When Causal Inference Meets Graph Machine Learning: Unleashing the Potential of Mutual Benefit

---

A

Dissertation

Presented to

the faculty of the School of Engineering and Applied Science

University of Virginia

---

in partial fulfillment

of the requirements for the degree

Doctor of Philosophy

by

Jing Ma

August 2023

APPROVAL SHEET

This

Dissertation

is submitted in partial fulfillment of the requirements
for the degree of

Doctor of Philosophy

Author: Jing Ma

This Dissertation has been read and approved by the examing committee:

Advisor: Jundong Li

Advisor: Aidong Zhang

Committee Member: Yangfeng Ji

Committee Member: Hongning Wang

Committee Member: Anil Vullikanti

Committee Member: Sheng Li

Committee Member: Emre Kiciman

Committee Member:

Accepted for the School of Engineering and Applied Science:

Jennifer L. West, School of Engineering and Applied Science

August 2023

Abstract

Recent years have witnessed rapid development in graph-based machine learning (ML) in various high-impact domains (e.g., healthcare, recommendation, and security), especially those powered by effective graph neural networks (GNNs). Currently, the mainstream graph ML methods are based on statistical learning, e.g., utilizing the statistical correlations between node features, graph structure, and labels for node classification. However, statistical learning has been widely criticized for only capturing the superficial relations between variables in the data system, and consequently, rendering the lack of trustworthiness in real-world applications. For example, ML models often make biased predictions toward underrepresented groups. Besides, these ML models often lack explanation for humans. Therefore, it is crucial to understand the causality in the data system and the learning process. Causal inference is the discipline that aims to investigate the causality inside a system, for example, to identify and estimate the causal effect of a certain treatment (e.g., wearing a face mask) on an important outcome (e.g., COVID-19 infection). Involving the concepts and philosophy of causal inference into ML methods is often considered as a significant component of human-level intelligence and can serve as the foundation of artificial intelligence (AI). However, most traditional causal inference studies rely on strong assumptions and focus on independent and identically distributed (i.i.d.) data. Thus, most of them cannot be directly grafted on graphs. Therefore, causal inference on graphs is still faced with many unique barriers in effectiveness.

Fortunately, the interplay between causal inference and graph ML has the potential to bring mutual benefit to each other. In this thesis, we will present the challenges and our research contributions for bridging the gap between causal inference and graph ML. Our research aims to unleash the mutual benefit in these two areas, mainly including two key research perspectives: Q1) *How to leverage graph ML methods to facilitate causal inference in effectiveness?* Q2) *How to leverage causality to facilitate graph ML models in model trustworthiness (e.g., model fairness and explanation)?* Correspondingly, we introduce the background, challenges, and related work in Part I. In Part II, we introduce our detailed research problems and methodologies for causal inference on graph data powered by graph ML technologies (Q1). In Part III, we present our work in causality-involved trustworthy graph ML methods (Q2). In Part IV, we further introduce future research directions on causal machine learning, trustworthy AI, and graph mining, providing insights that manifest in real-world scenarios to facilitate future high-stakes applications.

Acknowledgements

This thesis and my whole research journey would not have been possible without the help and support of many people. I would like to express my sincere gratitude to my advisors Dr. Jundong Li and Dr. Aidong Zhang, for their invaluable guidance, support, and encouragement throughout my research work and academic life. Their expertise and insights have been essential in shaping my ideas and refining my work.

Also, I would like to thank other members of my thesis committee Dr. Yangfeng Ji, Dr. Hongning Wang, Dr. Emre Kiciman, Dr. Anil Vullikanti, and Dr. Sheng Li, for their time and input in reviewing my work and providing valuable feedback that helped me improve the quality of my research.

I am grateful to my mentors and collaborators in my summer internships at Microsoft Research (MSR). I have been incredibly fortunate to have the opportunity to work with and learn from Dr. Emre Kiciman and Dr. Sergii Babkin in Summer 2022, and Dr. Mengting Wan, Dr. Longqi Yang, Dr. Brent Hecht, and Dr. Jaime Teevan in Summer 2021.

My appreciation also goes out to my collaborators Dr. Ruocheng Guo, Dr. Chen Chen, Dr. Dezhi Hong, Dr. Daniel Mietchen, Dr. Saumitra Mishra, Yushun Dong, Song Wang, Zheng Huang, Yaochen Zhu, who provided an inclusive and collaborative research environment that enabled me to grow both professionally and personally.

I am incredibly grateful to my family and friends, who provided unwavering support and encouragement throughout my academic journey.

Finally, I would like to acknowledge the funding agencies National Science Foundation (NSF), J.P. Morgan Chase, and the Department of Computer Science at University of Virginia, whose financial support made my research possible.

Thank you all for your contributions to my academic and personal growth.

Contents

List of Figures

List of Tables

Part I

Background

CHAPTER 1

Introduction

1.1 Graphs and Graph Machine Learning

1.1.1 What is Graph Machine Learning?

In the past few decades, graphs (i.e., networks) have been a widespread and essential technique for modeling a variety of real-world systems that are composed of interconnected units, such as road networks [1], social networks [2], professional networks [3], economic networks [4], and molecular graphs [5]. In many high-impact domains, there has been a significant advancement in graph-based machine learning (ML), which has emerged as an effective tool for uncovering patterns and structures in these complex data and has shown promising results in various tasks, including node classification [6], link prediction [7], graph classification [8], community detection [9], and graph clustering [10]. Especially, in recent years, a large number of powerful graph neural networks (GNNs) have emerged to facilitate these graph ML tasks. Representative GNNs include graph convolutional network [11], graph variational autoencoder [12], graph attention network [13], graph transformer [14], and so on.

1.1.2 Limitations of Graph Machine Learning

Despite the success achieved by graph ML models, many of them still often make unreliable or incorrect predictions or decisions, which can have serious consequences in real-world important applications such as healthcare, finance, and criminal justice. Therefore, it is essential to address these issues by ensuring that the models are transparent, fair, and reliable, and that they adhere to ethical and legal standards, i.e., improve the trustworthiness of graph ML models.

What is a trustworthy graph machine learning model? Although there has not been a unified and formal definition for model trustworthiness, in general, a trustworthy graph machine learning model is one that not only produces accurate and reliable predictions, but also exhibits transparency and interpretability. It should be free from biases and demonstrate

consistent performance across different datasets and scenarios. Additionally, it should be able to provide explanations for its predictions, allowing for easy verification and evaluation of its performance. This type of model should also abide by ethical and fairness standards to ensure that its predictions do not discriminate against certain individuals or groups. Ultimately, a trustworthy graph machine learning model should inspire confidence and trust in its predictions. Trustworthiness plays an increasingly important role in many high-stakes decision-making processes, such as medical diagnoses, financial predictions, and legal judgments. In these scenarios, the decisions made by the models can have significant consequences for individuals and society.

The current mainstream graph ML methods primarily rely on learning from statistical correlations. More specifically, they utilize the statistical correlations between node features, graph structures, and labels to perform different tasks. However, statistical correlations have been widely criticized for only capturing the superficial relations between variables in the data system, and consequently, often rendering the lack of trustworthiness in many real-world applications. Therefore, incorporating causality instead of simple statistical correlations into graph ML is important for improving model trustworthiness.

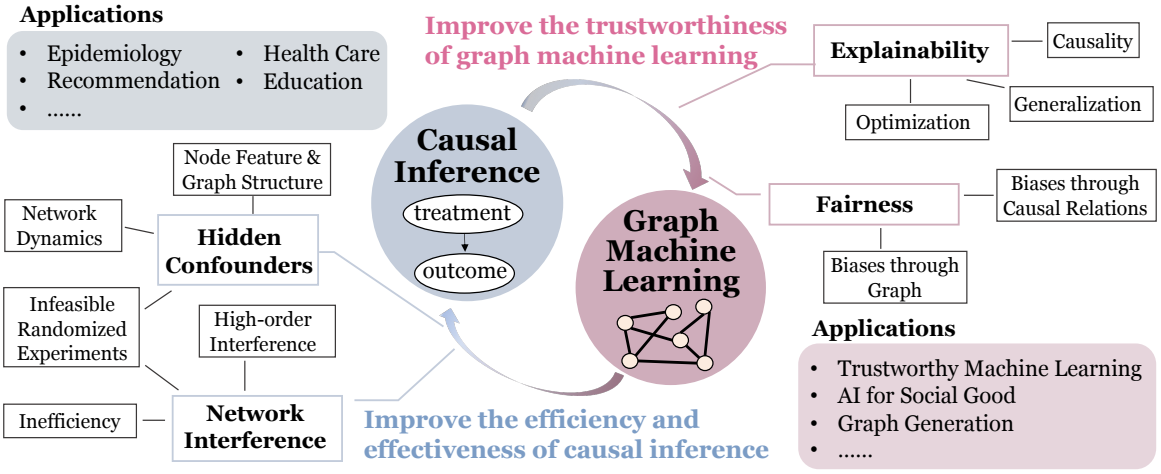


FIGURE 1.1. An overview of my research work.

