An example can be seen in Figure 6.4



(a) Without Modified Shortcut



(b) With Modified Shortcut

Figure 6.4: Comparison of Initial and Proposed Networks With Modified Shortcuts to Nodes in C_k

In this example, s_k is node 1 and t_k is node 5. There exists a route between these nodes in the form $1 \rightarrow 4 \rightarrow 3 \leftarrow 2 \leftarrow 5$. In the original network which is shown in Figure 6.4a there exist two directed paths from node 1 to node 3 which is in C_k. So, a shortcut arc is added which is directed from node 1 to node 3 as can be seen in Figure 6.4b.

The issue with this approach is that when labeling occurs using the directed arc from node 1 to node 3, the arcs (1,2) and (2,3) are deemed as past and the route is now blocked. Thus, networks are no longer equivalent.

6.3 Approaches to Cycles in Independent Rules

A challenging part of our study was to investigate the cycles and construct the problem so that if the nodes on C_k are labeled, they are labeled from the correct direction. If a route does exist between two nodes, labeling conditions should be adjusted correctly so that this route is found.

This problem is tackled in the final model with the creation of duplicate variables and their respective constraints. We have thought of alternative ways to model such cycles so that we may end up with a simpler model.

An example can be seen in Figure 6.5. Here consider node 1 as s_k and node 6 as t_k. One can observe that there exists a route between these nodes. The route is $1 \rightarrow 2 \rightarrow 4 \leftarrow 5 \leftarrow 6$. However, if node 4 is labeled from the incorrect direction i.e. from node 5 then the route between these nodes is blocked. In our final model, this issue is solved with duplicate labels. In this network, in the solution of the model, at least one of the nodes 3 and 5 would have a duplicate label or an out-label thus, would be able to label t_k. We tried to think of alternative ways to model these types of cycles.

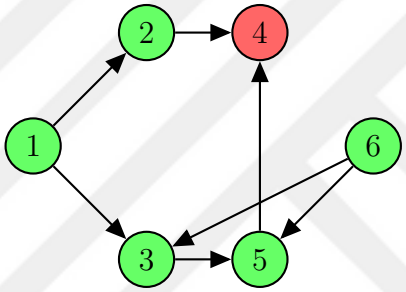


Figure 6.5: Example for a Cycle

We wanted to enforce a change in direction to label t_k if possible. So we wrote the following constraint shown in (6.1)

$$O_i^k + P_i^k \geq z_{nc} + z_{ic} - O_{t_k}^k - I_{t_k}^k + O_n^k + I_n^k + x_{ic}^k - 3 \quad \forall k \in K_I, \forall c \in C_k, \forall i, n \in N_k \tag{6.1}$$

Here we introduce a new binary variable $P_i^k \quad \forall k \in K_I, \forall i \in N_k$ which will be referred as the penalty variable. The logic of this constraint is the following; if all of these conditions hold for a node $i \in N_k$ either the penalty variable becomes 1 or the node i is forced to be out-labeled.

- There are directed arcs from nodes n and i which are in N_k to node c which is in C_k
- t_k is not labeled
- Node n is labeled

- Node c is labeled from node i

Our goal in this constraint is to check both directions for labeling within a cycle that contains a node in C_k. To change the direction of the labeling, a node in C_k should have two incoming arcs. t_k should not be labeled as if it is labeled there is no need for us to change the direction of the labeling. Both sides of the node in C_k should be labeled so we can change the direction. And node i is the immediate parent of node c as if the last condition is not true because of the structure of the cycle either i would have already been out-labeled or a labeling cycle would not have been formed.

The penalty variable is added to relax this constraint when out-labeling is not a possibility for node i. This can happen in the case represented in Figure 6.6

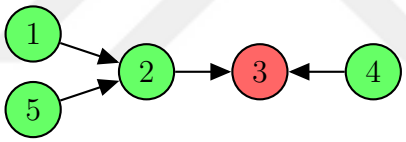


Figure 6.6: Example for Relaxation

In this example consider node 1 as s_k and node 5 as t_k a route does not exist between these nodes. If the constraint (6.1) is added for the case where i is 2, c is 3, and, n is 4 the RHS of this constraint equals to 1. This will force node 2 to be out-labeled which is impossible because of the structure of our network. So, the penalty term will be equal to 1.