1.2 Causal Inference and Graph

1.2.1 What is Causal Inference?

Causal inference [15] is a discipline that investigates the causality within a system, enabling estimation of the causal effect of a specific cause or also known as *treatment* (e.g., wearing a face mask) on an *outcome* (e.g., COVID-19 infection). Incorporating the concepts and

philosophy of causal inference into ML methods is considered a crucial component of human-level cognition and can serve as the foundation of artificial intelligence (AI). For example, to evaluate the impact of a face mask requirement policy on mitigating the spread of COVID-19, it is necessary to assess the causal effect of this policy rather than relying solely on correlations. Classical causal inference theory is built upon two main frameworks: Pearl's structural causal model (SCM) [15] and Rubin's potential outcome framework [16]. Here we introduce some important concepts for causal inference:

DEFINITION 1. *(Structural Causal Model) A structural causal model (SCM) [15] is denoted by a triple (U, V, F): U is a set of exogenous variables, and V is a set of endogenous variables. The structural equations $F = \{F_1, ..., F_{|V|}\}$ determine the value for each $V_i \in V$ with $V_i = F_i(\mathtt{PA}_i, U_i)$, here $\mathtt{PA}_i \subseteq V \backslash V_i$ denotes the "parents" of V_i, and $U_i \subseteq U$.*

DEFINITION 2. *(Treatment and outcome) In causal inference, a treatment is an intervention or an action assigned to each unit, and an outcome is a variable of interest which is causally influenced by the treatment. Usually, a treatment is a binary variable with value as 1 or 0, and the outcome $Y \in \mathbb{R}$.*

DEFINITION 3. *(Potential outcome) Potential outcome refers to the hypothetical outcome of a unit (e.g., an individual or an instance) under a certain treatment assignment. It assumes that each unit has a potential outcome for each possible treatment option, but only one of these potential outcomes can be observed in reality, depending on the actual treatment received. For unit i with treatment 1 or 0, we denote its corresponding potential outcomes as $Y_i(1)$ and $Y_i(0)$, respectively. The subscript $(\cdot)_i$ can be omitted if it does not refer to any specific unit.*

DEFINITION 4. *(Treatment effect) Treatment effect, or also known as causal effect of a specific treatment on an outcome is the impact of manipulating the treatment on the outcome for one of a group of units.*

- *(Individual treatment effect) For one single unit i, the individual treatment effect (ITE) is formally defined as the difference between the two potential outcomes:*

$$\tau_i = Y_i(1) - Y_i(0). \tag{1.1}$$

- *(Conditional average treatment effect) Conditional average treatment effect (CATE) is the average causal effect of a treatment for a specific subgroup of units with a particular set of observed characteristics $X = \mathbf{x}$:*

$$\tau(\mathbf{x}) = \mathbb{E}[Y(1) - Y(0)|X = \mathbf{x}]. \tag{1.2}$$

- *(Average treatment effect) Average treatment effect (ATE) is the average causal effect of a treatment for all the units:*

$$\tau_{ATE} = \mathbb{E}[Y(1) - Y(0)]. \tag{1.3}$$

In the past, causal effect estimation relied on experimental design or randomized controlled trials (RCTs) [17, 18] that involve manipulating a treatment variable and observing its impact on the outcome. However, in many cases, conducting experiments may be impractical or unethical [18], and researchers must rely on observational data. In treatment effect estimation from observational data, *confounder* is one of the major challenges. Confounders are variables that causally influence both the treatment and the outcome, which lead to non-causal associations between the treatment and the outcome. Therefore, the presence of confounders can often bring confounding biases for treatment effect estimates. To obtain unbiased and reliable estimates of treatment effects, it is essential to mitigate confounding biases by appropriately adjusting for confounders in the analysis. Researchers and practitioners must be aware of the potential presence of confounders in their datasets and take appropriate measures to minimize their impact on the results of the analysis.

1.2.2 Causal Inference on Graphs

Causal inference on graphs is a crucial area of research with many real-world applications and has recently attracted a lot of attention. For instance, it can help in understanding the impact of social and economic policies on individuals with connections. It can also be used to evaluate the effectiveness of medical treatments and interventions in a system with relational information (e.g., physical contact network). To address the aforementioned issues of graph machine learning, it is essential to incorporate causality into the learning process. However, traditional causal inference studies [15, 19] often rely on strong assumptions and focus on independent and identically distributed (i.i.d.) data. In this context, causal inference on graphs encounters many challenges in terms of effectiveness, creating a gap between causal inference and graph machine learning. Despite this, if looking at it from a different perspective, the graph structure can also provide additional clues and benefits for causal learning.

1.3 Research Goal, Challenges, and Contributions

The primary goal of my research is to bridge the gap between causal inference and graph ML, aiming to unleash the mutual benefit from each other. This dissertation mainly includes two directions: 1) facilitate causal inference on graph data with graph ML methods; 2) facilitate graph ML models by leveraging causal inference. Fig. 1.1 shows an overview of my research work. Representative research questions in this area include: How to leverage the graph structure among units for causal inference when randomized experiments are infeasible? How to handle the interference between connected units? How to improve the

trustworthiness (e.g., fairness, explainability) of graph ML models from a causal view? To answer these questions, we need to address the following fundamental challenges:

- (C1): *Hidden confounders* - Classical causal inference methods are based on the unconfoundedness assumption (a.k.a. strong ignorability assumption) [20], which assumes that hidden confounders (unobserved variables that influence both treatment and outcome) do not exist. However, in the real world, hidden confounders widely exist and are often time-varying together with data dynamics [21]. This phenomenon can cause serious confounding biases in causal effect estimation, thus it necessitates sophisticated tools to mitigate the influence of hidden confounders.
- (C2): *Network interference and entanglement* - Traditional causal inference methods assume that different units do not interfere or entangle with each other. These assumptions are often impractical in real-world graphs, where interference and entanglement are ubiquitous among connected units [22]. Conventional causal inference methods are often insufficient or inefficient to characterize such relations.
- (C3): *Fairness* - The predictions of ML models are often biased towards certain demographic groups w.r.t. sensitive attributes (e.g., age, gender, religion, disability) [23]. In graphs, such biases can also be induced by one's neighboring nodes, as well as the causal relations between variables. These make existing fair ML algorithms incapable of mitigating such biases without consideration of graph connections and causality.
- (C4): *Explainability* - Most graph ML models are opaque and lack explainability. Although preliminary studies [24, 25] have explored to generate explanations for graph ML models, most of these methods are still limited in optimization, generalization, and especially, consistency with the underlying causality.

My research work mainly covers different aspects of the aforementioned two directions by tackling the challenges (C1, C2, C3, and C4). Specifically, we study the following topics and develop corresponding principled algorithms: (1) **Graph ML for causal inference**, including a) causal effect estimation under hidden confounders; and b) causal effect estimation under interference; and c) causal effect estimation under treatment entanglement; (2) **Causality for Graph ML**, including d) counterfactual fairness in graph node representation learning; and e) counterfactual explanation for graph ML models. Furthermore, we investigate real-world downstream applications such as epidemiology.

CHAPTER 2

Related Work

In recent years, there have been continuously increasing attention in the research domains relevant to this dissertation. In this section, we review the related work including the following main categories: causal inference (including i.i.d. data and graph data), trustworthy graph learning and causality.

2.1 Causal Inference

2.1.1 Causal Inference on i.i.d. Data

Causal inference is a field of study that focuses on understanding the causal relationships between variables in a system and identifying how changes to one variable can affect another. Two main categories of causal inference are *causal discovery* (i.e., analyzing data to infer the underlying causal structure of the system), which can be represented as a directed acyclic graph (DAG) [15] and *causal effect estimation* (i.e., quantifying the effect of a treatment on an outcome). Causal inference plays a critical role in numerous fields, such as economics, medicine, and engineering.

Causal discovery. Many traditional methods of causal discovery rely on graph-based models, such as Bayesian networks or causal Bayesian networks, which aim to capture the dependencies between variables in the data. In recent years, there has been increasing interest in developing more scalable and efficient algorithms for causal discovery. These include constraint-based methods [26, 27] that utilize statistical tests to identify causal relationships between variables, and score-based methods [28] that use optimization techniques to identify the most likely causal structure given the data. Additionally, there has been a growing trend towards combining causal discovery with machine learning methods, such as deep neural networks, to learn causal representations from high-dimensional data [29]. These developments in causal discovery have the potential to advance many fields, including medicine, economics, and social sciences, by enabling more accurate and effective decision-making based on causal relationships.

Causal effect estimation. Causal effect estimation is another important area in causal inference that has attracted considerable attention from researchers. Common approaches for causal effect estimation include propensity score matching or weighting [30, 31] and distribution balance of confounding variables between the treatment and control groups [32, 33]. Other methods include regression adjustment, instrumental variable estimation [34, 35], difference-in-differences estimation [36], and so on. Recently, more researchers develop deep learning-based approaches [37, 38] for causal effect estimation, which aim to learn complex non-linear relationships between variables and achieve better performance. Several studies [39, 40] have also investigated the use of reinforcement learning and counterfactual reasoning for causal effect estimation. Despite significant progress in this area, there are still many challenges and limitations, such as the curse of dimensionality and the potential for bias in model selection. Besides, most of existing causal inference studies are limited in i.i.d. data and cannot handle other data types such as graphs.

2.1.2 Causal Inference on Graphs

Causal effect estimation on graphs. There have been many efforts made for causal effect estimation on graph data in recent years. A line of works leverage graph structure as a proxy for hidden confounders. Among them, Guo et al. design a network deconfounder (NetDeconf) [41] for individual treatment effect (ITE) estimation on graphs. Network deconfounder is implemented based on a graph convolution network (GCN) [11]. Another work [42] proposes a minimax game based ITE estimator (IGNITE) which enables ITE estimation on graph data at both individual level and group level. Chu et al. [43] develop a graph infomax adversarial learning model (GIAL) for treatment effect estimation on graph observational data. GIAL identifies the patterns of hidden confounders by fully leveraging the graph information and recognizing the imbalance in graph structure. Another group of studies [44, 45, 46, 47, 48] focuses on treatment effect estimation under network interference. Many of these studies also rely on (graph) neural network techniques. Besides, different from traditional causal inference with binary treatment assignments, a few recent explorations [49, 50] study the problem of treatment effect estimation with graph-structured treatments.

Causal discovery with graphs. Apart from causal effect estimation on graphs, another important research category in causal inference is causal discovery [51, 52], which targets on identifying the causal relationships between variables and recovering the underlying causal model. Traditional causal discovery approaches include 1) conditional independence constraint-based methods such as PC algorithm [26] and Fast Causal Inference (FCI) [27], and 2) score-based methods such as Greedy Equivalence Search (GES) [28]. Noticeably, graphical models have been widely used to represent causal relationships among variables, where the

graph structure represents the dependencies among variables and the edge direction indicates the causal relationship. With the recent progress in GNNs and the natural connection between graphs and causal structure, more cutting-edge studies have utilized GNNs to facilitate causal discovery [53, 54, 55, 56].

2.2 Trustworthy Graph Learning and Causality

Trustworthy graph learning. Graph learning has become an increasingly popular area of research due to its effectiveness in modeling complex relationships between data points in a variety of applications. However, with the growing use of graph learning in high-stakes decision-making processes, there is a growing need for trustworthy models that are reliable, transparent, and fair. In recent years, researchers have begun to explore to improve the trustworthiness of graph learning models.

In general, these works can be summarized in the following main groups:

- **Generalization**: a large number of studies aim to improve the generalization of graph ML models [57], i.e., improving the model ability to accurately predict outcomes for unseen data. These studies include different branches such as graph data augmentation [58, 59], graph invariant learning [60, 61], graph disentangled learning [62, 63], graph adversarial training [64, 65], self-supervised graph learning [66, 67], and causality-based graph learning [68, 69, 70].
- **Fairness**: many existing graph algorithms lack consideration for fairness and tend to exhibit biases against certain demographic populations. It has been shown that these biases are ubiquitous and easily amplified by graph structure [71]. Fairness of graph learning models can be defined in different ways, including group fairness [72, 73, 74], individual fairness [75, 76, 77], degree-related fairness [78, 79], and application-specific fairness [80, 81, 82].
- **Explanation**: Explanation in graph ML aims to provide human-interpretable reasons or justifications for a model's predictions. Especially, when ML models become increasingly complex and are used in high-stakes applications, the need for model explanation has become more pressing. Recently, a large number of efforts have been made for explanation in graph ML models [83], including two main groups: model-level explanations [84] and instance-level explanations. Instance-level methods include gradients or features-based approaches [85, 86], perturbation-based methods [24, 87, 88, 89], decomposition methods [90, 91], and surrogate methods [92, 93].

Causality in trustworthy graph learning. Causality plays a crucial role in graph learning. Unlike other methods, causality-based techniques enable us to gain a deeper understanding of the complex causal relationships between variables and their effects on each other. Merely observing correlations between variables may lead to incorrect assumptions and erroneous conclusions. Therefore, incorporating causality into graph learning is critical for obtaining accurate insights and making trustworthy decisions. In recent years, there has been a surge of research aimed at enhancing traditional graph learning with causality-based approaches. Among them, a lot of research works improve the generalization of graph ML models [70, 94, 61, 95] by capturing the causal features in graph data and mitigating the effects of biases introduced by spurious correlations. Moreover, numerous studies [96, 97, 98, 99] are dedicated to enhancing the explainability of graph ML models through a causal lens. Additionally, as there is growing concerns about the fairness of AI towards underrepresented groups, there has been a surge of interest in promoting fairness in graph learning by examining the causal relations between sensitive attributes (e.g., gender) and other variables [100, 101].

Part II

Improve Causal Inference on Graph Data with Graph Learning Techniques

CHAPTER 3

Overview of Part II

In Part II, we present our work on utilizing machine learning techniques to the problem of causal inference on graph data. Specifically, we focus on addressing the aforementioned multifaceted challenges (hidden confounders, network interference, and treatment entanglement) in this area.

In Chapter 4, we handle the issue of hidden confounders in a dynamic environment by designing a novel approach that leverages the graph structure to infer and control for these confounders. In this way, the confounding biases in treatment effect estimation can be effectively eliminated. We also demonstrate how this method can be applied to real-world problems, such as the assessment of COVID-19 policies.

In Chapter 5, we address the challenge of network interference, even on complicated graphs such as hypergraphs, where we consider the treatment effect estimation problem when high-order interference exists. In this case, causal effects could be propagated through group interactions. This study has a wide spectrum of applications in those scenarios with group connections (i.e., connections among more than two units), such as mass gatherings, team projects, and living communities.

In Chapter 6, we address the challenge of treatment entanglement, where treatments are entangled together through network connections. This scenario is ubiquitous in real-world graphs such as room sharing and information dissemination. We investigate this problem from different angles.

By developing solutions for these challenges, we aim to contribute to the development of effective causal inference methods for graph data, and enhance their practical applicability in real-world network-related scenarios.

CHAPTER 4

Causal Effect Estimation with Hidden Confounders on Dynamic Graphs

---

As aforementioned, traditional methods for causal effect estimation [37, 102, 103] often heavily rely on the strong ignorability assumption, also known as the unconfoundedness assumption [20], which assumes the absence of unobserved confounders. Unfortunately, this assumption is often unrealistic in real-world applications. For instance, when estimating the effect of taking medication on an individual's health, their socioeconomic status could be a hidden confounder that affects both their health and medication choice, yet it is not typically observed. This can lead to biased causal effect estimates. In recent years, various techniques [38, 104] have been proposed to relax the strong ignorability assumption by incorporating latent confounders. However, these approaches require the ability to extract latent confounders from observational data features using neural networks or factor models.

However, the role of network structures in deconfounding has been often overlooked in traditional literature, with few studies [41] recognizing its importance and leveraging it for treatment effect estimation. However, the graph topology among units is prevalent in various types of observational data, such as social networks of people, electrical grids of power stations, and spatial networks of geometric objects. Moreover, in situations where confounders are difficult to measure, an alternative approach is to capture their patterns and control their impact by leveraging the network structure. In the previous example, a patient's social network patterns may indicate their socioeconomic status. An early exploration utilizes this fact and proposes a method called network deconfounder [41], which uses the network structure and the observed node features to minimize confounding bias in treatment effect estimation. Here, the graph structure and observed features act as proxies for the hidden confounders.

Despite the fact that the graph topology can serve as a source of proxies for hidden confounders, most existing studies [41, 105] overwhelmingly assume that the observational graph data and the hidden confounders are static. In fact, all variables are naturally dynamic in many real-world occasions. In the previous example, patients' socioeconomic status and social network can be varying over time. Another example is, when estimating the treatment effect of wearing a face mask on COVID-19 infection, the residents' vigilance may be a hidden

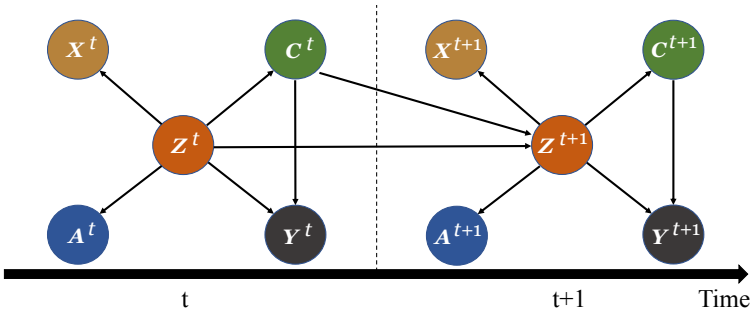


FIGURE 4.1. Causal graph for the studied problem. At time t, we use $\mathbf{X}^t$, $\mathbf{A}^t$, $\mathbf{Z}^t$, $\mathbf{C}^t$, $\mathbf{Y}^t$ to denote the features of observational data, relations among observational data, representations of hidden confounders, treatment assignment, and outcomes, respectively. The hidden confounders $\mathbf{Z}^{t+1}$ at $t+1$ causally affect the treatment assignment $\mathbf{C}^{t+1}$ and the outcome $\mathbf{Y}^{t+1}$ at that time. To infer $\mathbf{Z}^{t+1}$, we can leverage the networked observational data $\mathbf{X}^{t+1}$ and $\mathbf{A}^{t+1}$ at $t+1$, previous hidden counfounders $\mathbf{Z}^t$, and treatment assignment $\mathbf{C}^t$. The black arrows represent causal relations.

confounder that cannot be explicitly measured, but it may be reflected in residents' mobility network. Noticeably, as time goes on, the mobility network, the face mask practice, the COVID-19 infection risk, and the residents' vigilance are all time-varying at different time periods. In this case, the residents' vigilance can be influenced by the situation in previous time periods. For example, the recent number of death cases would affect people's vigilance in next a few days. In these scenarios, it is important to study the problem of deconfounding with observational graph data in a time-varying environment.

For this problem, we propose a dynamic graph neural network-based framework DNDC [21] to estimate causal effects under a dynamic networked environment. Generally, DNDC learns representations of confounders at each time period by encoding the dynamic graph data (including the current graph and historical information) into the representation space. DNDC systematically models the evolution patterns of different data modalities for unbiased ITE estimation. Specifically, DNDC uses a recurrent neural network (RNN) [106, 107] to capture the temporal information, and adopts a graph convolutional network (GCN) [11] based module to handle the relational information. ITE estimation in a dynamic network has a wide range of applications, such as epidemiology, economics, and recommendation across different time periods.

4.1 Problem Definition

Suppose a dataset with time-evolving networked observational data across T different time periods is given, denoted by $\{\mathbf{X}^t, \mathbf{A}^t, \mathbf{C}^t, \mathbf{Y}^t\}_{t=1}^T$. Here, units (instances) are connected as nodes in a dynamic network, and $(\cdot)^t$ denotes the t-th time period. $\mathbf{X}^t = \{\mathbf{x}_1^t, ..., \mathbf{x}_{n^t}^t\}$ stands for the node attributes (features) at time period t. $\mathbf{x}_i^t$ represents the node features of the i-th instance (e.g., features of each person), n^t is the number of nodes, and $\mathbf{A}^t$ is the adjacency matrix of the network (e.g., people's contact network). For simplicity, the network is assumed to be undirected and unweighted, but this work can be naturally extended to more general cases such as directed and weighted networks. At time period t, the treatment for these n^t nodes is denoted by $\mathbf{C}^t = \{c_1^t, ..., c_{n^t}^t\}$, where c_i^t is either 1 or 0 (e.g., if a person wears face mask or not). The observed outcome of all instances at time period t is denoted by $\mathbf{Y}^t = \{y_1^t, ..., y_{n^t}^t\}$ (e.g., COVID-19 infection). $\mathbf{Z}^t = \{\mathbf{z}_1^t, ..., \mathbf{z}_{n^t}^t\}$ stands for the hidden confounders (e.g., people's vigilance). The superscript $(\cdot)^{<t}$ denotes the historical data before time period t. For example, all the node features before time period t can be referred to as $\mathbf{X}^{<t} = \{\mathbf{X}^1, \mathbf{X}^2, ..., \mathbf{X}^{t-1}\}$, and $\mathbf{C}^{<t}, \mathbf{A}^{<t}$ are defined similarly. $\mathbf{H}^t = \{\mathbf{X}^{<t}, \mathbf{A}^{<t}, \mathbf{C}^{<t}\}$ denotes all the historical data before time period t. The causal graph in this problem is shown in Fig. 4.1.