However, if we let all penalty terms to be 1 this constraint will be redundant. To avoid redundancy we dwelled on several ideas for putting an upper bound to the penalty variable determined by the nature of the network. This proved to be a challenging task and we could not come up with an adequate formulation.

Another idea about controlling the penalty variable was to put it in the objective function for the independent rules. The important point here is to make the penalty variable more valuable than the violation of the rule. For a minimization

problem if the penalty variable's coefficient is not large enough the model would still make the constraint (6.1) redundant by making the penalty variable equal to 1 and avoid the labeling of t_k.

This approach was very promising and worked perfectly for the cases where the directed arcs z were given as parameters. However, our main goal in discovering causal graphs is to keep arcs as variables. When we switched to taking z's as decision variables we encountered issues.

Recall that the penalty variable has to be more important than the weight of the violation for the limitation of P_i^k to work properly. Thus, we ensure that the model yields a network with the least amount of P_i^k. Our aim of finding the network with the least amount of weighted violations is compromised as some of the optimal structures might be missed when our primary aim is to minimize P_i^k.

6.4 Example Network

In this section, we will be going through some example graphs and the labeling results of our nodes. Consider the network depicted in Figure 6.7. In this example, the s_k is node 1 and t_k is node 6. The reader can observe that there exists a route in the form of $1 \rightarrow 3 \rightarrow 5 \leftarrow 2 \leftarrow 4 \rightarrow 6$.

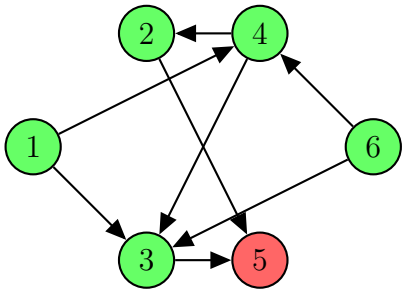


Figure 6.7: Example for the Duplicate Label

This case is especially important for understanding how the duplicate variables work and why they are necessary. If the duplicate variables were not created the

route could not have been found. The results without the duplicate variable can be seen in Figure 6.8

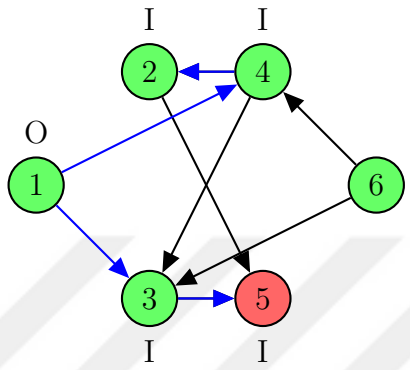


Figure 6.8: Labeling Without Duplicate Variables

In Figure 6.8 the arcs that are painted in blue represent the arcs that are used for labeling. The labels of the nodes can be seen near the nodes. It can be observed that in-labeling nodes 3 and 4 block the route. When we add the duplicate labels we end up with the solution depicted in Figure 6.9.

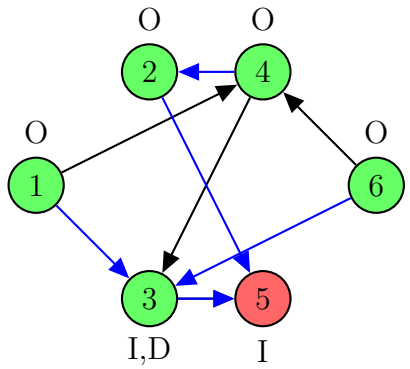


Figure 6.9: Labeling With Duplicate Variables

As one can observe the duplicate variable works as intended. Node 3 has the potential to be out-labeled hence it has the duplicate label. This allows t_k to be labeled from node 3. Note that this is an alternative solution for our model as node 4 could have also had a duplicate label. Moreover, our model also does not

restrict the solution without the duplicate variable where the labeling tree is $1 \rightarrow 3 \rightarrow 5 \leftarrow 2 \leftarrow 4 \rightarrow 6$.