This work is based on the widely-adopted potential outcome framework [20, 108]. The potential outcome of the i-th node under treatment c at time period t is denoted by $y_{c,i}^t \in \mathbb{R}$, which is the outcome that would occur if instance i had received treatment c at time period t. We represent the potential outcomes of all instances at time period t by $\mathbf{Y}_1^t = \{y_{1,1}^t, ..., y_{1,n^t}^t\}$ and $\mathbf{Y}_0^t = \{y_{0,1}^t, ..., y_{0,n^t}^t\}$, corresponding to treatment 1 and 0, respectively. Then the individual treatment effect (ITE) on time-varying observational graph data can be defined as:

$$\tau_i^t = \tau^t(\mathbf{x}_i^t, \mathbf{H}^t, \mathbf{A}^t) = \mathbb{E}[y_{1,i}^t - y_{0,i}^t | \mathbf{x}_i^t, \mathbf{H}^t, \mathbf{A}^t]. \tag{4.1}$$

Based on the above definition of ITE, the average treatment effect (ATE) at time period t is defined as $\tau_{ATE}^t = \frac{1}{n^t} \sum_{i=1}^{n^t} \tau_i^t$.

The studied problem of learning ITE with dynamic observational graph data is defined as follows:

DEFINITION 5. *(Learning ITE on Dynamic Observational Graph Data). Given the dynamic observational graph data $\{\mathbf{X}^t, \mathbf{A}^t, \mathbf{C}^t, \mathbf{Y}^t\}_{t=1}^T$ across T different time periods, the goal is to estimate the ITE τ_i^t for each instance i at each time period t.*

Most existing works [37, 102, 103] rely on the *strong ignorability* assumption [109], assuming that observed features can contain all the confounders, i.e., no hidden confounders exist. The formal definition of this assumption is as follows:

DEFINITION 6. *(Strong Ignorability Assumption). Given an instance's observed features, the potential outcomes of this instance are independent of its treatment assignment:* $y_{1,i}^t, y_{0,i}^t \perp\!\!\!\perp c_i^t | \mathbf{x}_i^t$.

However, this assumption is often untenable due to the existence of hidden confounders in many real-world scenarios [110]. Our method relaxes this assumption as there exist hidden confounders $\mathbf{Z}^t$ at each time stamp t which causally influence the treatment $\mathbf{C}^t$ and the potential outcomes ($\mathbf{Y}_1^t$ and $\mathbf{Y}_0^t$). Conditioning on $\mathbf{Z}^t$, the treatment assignment is randomized, i.e., $y_{1,i}^t, y_{0,i}^t \perp\!\!\!\perp c_i^t | \mathbf{z}_i^t$. We aim to learn the representations of hidden confounders for unibased ITE estimation based on the following relaxed assumption regrading hidden confounders:

ASSUMPTION 1. *(Existence of Hidden Confounders) (i) The hidden confounders may not be accessible, but we assume that the instance features and network structures are both correlated with hidden confounders, and can be considered as proxy variables for hidden confounders. (ii) Hidden confounders at each time stamp can be influenced by historical information, such as the hidden confounders and treatment assignment in previous time stamps.*

Based on the above assumption, we now show the identification result of this problem. For simplicity, we drop the instance index i for notations $\mathbf{z}^t, \mathbf{x}^t, y^t, c^t$:

THEOREM 1. *(Identification of ITE) If we recover* $p(\mathbf{z}^t | \mathbf{x}^t, \mathbf{H}^t, \mathbf{A}^t)$ *and* $p(y^t | \mathbf{z}^t, c^t)$*, then the proposed DNDC can recover the ITE under the causal graph in Fig. 4.1.*

The detailed proof is in Appendix A.

4.2 Proposed Method

We propose a framework DNDC [21] for ITE estimation in dynamic networked data. The overall structure of DNDC, as shown in Fig. 4.2, is composed of three key elements: confounder representation learning, potential outcome and treatment prediction, and representation balancing. The DNDC model captures hidden confounders over time by mapping current networked observational data and historical information into a latent representation space. The learned representations are then used for predicting potential outcomes and treatments. To ensure the balance between the representations of hidden confounders in the treatment group and the control group, an adversarial learning-based balancing technique is developed.

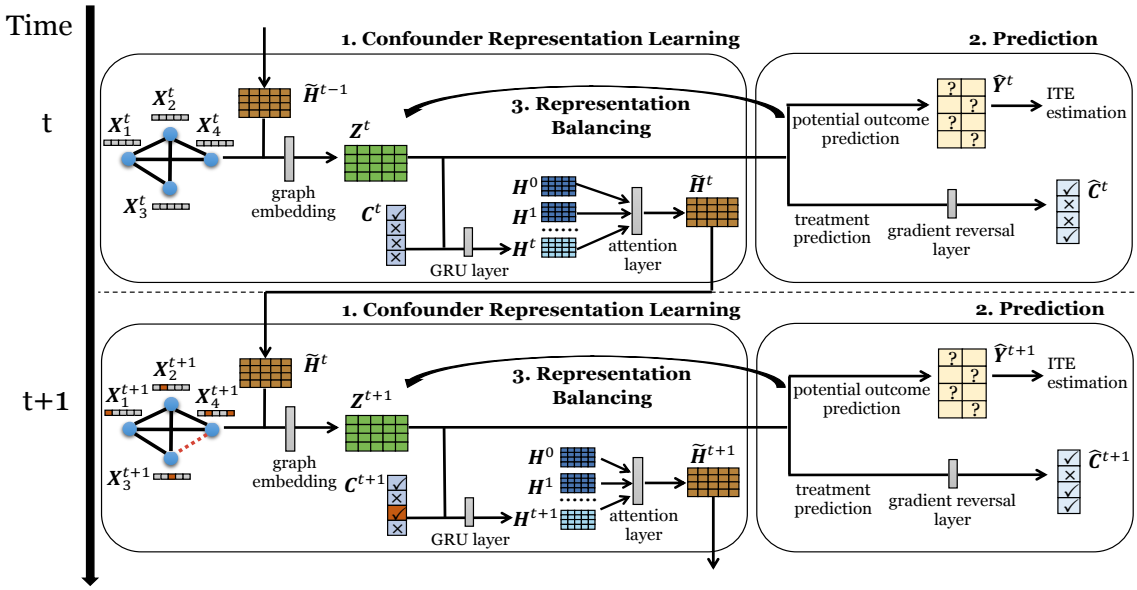


FIGURE 4.2. An illustration of the framework DNDC.

4.2.1 Confounder Representation Learning

As the hidden confounders are related to node features, graph structure, as well as the historical information, DNDC leverages all of them in confounder representation learning. More specifically, to well handle the graph data, graph convolutional networks (GCNs) [11] are used in this process:

$$\mathbf{z}_i^t = g(([\mathbf{X}^t, \tilde{\mathbf{H}}^{t-1}])_i, \mathbf{A}^t) = \hat{\mathbf{A}}^t \text{ReLU}((\hat{\mathbf{A}}^t[\mathbf{X}^t, \tilde{\mathbf{H}}^{t-1}])_i \mathbf{U}_0)\mathbf{U}_1, \tag{4.2}$$

where $g(\cdot)$ is a learnable transformation function parameterized by GCNs. In the above equation, two GCN layers (with parameters $\mathbf{U}_0$ and $\mathbf{U}_1$, respectively) are stacked to capture the non-linear dependency between the hidden confounders and the input, but the framework itself does not have any restriction regarding the number of GCN layers. To leverage the data in previous time periods, a historical embedding $\tilde{\mathbf{H}}^{t-1} \in \mathbb{R}^{n^t \times d_h}$ is learned to encode the historical information before time period t, including previous hidden confounders and treatment assignment. d_h is the dimension of historical embedding. Here, $[\cdot, \cdot]$ stands for the concatenation operation and $(\cdot)_i$ represents the i-th row of the matrix. $\mathbf{z}_i^t \in \mathbb{R}^{d_z}$ denotes the representation of confounders for instance i at time period t, d_z is the dimension of confounder representation. $\hat{\mathbf{A}}^t$ is the normalized adjacency matrix computed from $\mathbf{A}^t$ with the renormalization trick [11].

To enable the historical embedding to characterize the evolution patterns of dynamic networked data, a gated recurrent unit (GRU) [111] based memory unit is used. Specifically, in

the GRU, the current information ($\mathbf{Z}^t$, $\mathbf{X}^t$, $\mathbf{C}^t$) and previous hidden state $\mathbf{H}^{t-1}$ are embedded into a new hidden state $\mathbf{H}^t \in \mathbb{R}^{n^t \times d_h}$: $\mathbf{H}^t = \text{GRU}(\mathbf{H}^{t-1}, [\mathbf{Z}^t, \mathbf{X}^t, \mathbf{C}^t])$. An attention mechanism [112, 113] among different hidden states of GRU is adopted to model the importance of the historical influence from different time periods. For any node with hidden state $\mathbf{h}^t \in \mathbb{R}^{d_h}$ at time period t, the attention weight $\alpha_{t,s}$ that models the importance of the hidden states of GRU from time period s on those of time period t ($s < t$) can be calculated with different attention score functions (e.g., bilinear [112] function or the scaled dot product [113] function) on $\mathbf{h}^t$ and $\mathbf{h}^s$. Then $\tilde{\mathbf{h}}^t = \text{MLP}([\mathbf{h}^t, \sum_{s=1}^{t-1} \alpha_{t,s}\mathbf{h}^s])$, and they form a matrix $\tilde{\mathbf{H}}^t$ with all instances.

4.2.2 Outcome and Treatment Prediction

Based on the learned confounder representations, DNDC predicts the potential outcomes with two learnable functions $f_1, f_0 : \mathbb{R}^{d_z} \to \mathbb{R}$, corresponding to the two cases when treatment is 1 or 0, i.e.,

$$\hat{y}^t_{1,i} = f_1(\mathbf{z}^t_i),\ \hat{y}^t_{0,i} = f_0(\mathbf{z}^t_i). \tag{4.3}$$

For each instance i, both of its *factual outcome* $y^t_{F,i}$ and *counterfactual outcome* $y^t_{CF,i}$ (unobserved outcome with the treatment different from reality) are predicted. The loss function of the potential outcome prediction is formulated as:

$$\mathscr{L}_y = \mathbb{E}_{t\in[T],i\in[n^t]}[(\hat{y}^t_{F,i} - y^t_{F,i})^2]. \tag{4.4}$$

To better learn the confounder representations, DNDC also uses treatment as supervision. The loss function of treatment prediction is:

$$\mathscr{L}_c = -\mathbb{E}_{t\in[T],i\in[n^t]}[(c^t_i \log(\hat{s}^t_i) + (1 - c^t_i)\log(1 - \hat{s}^t_i))]. \tag{4.5}$$

The treatment predictor takes confounder representations as input. It is implemented with an MLP module and a softmax layer. $\hat{s}^t_i$ is the output of the softmax layer, which can be considered as the prediction of propensity score for instance i at time period t: $\hat{s}^t_i = \text{softmax}(\text{MLP}(\mathbf{z}^t_i))$.

4.2.3 Representation Balancing

It has been shown in previous literature [37] that minimizing the discrepancy between the confounder representation distribution of the treatment group and that of the control group can benefit causal effect estimation. DNDC uses a gradient reversal layer [114] for representation balancing. The gradient reversal layer does not change the input during forward-propagation, but during back-propagation, it reverses the gradient by multiplying it by a negative scalar.

In this way, the gradient reversal layer can (1) train the treatment predictor by minimizing the treatment prediction loss $\mathscr{L}_c$; and (2) enable representation balancing via maximizing $\mathscr{L}_c$ w.r.t. the model parameters of the confounder representation learning.

To show how the proposed adversarial learning based balancing method works in the training process, we write the gradient updates that happen while minimizing Eq. (4.7) as:

$$\begin{aligned}
\theta_z &\leftarrow \theta_z - \mu(\frac{\partial \mathscr{L}_y}{\partial \theta_z} - \omega \frac{\partial \beta \mathscr{L}_c}{\partial \theta_z} + 2\gamma\theta_z), \\
\theta_c &\leftarrow \theta_c - \mu(\frac{\partial \beta \mathscr{L}_c}{\partial \theta_c} + 2\gamma\theta_c), \\
\theta_y &\leftarrow \theta_y - \mu(\frac{\partial \mathscr{L}_y}{\partial \theta_y} + 2\gamma\theta_y),
\end{aligned} \tag{4.6}$$

where θ_z, θ_c, and θ_y are the model parameters of the hidden confounder representation learning, the treatment prediction, and the potential outcome prediction, respectively. When updating θ_z, the gradient backpropagated from the treatment predictor is reversed by multiplying with a negative constant $-\omega$. The positive real scalar μ stands for the learning rate of the optimization process.

4.2.4 Loss Function

The overall loss function is formulated as follows:

$$\mathscr{L}\{\{\mathbf{x}_i^t, y_i^t, c_i^t\}_1^{n^t}, \mathbf{A}^t\}_1^T = \mathscr{L}_y + \beta\mathscr{L}_c + \gamma||\mathbf{\Theta}||^2, \tag{4.7}$$

where $\mathbf{\Theta}$ is the set of parameters in this framework, and $||\mathbf{\Theta}||^2$ is a regularization term. β, γ are the hyperparameters to control the weight for treatment prediction and model regularization, respectively.

4.3 Experimental Evaluation

4.3.1 Datasets and Simulation

4.3.1.1 Datasets

The datasets used in the experiments are based on three real-world attributed networks, Flickr, BlogCatalog, and PeerRead. The key statistics of these datasets are shown in Table 4.1, including the number of instances, links, features, and time stamps, as well as the ratio of the treated instances, and the average ATE and its standard deviation over 10 experiments.

TABLE 4.1. Detailed statistics of the datasets.

Dataset	Flickr	BlogCatalog	PeerRead
# of instances	$7,575$	$5,196$	$6,867 \sim 7,601$
# of links	$236,582$ $\sim 240,374$	$170,626$ $\sim 173,524$	$11,819$ $\sim 13,684$
# of features	$12,047$	$8,189$	$1,087$
# of time stamps	25	20	17
ratio (%) of treated	48.72 ± 1.42	46.52 ± 1.58	56.52 ± 3.36
Avg ATE $\pm$ STD	14.35 ± 21.10	20.45 ± 16.63	60.12 ± 83.57

Flickr. Flickr[1] is an image and video based social network, where each node represents a user and each edge stands for the friendship between two users. At each time stamp, we randomly inject/remove $0.1 \sim 1.0\%$ edges, and perturb 0.1% node features (based on the noises sampled from $N(0, 0.01^2)$), yielding a dynamic network across 25 time stamps. The features are a list of tags showing users' interest. We train a 50-topic LDA model [115] on the features and select the top-25 most frequent words from each topic to create hidden confounding. We create a semi-synthetic dataset with the following assumptions: (1) *Treatment.* A user has more viewers from either mobile devices (treated) or desktops (controlled). (2) *Outcome*. A user receives some reviews from the viewers of her posts. (2) *Confounders.* Viewers of a user have their preferences of devices which are influenced by the *topics* of the user's posts. These topics of users causally influence both the devices chosen by their viewers and the reviews they get. (4) *Historical influence*. The topics of a user's posts can be influenced by her previously observed treatment (more viewers from mobile devices or desktops) and the social network, e.g., if a user finds more viewers of her posts are from mobile devices, then she may consider to post more about the topics (e.g., sports) that are preferred by users on mobile devices. To study the ITE of peoples' device preference on the reviews, we simulate the confounders, treatment assignments, and outcome. The detailed simulation process is described in Section 4.3.1.2.

BlogCatalog. BlogCatalog[2] is a social network website where bloggers can share their blogs, where each node represents a blogger and each edge stands for a social relationship between two bloggers. The node features are the bag-of-word representations of the blogger's blogs. As this dataset is also a static data, we follow the same process as Flickr to generate a time-evolving attributed network across 20 different time stamps. We further simulate the treatment assignment, confounders, and outcome in the same way as Flickr.

[1] https://www.flickr.com/

[2] https://www.blogcatalog.com/

PeerRead. PeerRead[3] is a dataset of computer scientific peer reviews for papers, and has been used in previous research of causal inference [116]. This dataset contains a real-world dynamic network of coauthor relations over time. We select 17 time stamps of dynamic network which contains $6867 \sim 7601$ authors. In this dataset, each node refers to an author, and each edge represents their co-author relationship. The features are the bag-of-word representations of their paper titles and abstracts. The confounders are their research areas. The treatment is whether the authors' papers contain buzzy words in their titles and abstracts, which are words in a preset dictionary `{"deep", "neural", "network", "model"}`. The outcome denotes the citation numbers of authors. To study the ITE of buzzy words on the authors' citation, we use the real-world treatment, and simulate the confounders and potential outcomes in the same way as Flickr.

4.3.1.2 Simulation

We incorporate the effect of historical influence (as a p-order autoregressive term [117]) and network information to simulate the confounders $\boldsymbol{z}_i^t$ at time stamp t as:

$$\boldsymbol{z}_i^t = (\frac{1}{\sum_{k=1}^{3} \lambda_k})(\lambda_1 \boldsymbol{\psi}_i^t + \lambda_2 \sum_{u \in N(i)} f(\boldsymbol{x}_u^t) + \lambda_3 f(\boldsymbol{x}_i^t)) + \epsilon^t, \tag{4.8}$$

$$\psi_{i,j}^t = \frac{1}{p} \left(\sum_{r=1}^{p} \alpha_{r,j} z_{i,j}^{t-r} + \sum_{r=1}^{p} \beta_r c_i^{t-r} \right), \tag{4.9}$$

where $\boldsymbol{z}_i^t$ denotes the hidden confounders of instance i at time stamp t. $\boldsymbol{\psi}_i^t$ denotes the historical information. $z_{i,j}^t$ and $\psi_{i,j}^t$ represents the j-th dimension of $\boldsymbol{z}_i^t$ and $\boldsymbol{\psi}_i^t$, respectively. $N(i)$ denotes the neighboring nodes of node i. The function $f(\cdot)$ maps the bag-of-words features of instances to their LDA topics [115]. Here, we train an LDA model with 50 topics with the whole training corpus. The parameters λ_1, λ_2, and λ_3 control the impact of historical information, current network structure, and current features on the confounders. In the experiments, we set $\lambda_1 = 0.3, \lambda_2 = 0.3, \lambda_3 = 0.3$ by default. $\epsilon^t \sim \mathcal{N}(0, 0.001^2)$ is the random noise, $\alpha_{r,j} \sim \mathcal{N}(1 - (r/p), (1/p)^2)$ controls the impact of historical information from the previous p time stamps, where p is set to 3 by default, $\beta_r \sim \mathcal{N}(0, 0.02^2)$ controls the impact of previous treatment assignment.