6.5 Model Without Bi-Directed Arcs

Recall that in our model bi-directed edges stand for a structure that can be modeled with directed edges. The actual relationship is depicted in Figure 2.11. Representation with bi-directed edges is shown in Figure 2.12. If a decision maker would like to construct a model with only directed edges our model can easily be reformulated.

This reformulation would require the integration of dummy nodes representing the latent confounders. The model without bi-directed arcs can be seen in Appendix B. Dummy nodes are taken inside N_k. The critical adjustments that one has to make in the model, are ensuring latent confounders can only be connected to their effects (B.44) and they cannot connect with incoming arcs (B.45). Moreover, if the latent confounder is attached to the network it must be attached to both its causes (B.46).

In our trial runs it is observed that modeling with bi-directed arcs provides better scalability.

Chapter 7

Conclusion and Future Work

In this thesis, we present a novel constraint-based approach for designing causal graphs with latent confounders and feedback loops using a mathematical modeling framework. Our integer programming model takes a set of rules that define independency and dependency relations, and aims to generate a network with minimum (weighted) violations of these rules.

These rules, as well as their weights if any, are determined through statistical tests. Our approach is one of the few methods in the literature that can handle both confounders and feedback loops. Moreover, it guarantees finding a network with minimum violations, even when the minimum is nonzero. The structure of our mathematical model also allows flexibility in the sense that any further information such as known direct causation relations can easily be incorporated, enabling a high degree of customization. Such additional information would also increase the computational efficiency of the model.

We demonstrate the computational feasibility of our model on a large number of test instances with different characteristics. The results of our experiments show that our approach is comparable to existing approaches in the literature and has the potential to outperform them when the graph is allowed to be denser (max degree).

For future work, the model can be extended so that it becomes more scalable. Several combinatorial optimization methods can be applied to our model.

Two major ideas are implementing Benders Decomposition and Lagrangian Decomposition. For the former, with proper cuts and adjustments, our model has the potential to be written in a way that only the variables that represent the edges remain as binary variables and the rest of the variables as continuous variables. Since our model merges two models for dependent and independent routes, theoretically, relaxing the integrality constraints for variables in one of these problems should allow Benders Decomposition. Moreover, it can be easily inspected that all but the variables representing the edges are separate for each rule. Hence if these variables are copied by utilizing Lagrangian Decomposition the model can be decomposed into as many subproblems as the number of rules. Then the branch and price method can be applied. A method that is proven to be effective in handling large-scale optimization problems.

As already observed in the existing literature, constructing DMGs for causal discovery is a very difficult problem, leading to exact approaches not being scalable. We believe that our mathematical modeling approach would constitute an important base for further implementations of computational optimization and would trigger a number of future research questions in this area.

Bibliography

[1] J. Pearl, "Causal inference in statistics: An overview," *Statistics Surveys*, vol. 3, 1 2009.

[2] J. Pearl, *Causality: Models, Reasoning, and Inference*. Cambridge University Press, 2000.

[3] F. Eberhardt, N. Kaynar, and A. Siddiq, "Discovering causal models with optimization: Confounders, cycles, and instrument validity," *Management Science*, 7 2024.

[4] M. Hernan and J. Robins, *Causal Inference: What If*. Chapman & Hall/CRC monographs on statistics & applied probability, Taylor & Francis, 2023.

[5] R. J. Evans, "Graphs for margins of bayesian networks," *Scandinavian Journal of Statistics*, vol. 43, no. 3, pp. 625–648, 2016.

[6] M. Drton and M. H. Maathuis, "Structure learning in graphical modeling," *Annual Review of Statistics and Its Application*, vol. 4, pp. 365–393, 3 2017.

[7] C. Squires and C. Uhler, "Causal structure learning: A combinatorial perspective," *Foundations of Computational Mathematics*, vol. 23, pp. 1781–1815, 10 2023.

[8] S. Triantafillou and I. Tsamardinos, "Score-based vs constraint-based causal learning in the presence of confounders," in *CFA@UAI*, 2016.

[9] P. Spirtes, C. Glymour, and R. Scheines, *Causation, Prediction, and Search*, vol. 81. 01 1993.

[10] R. Neapolitan, *Learning Bayesian Networks*. Artificial Intelligence, Pearson Prentice Hall, 2004.

[11] D. Koller and N. Friedman, *Probabilistic Graphical Models: Principles and Techniques - Adaptive Computation and Machine Learning*. The MIT Press, 2009.

[12] D. M. Chickering, "Optimal structure identification with greedy search," *J. Mach. Learn. Res.*, vol. 3, p. 507–554, mar 2003.

[13] M. Koivisto and K. Sood, "Exact bayesian structure discovery in bayesian networks," *J. Mach. Learn. Res.*, vol. 5, p. 549–573, dec 2004.

[14] J. Cussens, "Gobnilp: Globally optimal bayesian network learning using integer linear programming," 2013.

[15] J. Cussens, "Branch-price-and-cut for causal discovery," in *Proceedings of the Second Conference on Causal Learning and Reasoning* (M. van der Schaar, C. Zhang, and D. Janzing, eds.), vol. 213 of *Proceedings of Machine Learning Research*, pp. 642–661, PMLR, 11–14 Apr 2023.

[16] T. Jaakkola, D. Sontag, A. Globerson, and M. Meila, "Learning bayesian network structure using lp relaxations," in *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics* (Y. W. Teh and M. Titterington, eds.), vol. 9 of *Proceedings of Machine Learning Research*, (Chia Laguna Resort, Sardinia, Italy), pp. 358–365, PMLR, 13–15 May 2010.

[17] Y. W. Park and D. Klabjan, "Bayesian network learning via topological order," *Journal of Machine Learning Research*, vol. 18, no. 99, pp. 1–32, 2017.

[18] H. Manzour, S. Küçükyavuz, H.-H. Wu, and A. Shojaie, "Integer programming for learning directed acyclic graphs from continuous data," *INFORMS Journal on Optimization*, vol. 3, pp. 46–73, 1 2021.

[19] S. Kucukyavuz, A. Shojaie, H. Manzour, L. Wei, and H.-H. Wu, "Consistent second-order conic integer programming for learning bayesian networks," 2022.

[20] D. M. Chickering, *Learning Bayesian Networks is NP-Complete*, pp. 121–130. New York, NY: Springer New York, 1996.

[21] A. Hyttinen, P. O. Hoyer, F. Eberhardt, and M. Järvisalo, "Discovering cyclic causal models with latent variables: A general sat-based procedure," *CoRR*, vol. abs/1309.6836, 2013.

[22] A. Hyttinen, F. Eberhardt, and M. Järvisalo, "Constraint-based causal discovery: Conflict resolution with answer set programming," in *Conference on Uncertainty in Artificial Intelligence*, 2014.

[23] K. Rantanen, A. Hyttinen, and M. Järvisalo, "Discovering causal graphs with cycles and latent confounders: An exact branch-and-bound approach," *International Journal of Approximate Reasoning*, vol. 117, pp. 29–49, 2 2020.

[24] P. Spirtes, C. Glymour, and R. Scheines, *Causation, Prediction, and Search*. The MIT Press, 1 2001.

[25] P. Spirtes, "An anytime algorithm for causal inference," in *Proceedings of the Eighth International Workshop on Artificial Intelligence and Statistics* (T. S. Richardson and T. S. Jaakkola, eds.), vol. R3 of *Proceedings of Machine Learning Research*, pp. 278–285, PMLR, 04–07 Jan 2001. Reissued by PMLR on 31 March 2021.

[26] P. Spirtes and K. Zhang, "Causal discovery and inference: concepts and recent methodological advances," *Applied Informatics*, vol. 3, p. 3, 12 2016.

[27] F. Eberhardt, "Introduction to the foundations of causal discovery," *International Journal of Data Science and Analytics*, vol. 3, pp. 81–91, 3 2017.