To synthesize the observed treatment assignment, we select two points $\boldsymbol{r}_0$ and $\boldsymbol{r}_1$ in the LDA topic space as the centroids for the treated and controlled groups. We simulate the treatment as follows:

$$c_i^t \sim \text{Bernoulli}(\frac{exp(p_{i,1}^t)}{exp(p_{i,0}^t) + exp(p_{i,1}^t)}), \tag{4.10}$$

[3] https://github.com/allenai/PeerRead

$$p_{i,0}^t = \boldsymbol{r}_0^{'} \boldsymbol{z}_i^t,\ p_{i,1}^t = \boldsymbol{r}_1^{'} \boldsymbol{z}_i^t. \tag{4.11}$$

Then we synthesize the potential outcomes as below by setting $c = 1$ or $c = 0$:

$$y_{c,i}^t = S(p_{i,0}^t + c \cdot p_{i,1}^t) + \eta^t \tag{4.12}$$

where S is a scaling factor, and is specified as $S = 20$. $\eta^t \sim \mathcal{N}(0, 0.5^2)$ is a random noise term.

4.3.2 Evaluation Metrics

We adopt two widely used evaluation metrics – Rooted Precision in Estimation of Heterogeneous Effect (PEHE) [102] and Mean Absolute Error (ATE) [118] to measure the quality of the estimated individual treatment effects at different time stamps:

$$\sqrt{\epsilon_{PEHE}^t} = \sqrt{\frac{1}{n^t} \sum_{i \in [n^t]} (\hat{\tau_i^t} - \tau_i^t)^2}. \tag{4.13}$$

$$\epsilon_{ATE}^t = |\frac{1}{n^t} \sum_{i \in [n^t]} \hat{\tau_i}^t - \frac{1}{n^t} \sum_{i \in [n^t]} \tau_i^t|. \tag{4.14}$$

In our experiments, we take the average performance over all time stamps for evaluation. We denote them by $\sqrt{\epsilon_{PEHE}}$ and ϵ_{ATE}, respectively.

4.3.3 Experiment Settings

Baselines. To investigate the effectiveness of our framework in learning ITEs from time-evolving networked observational data, we compare our framework with multiple state-of-the-art methods:

- **Causal Forest (CF)** [103]. Based on the strong ignorability assumption [109], CF learns ITE as an extension of Breiman's random forest [119]. We set the number of trees as 100.
- **Bayesian Additive Regression Trees (BART)** [102]. BART is a Bayesian regression tree based ensemble model that is widely used in learning ITE. It is also based on the strong ignorability assumption.
- **Counterfactual Regression (CFR)** [37]. CFR also learns representation for the confounders based on the strong ignorability assumption. Balancing techniques including Wasserstein-1 distance and maximum mean discrepancy are adopted and we refer these two variants as CFR-Wass and CFR-MMD.

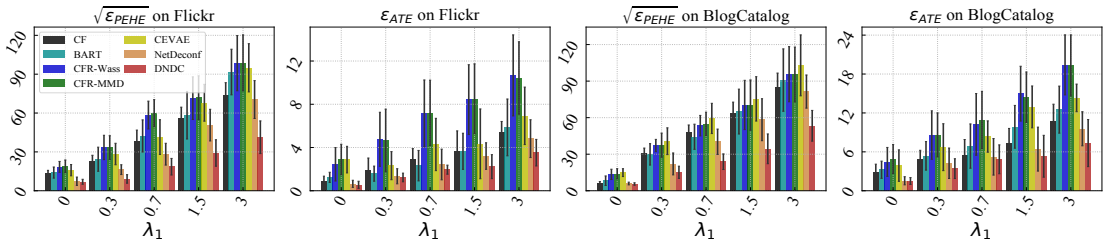


FIGURE 4.3. Performance comparison between DNDC and baselines under different settings of historical information influence.

- **Causal Effect Variational Autoencoder (CEVAE)** [38]. CEVAE is a deep latent-variable model for learning ITE, which learns representations of confounders as Gaussian distributions through propagating information from original features, observed treatments, and factual outcomes.
- **Network Deconfounder (NetDeconf)** [120]. NetDconf relaxes the strong ignorability assumption by assuming that the hidden confounders of observational data can be controlled with auxiliary relational information among data.

Setup. All the aforementioned baselines are designed for static data and thus we run these algorithms at each time stamp independently. On the other hand, only our proposed DNDC can well capture the temporal dependency for ITE estimation. The data instances (nodes) are randomly split into 60-20-20% of training/validation/test data. We evaluate the average $\sqrt{\epsilon_{PEHE}}$ and ϵ_{ATE} over all the time stamps. All the results are reported with mean and standard deviation over 10-time repeated executions. Unless otherwise specified, we set our learning rate as $5e-4$, $d_h = 64$, $d_z = 64$, $\beta = 1.0$, $\gamma = 0.01$, and we use Adam as our optimizer. For all the baselines based on confounder representation learning such as CEVAE and NetDeconf, we also set the dimension of the learnt representation as d_z, same as our proposed method. As described in Section 4.3.1.2, in the experiments, we use three hyperparameters λ_1, λ_2, and λ_3 to control the influence of the historical information, current network structure, and current feature information on the current confounders, respectively.

4.3.4 ITE Estimation Performance under Varying Influence from Historical Information

To investigate the performance of DNDC under different levels of influence from historical information on confounders, an experiment is designed with varying λ_1 together with fixed λ_2 and λ_3. Fig. 4.3 shows the comparison of the ITE estimation performance between DNDC and other baselines. Generally speaking, we observe that DNDC consistently outperforms all the baselines with lower $\sqrt{\epsilon_{PEHE}}$ and ϵ_{ATE}. When $\lambda_1 = 0$, the historical information has no

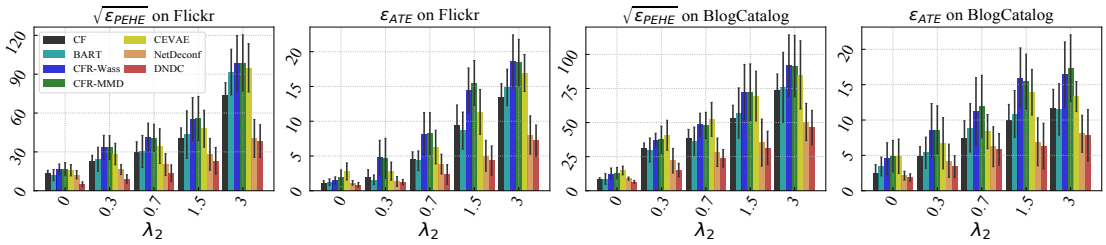


FIGURE 4.4. Performance comparison between DNDC and baselines under different settings of network structure influence.

impact on the current confounders. In this case, DNDC and NetDeconf [41] achieve the best performance due to their capability of utilizing the network structure. When λ_1 increases, the current ITE estimation relies more on historical information, while other baselines without consideration of historical information fail in this scenario. But DNDC is stably better as it can leverage historical data.

4.3.5 ITE Estimation Performance under Varying Influence from Network Structure

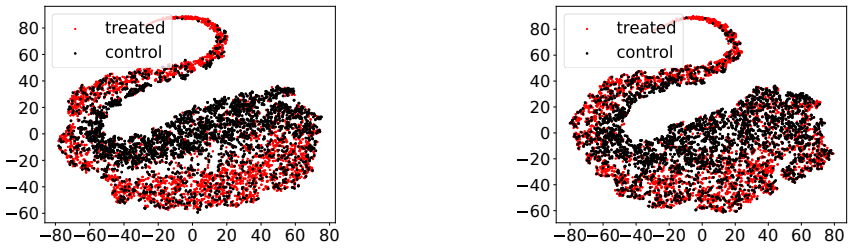
To evaluate DNDC in leveraging the relational information in graphs, an experiment with different values of λ_2 but fixed values of λ_1 and λ_3 is conducted. As shown in Fig. 4.4, when $\lambda_2 = 0$, the hidden confounders are independent of the graph structure, in this case, NetDeconf loses its superiority over other baselines. But DNDC can still achieve better ITE estimation by capturing the historical influence on the hidden confounders at the current time period. When λ_2 increases, the confounder representation learning component in DNDC captures the confounders buried in the graph structure, and achieves better ITE estimation performance.

4.3.6 The Impact of Representation Balancing

To evaluate the impact of the proposed adversarial learning based representation balancing method, we compare the performance of our balancing method with other two commonly used representation balancing methods: Wasserstein-1 (Wass) distance [121] and maximum mean discrepancy (MMD) [37]. Table 4.2 shows the results of ITE estimation performance with these different representation balancing techniques and our method consistently outperforms other baselines. Fig. 4.5 shows a specific example of the representation distributions with/without the gradient reverse layer. We observe that with the gradient reverse layer, the representation distributions of treated and control group become closer.

TABLE 4.2. Performance comparison with different representation balancing methods.