Appendix A

Algorithms for Route Checking

Algorithm 1 Has Collider Check

```
procedure HAS_COLLIDER(path, graph, direction, N, C)
    if len(path) ≤ 2 then
        return False
    end if
    for i ← 1 to len(path) − 2 do
        if direction[i − 1] == 'forward' and direction[i] == 'incoming' and
path[i] ∈ N then
            return True
        end if
        if path[i] ∈ C and (direction[i − 1] ≠ 'forward' or direction[i] ≠
'incoming') then
            return True
        end if
    end for
    return False
end procedure
```

Algorithm 2 BFS Algorithm for Graph with Colliders

```
procedure BFS(graph, start, end, N, C)
    queue ← deque()
    queue.append(([start], []))
    while queue is not empty do
        path, directions ← queue.popleft()
        continue if isinstance(path, int) else node ← path[−1]
        if node == end return (path, directions)
        for i ← 0 to len(graph) − 1 do
            if i ∈ N or i ∈ C then
                if (graph[i][node] == 1 and i ∉ path) then
                    new_path ← path + [i]
                    new_directions ← directions + ['incoming']
                else if (graph[node][i] == 1 and i ∉ path) then
                    new_path ← path + [i]
                    new_directions ← directions + ['forward']
                end if
                if defined(new_path) and not
has_collider(new_path, graph, new_directions, N, C) then

                    if new_path ∉ [p for p, _ in queue]
                    queue.append((new_path, new_directions))
                end if
            end if
        end for
    end while
    return None
end procedure
```

Appendix B

IP for the Model Without Bi-Directed Arcs

A node is in V_A if the variable it represents is in the observed set, it is in V_L if it is a latent confounder. V is the union of these two sets hence the equation $V \setminus V_A = V_L$ holds. A node i in V_L has a set which is referred as V_{L_i} for the original nodes that this latent confounder can be connected. $|V_{L_i}| = 2 \quad \forall i \in V_L$ as a latent confounder can be connected to two nodes. Moreover $V_{L_i} = \{j, m\}$ where $j \in V_A$ and $m \in V_A \setminus j$ where j and m are the original nodes that dummy node i can be connected.

$$\min \qquad \sum_{k \in \mathcal{I}} \omega_k (I^k_{t_k} + O^k_{t_k}) + \sum_{k \in \mathcal{D}} \omega_k (1 - y^k) \tag{B.1}$$

Constraints (B.2)-(B.31) for each independent rule $k \in K_I$:

$$I^k_j + x^k_{ij} \geq 2(O^k_i + I^k_i - p^k_{ji} - \sum_{n \in V \setminus \{i,j\}} x^k_{nj} + z_{ij} - 1) \qquad i \in N_k \setminus t_k, j \in C_k \tag{B.2}$$

$$O^k_j + x^k_{ij} \geq 2(I^k_i - p^k_{ji} - \sum_{n \in V \setminus \{ij\}} x^k_{nj} + z_{ji} - 1) \qquad i \in C_k, j \in N_k \setminus s_k \tag{B.3}$$

$$O_j^k + x_{ij}^k \geq 2(O_i^k + D_i^k - p_{ji}^k - \sum_{n \in V \setminus \{i,j\}} x_{nj}^k + z_{ji} - 1) \qquad i \in N_k \setminus t_k, j \in N_k \setminus \{s_k, i\} \qquad \text{(B.4)}$$

$$O_j^k + I_j^k + x_{ij}^k \geq 2(O_i^k + I_i^k - p_{ji}^k - \sum_{n \in V \setminus \{i,j\}} x_{nj}^k + z_{ij} - 1) \qquad i \in N_k \setminus t_k, j \in N_k \setminus \{i, s_k\} \qquad \text{(B.5)}$$

$$x_{ij}^k \leq z_{ij} + z_{ji} \qquad i \in V, j \in V \setminus \{i, s_k\} \qquad \text{(B.6)}$$

$$I_i^k + O_i^k \leq 1 \qquad i \in N_k \qquad \text{(B.7)}$$

$$O_{s_k}^k = 1 \qquad \text{(B.8)}$$

$$O_i^k = 0 \qquad i \in C_k \qquad \text{(B.9)}$$

$$\sum_{j \in V \setminus \{i\}} p_{ji}^k \leq |V|(O_i^k + I_i^k) \qquad i \in V \qquad \text{(B.10)}$$