Dataset		Wass	MMD	Gradient Reverse
Flickr	$\sqrt{\epsilon_{PEHE}}$	9.125 ± 1.566	9.531 ± 1.573	$\mathbf{8.131 \pm 1.342}$
	ϵ_{ATE}	1.839 ± 0.368	1.952 ± 0.433	$\mathbf{1.413 \pm 0.351}$
BlogCatalog	$\sqrt{\epsilon_{PEHE}}$	16.115 ± 2.857	17.035 ± 4.243	$\mathbf{15.132 \pm 2.542}$
	ϵ_{ATE}	4.815 ± 1.367	5.250 ± 1.345	$\mathbf{3.414 \pm 1.272}$
PeerRead	$\sqrt{\epsilon_{PEHE}}$	49.062 ± 4.452	49.643 ± 4.834	$\mathbf{47.716 \pm 4.014}$
	ϵ_{ATE}	5.482 ± 1.347	5.648 ± 1.617	$\mathbf{4.451 \pm 1.379}$

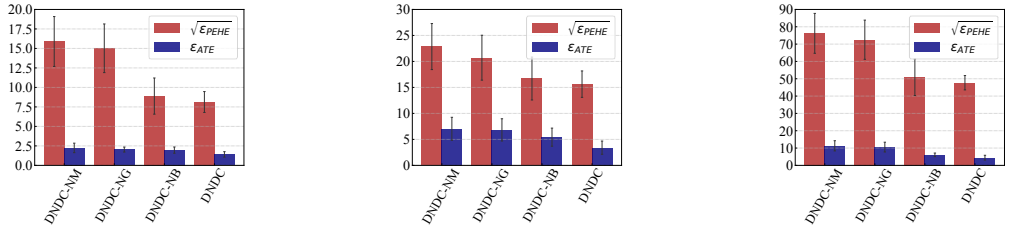


(A) No balancing (B) With balancing

FIGURE 4.5. Representation distributions with or without gradient reverse layer.

4.3.7 Ablation Study

To further investigate the impact of different components of DNDC, we conduct ablation study by comparing DNDC against the following variants: (1) *No GRU*: This variant omits the GRU and attention layers, which means that no historical information is utilized in learning the confounders. As this variant does not benefit from the memory of previous time stamps, we denote it by DNDC-*NM*. (2) *No GCNs*: In this variant, we replace the GCN layers with a simple MLP. We denote this variant by DNDC-*NG*. (3) *No balancing*: This variant does not use any representation balancing techniques and is denoted by DNDC-*NB*. Fig. 4.6 reports the ITE estimation performance of different variants of our proposed framework. We can see that DNDC-*NM* and DNDC-*NG* cannot render satisfactory performance as they cannot leverage historical information or network structure for learning representations of hidden confounders. The performance of DNDC-*NB* is degraded by the imbalance of distributions between the treated and the control group, while DNDC performs better with balancing method because the distribution balancing helps mitigate the confounding bias. In short, all three components contribute to the superior performance of DNDC.



(A) Flickr (B) BlogCatalog (C) PeerRead

FIGURE 4.6. Ablation study for different variants of DNDC.

4.4 Real-world Application

4.4.1 Motivation: Assessing the Impact of Covid-19 Related Policies

The coronavirus disease 2019 (COVID-19) has seriously affected different aspects of human life [122, 123, 124, 125, 126, 127, 128]. To mitigate its spread, decision-makers and public authorities have issued various policies [129, 130, 131, 132, 133, 134, 135]. Correspondingly, a natural question to ask is: *which policy is more effective to mitigate the spread of COVID-19 in a given context*? Various studies such as correlation analysis [136, 137] address this question from a statistical perspective. Such studies can only capture the statistical (but may not be causal) dependencies between the policies and the spread of COVID-19. Yet answering this question from a *causal* perspective is essential, as it can provide guidance to policymakers for addressing other pandemics or even further waves of the current one [138, 139, 140]. However, the gold standard of causal effect estimation, i.e., a randomized controlled trial (RCT) [141] is not applicable under pandemic circumstances due to ethical and practical issues [142]. Hence, the causal impact of different policies on the COVID-19 outbreak dynamics (e.g., the numbers of confirmed cases) is expected to be directly assessed with the observational data.

We assess the causal impact of different COVID-19 policies on the outbreak dynamics with observational data. Specifically, we study this problem: given a specific time period, what is the causal effect of COVID-19 policies (*treatments*) on the outbreak dynamics (*outcomes*) in each county (*unit/instance*)? In this problem, the vigilance of residents can be a hidden confounder, and if handled improperly, we may incorrectly take the statistical correlations between the presence of these policies and the outbreak dynamics as causal relations.

To remedy these issues, we adopt a weaker form of the unconfoundedness assumption [143], which enables us to capture the unobserved confounders from the proxy variables for them, i.e., the variables which have dependencies with the unobserved confounders. For example, certain confounders such as residents' vigilance in a county can be inferred from the popularity of web

searches about COVID-19 related keywords on Google. Intuitively, the more vigilant people are, the more frequently they will search COVID-19 terms. Besides, residents' vigilance can also be inferred from the relational information among different counties, such as a county-to-county distance network or mobility flow network. One potential reason is that neighboring counties tend to have more interactions and similar cultures, thus, their residents will have similar levels of vigilance. Historical information, such as the adopted policies and the spread of COVID-19 at earlier time periods may also influence the current confounders such as residents' vigilance. With such proxy variables, the confounders can be captured, and thus an unbiased causal effect estimation becomes possible.

4.4.2 Dataset Collections and Related Policies

We integrate data from several different data sources, including 391 counties in the United States. We consider our selected counties representative as they cover different states with different political ideologies. For treatments, as shown in Table 4.3, we collect COVID-19 related policies that have been enacted by different counties in the United States throughout 2020; for outcomes, we use the numbers of confirmed cases and death cases of different counties throughout 2020. To control the influence of unobserved confounders, we also collect data regarding the covariates of different counties and their relations. In particular, two types of networks (distance network and mobility network) among counties are used in our study as proxies for hidden confounders. More details can be found in Appendix A3.

4.4.3 Formulating Policy Assessment as a Causal Effect Estimation Problem

We consider the COVID-19 related policy types in n counties across T time periods. Different counties are described by the same set of covariates (i.e., features) over time, and we denote them by $\mathbf{X}^t = \{\mathbf{x}_i^t\}_{i=1}^n$, where $\mathbf{x}_i^t$ represents the covariates of the i-th county at time period t (e.g., in Albemarle county, VA, residents' search popularity of COVID-19 related keywords on Google throughout March, 2020). We represent the adjacency matrix of the network (e.g., the distance network) at time period t as $\mathbf{A}^t \in \mathbb{R}^{n\times n}$, where $\mathbf{A}_{ij}^t$ is the weight of edge $i \to j$ in $\mathbf{A}^t$, and $\mathbf{A}_{ij}^t = 0$ if there does not exist such edge. We assume that the edge weight can reflect the similarity between counties. For each policy type, we use $\mathbf{P}^t = \{p_i^t\}_{i=1}^n$ to denote whether policies of this type are in effect in these n counties at time period t, where p_i^t is either 1 (treated) or 0 (not treated, i.e., controlled), corresponding to whether the policy type is in effect in the i-th county or not. At each time period, the treated counties form the *treatment group*, while the controlled counties form the *control group*. Here, we denote

TABLE 4.3. Examples of detailed policies about selected policy types in each category (including the states that enacted them). The three parts correspond to the categories of social distancing, reopening, and mask requirement, respectively.

Top policy types	Example policies
State of emergency	Limit in-person gatherings (VA)
Nursing homes	Limit visits to hospitals (OH)
Food and drink	Outdoor dining, and delivery only (RI)
Childcare (K-12)	Close public schools (VI)
Gatherings	Gatherings limited to 10 people (OH)
Phase 2	Reopen lodging establishments (NM)
Entertainment	Some businesses may reopen (ND)
Outdoor and recreation	Contact sport practices to reopen (OH)
Personal care	Health care may reopen (ND)
Food and drink	Reopen on-site dining (TN)
Any mask requirement	Face covering required in public (IN)
Public facing businesses	Facing businesses require masks (WI)
Food and drink	Restaurants require face masks (MP)
Phase 3	Outdoor venues require masks (ID)
Phase 2	Mandate masks in K-12 schools (UT)

a specific manifestation of outbreak dynamics (such as the number of confirmed cases) at time period t as $\mathbf{Y}^t = \{y_i^t\}_{i=1}^n$, which are also referred to as *observed outcomes*. The history of covariates before time period t is denoted by $\bar{\mathbf{X}}^t = (\mathbf{X}^1, \mathbf{X}^2, ..., \mathbf{X}^{t-1})$, and the history of treatments $\bar{\mathbf{P}}^t$ and network structures $\bar{\mathbf{A}}^t$ are defined similarly. Then, we denote all the historical observational data before time period t as $\bar{\mathbf{H}}^t = \{\bar{\mathbf{X}}^t, \bar{\mathbf{A}}^t, \bar{\mathbf{P}}^t, \bar{\mathbf{Y}}^t\}$.

We frame the causal assessment of COVID-19 related policies as a causal effect estimation problem from time-varying observational data. Our goal is to investigate to what extent a *cause* (i.e., *treatment*, e.g., a policy in effect) would causally affect an *outcome* (e.g., the number of confirmed cases) for each instance (e.g., a county) at different time periods. To estimate the causal effect of a policy on the outbreak dynamics at time period t, we should compare the potential outcomes of the outbreak dynamics in each county if this policy had/had not been in effect during time period t. Generally, the *potential outcome* [108, 20] means the outcome that would be realized if the instance got treated/controlled, e.g., "In March, what would the number of confirmed cases be in Albemarle county, VA, if the mask requirement policy had/had not been in effect during that time period?". We denote the potential outcomes of all counties if the policy had/had not been in effect at time period t by $\mathbf{Y}_1^t = \{y_{1,i}^t\}_{i=1}^n$ and $\mathbf{Y}_0^t = \{y_{0,i}^t\}_{i=1}^n$. In our setting, the potential outcome $y_{p,i}^t$ is the outcome that would be

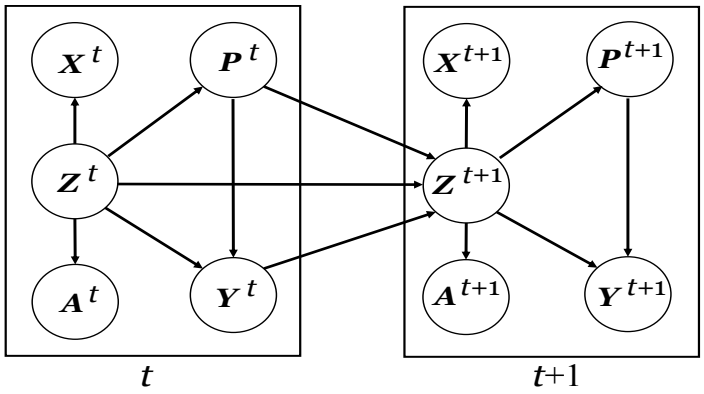


FIGURE 4.7. Causal graph of the COVID-19 problem.