$$\sum_{j \in V \setminus \{i\}} p_{ij}^k \leq |V|(O_i^k + I_i^k) \qquad i \in V \qquad \text{(B.11)}$$

$$p_{jm}^k + p_{ij}^k - 1 \leq p_{im}^k \qquad i \in V, j \in V \setminus i, m \in V \setminus \{i, j\} \qquad \text{(B.12)}$$

$$p_{ij}^k + p_{ji}^k \leq 1 \qquad i \in V, j \in V \setminus i \qquad \text{(B.13)}$$

$$x_{ij}^k \leq p_{ij}^k \qquad i \in V, j \in V \setminus \{i, s_k\} \qquad \text{(B.14)}$$

$$p_{ij}^k + x_{mj}^k \leq 1 + p_{im}^k \qquad i \in V, j \in V \setminus \{i, s_k\}, m \in V \setminus \{i, j\} \qquad \text{(B.15)}$$

$$O_i^k + I_i^k = \sum_{j \in V \setminus \{i\}} x_{ji}^k \qquad i \in N_k \setminus s_k \qquad \text{(B.16)}$$

$$I_i^k = \sum_{j \in V \setminus \{i\}} x_{ji}^k \qquad i \in C_k \qquad \text{(B.17)}$$

$$I_i^k + O_j^k + z_{ij} - p_{ij}^k \leq 2 \qquad i \in N_k \setminus t_k, j \in N_k \setminus i \qquad \text{(B.18)}$$

$$I_i^k + I_j^k + z_{ij} - p_{ij}^k \leq 2 \qquad i \in N_k \setminus t_k, j \in C_k \qquad \text{(B.19)}$$

$$I_i^k + D_j^k + z_{ij} - p_{ij}^k \leq 2 \qquad i \in N_k \setminus t_k, j \in N_k \setminus \{i, s_k\} \qquad \text{(B.20)}$$

$$D_i^k \geq p_{il}^k + I_i^k + I_m^k + z_{ml} - p_{im}^k + O_l^k + D_l^k - 4 \qquad i \in N_k \setminus s_k, l \in N_k \setminus \{i, t_k\}, m \in N_k \setminus \{i, l, t_k\} \qquad \text{(B.21)}$$

$$D_i^k \geq p_{il}^k + I_i^k + O_m^k + D_m^k + z_{ml} - p_{im}^k + O_l^k + D_l^k - 4 \qquad i \in N_k \setminus s_k, m \in N_k \setminus \{i, t_k\}, l \in N_k \setminus \{i, m, t_k\} \qquad \text{(B.22)}$$

$$D_i^k \geq p_{il}^k + I_i^k + O_m^k + D_m^k + z_{lm} - p_{im}^k + O_l^k + D_l^k - 4 \qquad i \in N_k \setminus s_k, m \in N_k \setminus \{i, t_k\}, l \in N_k \setminus \{i, m, t_k\} \qquad \text{(B.23)}$$

$$D_i^k \geq p_{il}^k + I_i^k + O_m^k + D_m^k + z_{lm} - p_{im}^k + I_l^k - 4 \qquad i \in N_k \setminus s_k, m \in N_k \setminus \{i, t_k\}, l \in N_k \setminus \{i, m, t_k\} \qquad \text{(B.24)}$$

$$D_i^k \geq p_{il}^k + I_i^k + I_m^k + z_{lm} - p_{im}^k + I_l^k - 4 \qquad i \in N_k \setminus s_k, l \in N_k \setminus \{i, t_k\}, m \in C_k \qquad \text{(B.25)}$$

$$D_i^k \geq p_{il}^k + I_i^k + I_m^k + z_{lm} - p_{im}^k + O_l^k + D_l^k - 4 \qquad i \in N_k \setminus s_k, l \in N_k \setminus \{i, t_k\}, m \in C_k \qquad \text{(B.26)}$$

$$D_i^k \geq p_{il}^k + I_i^k + O_m^k + D_m^k + z_{ml} - p_{im}^k + I_l^k - 4 \qquad i \in N_k \setminus s_k, m \in N_k \setminus \{i, t_k\}, l \in C_k \qquad \text{(B.27)}$$

$$D_i^k \geq p_{il}^k + I_i^k + I_m^k + z_{ml} - p_{im}^k + I_l^k - 4 \qquad i \in N_k \setminus s_k, m \in N_k \setminus \{i, t_k\}, l \in C_k \qquad \text{(B.28)}$$

$$D_i^d \leq I_i^k \qquad i \in N_k \qquad \text{(B.29)}$$

$$x_{is_k}^k + p_{is_k}^k + x_{t_kj}^k + p_{t_kj}^k = 0 \qquad i \in V \setminus \{s_k\}, j \in V \setminus \{t_k\} \qquad \text{(B.30)}$$

$$I_i^k, O_i^k, x_{ij}^k, p_{ij}^k, D_i^k \in \{0,1\} \qquad i \in V, j \in V \setminus i \qquad \text{(B.31)}$$

Constraints (B.32)-(B.48) for each dependent rule $k \in K_D$:

$$\sum_{j \in V \setminus \{i\}} (u_{ij}^k + w_{ij}^k) - \sum_{j \in V \setminus \{i\}} (u_{ji}^k + w_{ji}^k) = y^k \qquad i = s_k \qquad \text{(B.32)}$$

$$\sum_{j \in V \setminus \{i\}} (u_{ij}^k + w_{ij}^k) - \sum_{j \in V \setminus \{i\}} (u_{ji}^k + w_{ji}^k) = -y^k \qquad i = t_k \qquad \text{(B.33)}$$

$$\sum_{j \in V \setminus \{i\}} (u_{ij}^k + w_{ij}^k) - \sum_{j \in V \setminus \{i\}} (u_{ji}^k + w_{ji}^k) = 0 \qquad i \in V \setminus \{s_k, t_k\} \qquad \text{(B.34)}$$

$$u_{ij}^k + w_{ij}^k + u_{ji}^k + w_{ji}^k \leq 1 \qquad i \in V, j \in V \setminus j \qquad \text{(B.35)}$$

$$u_{is_k}^k + w_{is_k}^k = 0 \qquad i \in V \setminus s_k \qquad \text{(B.36)}$$

$$u_{t_ki}^k + w_{t_ki}^k = 0 \qquad i \in V \setminus t_k \qquad \text{(B.37)}$$

$$u_{c_1c_2}^k + w_{c_1c_2}^k = 0 \qquad c_1 \in C_k, c_2 \in C_k \setminus c_1 \qquad \text{(B.38)}$$

$$u_{ij}^k + w_{ji}^k = 0 \qquad i \in C_k, j \in V \setminus i \qquad \text{(B.39)}$$

$$u_{ij}^k \leq z_{ij} \qquad i \in V, j \in V \setminus i \qquad \text{(B.40)}$$

$$w_{ij}^k \leq z_{ji} \qquad i \in V, j \in V \setminus i \qquad \text{(B.41)}$$

$$\sum_{j \in V \setminus \{i\}} u_{ji}^k + \sum_{j \in V \setminus \{i\}} w_{ij}^k \leq 1 \qquad i \in N_k \qquad \text{(B.42)}$$

$$\sum_{j \in V \setminus \{i\}} u_{ij}^k + \sum_{j \in V \setminus \{i\}} w_{ji}^k \leq 0 \qquad i \in C_k \qquad \text{(B.43)}$$

$$z_{ij} + z_{ji} = 0 \qquad i \in V_L, j \notin V_{L_i} \qquad \text{(B.44)}$$

$$z_{ji} = 0 \qquad i \in V_L, j \in V_{L_i} \qquad \text{(B.45)}$$

$$z_{ij} = z_{im} \qquad i \in V_L, \{j, m\} \in V_{L_i} \qquad \text{(B.46)}$$

$$y^k \in \{0,1\} \qquad \text{(B.47)}$$

$$w_{ij}^k, u_{ij}^k \in \{0,1\} \qquad i \in V, j \in V \setminus i \qquad \text{(B.48)}$$

$$z_{ij} \in \{0,1\} \qquad i \in V, j \in V \setminus i \qquad \text{(B.49)}$$