realized if the i-th instance is under treatment p at time period t. Then the *individual treatment effect (ITE)* [20] for each instance at time period t is defined as the difference between the potential outcome if the instance gets treated and the potential outcome if it gets controlled at that time period. In our problem, the ITE of each policy on the outbreak dynamics in each county at time period t is the difference between two potential outcomes:

$$\tau_i^t = \mathbb{E}[y_{1,i}^t - y_{0,i}^t | \mathbf{x}_i^t, \mathbf{A}^t, \bar{\mathbf{H}}^t]. \tag{4.15}$$

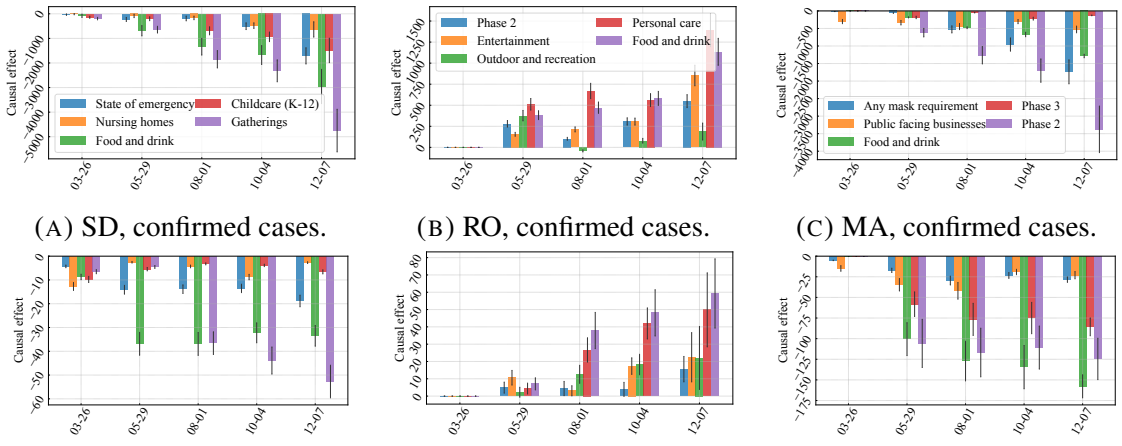
Then the average treatment effect (ATE) at time period t is computed as the average of ITEs over all counties:

$$\tau_{ATE}^t = \mathbb{E}_{i \in [n]}[\tau_i^t]. \tag{4.16}$$

Inspired by the previous work on causal effect estimation from time-evolving data [21], we design a causal graph for our studied problem as shown in Fig. 4.7, where each arrow means a causal relationship. We denote the time-varying (unobserved) confounders (e.g., residents' vigilance) by $\mathbf{Z}^t = \{\mathbf{z}_i^t\}_{i=1}^n$.

4.4.4 Causal Assessment of COVID-19 Policies

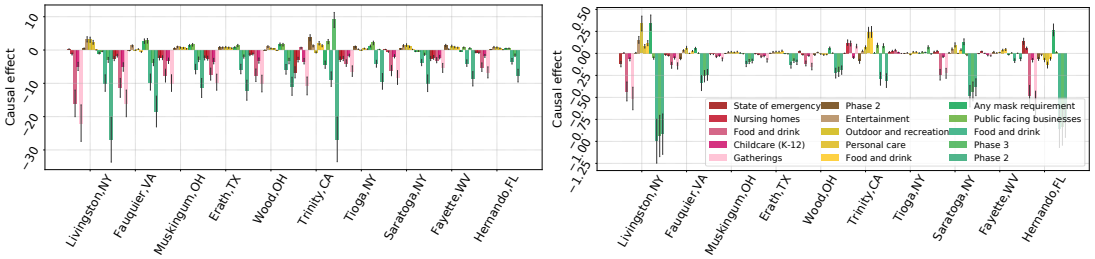
We estimate the causal effects of different policies on the outbreak dynamics at different time periods in 2020. Based on our estimation of their average treatment effects (ATEs), Table 4.3 shows the information about the top-5 most impactful policy types in each policy category with a certain goal. Some policy types may contain policies in different categories. For example, in Table 4.3, for the category Reopening, policy type "Phase 2" covers the reopening-related policies like "Reopen lodging establishments". For category Mask, "Phase 2" covers mask-related policies like "Mandate masks in schools". Fig. 4.8 summarizes our estimation of the ATEs of these policy types. Each column in Fig. 4.8 corresponds to a category: *social distancing*, *reopening*, and *mask requirement*. The first and second rows



(A) SD, confirmed cases. (B) RO, confirmed cases. (C) MA, confirmed cases.

(D) SD, death cases. (E) RO, death cases. (F) MA, death cases.

FIGURE 4.8. Causal effect estimation of different policy types at different time periods over year 2020. The three columns correspond to the policy categories of social distancing (SD), reopening (RO), and mask requirements (MA). The two rows correspond to the estimated causal effects on the number of confirmed cases and the number of death cases, respectively.



(A) Confirmed cases. (B) Death cases.

FIGURE 4.9. Causal effect estimation of different policy types on the outbreak dynamics in different counties. The red, yellow and green bars correspond to the policy categories of social distancing, reopening, and mask requirement, respectively.

show the estimated ATEs of these policy types on the number of confirmed cases and death cases, respectively. We have the following observations from Fig. 4.8:

At the macro-level, the policy types regarding social distancing and mask requirement have negative causal effects on both the number of confirmed cases and death cases, while the policy types about reopening have positive causal effects. These negative values of causal effect indicate that the corresponding policy types causally help reduce the spread of COVID-19, while the policy types with positive causal effects may have a contrary effect because they increase the risk of infection. These observations appear intuitively plausible and are also consistent with existing literature regarding COVID-19 related policies [130, 144].

At the micro-level, we zoom into the most impactful policy types in each category. In the category of social distancing, the policy types "Gatherings" and "Food and drink" seem to have the strongest effects. From the detailed description of these policies, they powerfully prohibit the number of individuals in multiple activities, especially in high-risk places such as restaurants. In the category of reopening, the estimated effects of policy types "Personal care" and "Food and Drink" indicate that reopening public places such as personal care center and restaurants heavily increase the risk of COVID-19 infection. In the category of mask requirement, "Phrase 2" and "Any mask requirement" are most impactful as they mandate face masks usage in many public spaces.

Generally, above observations are consistent for different outcomes including the number of confirmed/death cases, as well as for different time periods. Besides, we observe that the policies can have stronger effects when the the pandemic becomes more severe, such as during the outbreak at the end of 2020. In conclusion, above observations reveal the importance of in-time policies to limit close contact (within about 6 feet) among people in different aspects, e.g., distance, frequency and certain body parts (e.g., face) during the spread of respiratory pandemics like COVID-19.

Furthermore, we zoom into the county-level, and assess the ITEs of different policy types on the outbreak dynamics in each county. In Fig 4.9, we randomly select 10 counties as examples and show the estimated ITEs of different policy types in each of them. To compare different counties, each result is calculated from the original ITEs (as defined in Eq. 4.15) averaged over all the time periods, and then is normalized by the number of confirmed cases in the corresponding county at the last time period. From Fig. 4.9, we observe that the ITEs of the three categories of policies in each county generally have similar patterns as their ATEs over all counties, i.e., the "social distancing" and "mask requirement" policies are beneficial for controlling the spread of COVID-19, while the "reopening" policies have increased the risk. Besides, the policies have a stronger impact in high-risk locations such as California, New York, and Florida.

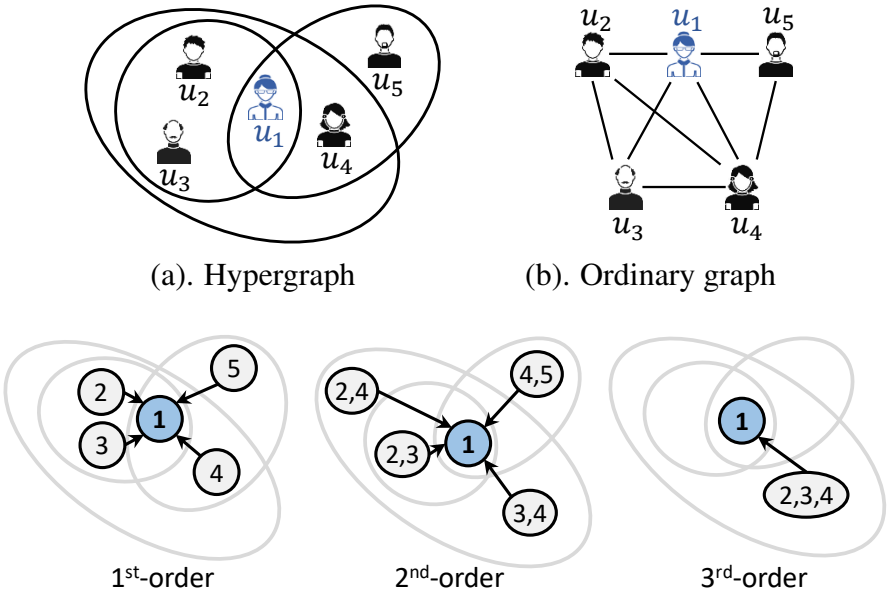
CHAPTER 5

Causal Effect Estimation under Interference on Hypergraphs

Classic causal effect estimation is based on the Stable Unit Treatment Value (SUTVA) assumption that there is no interference (i.e., spillover effect) among different units, requiring that the treatment of one unit does not impact the outcome of another unit. However, this assumption can be unrealistic in real-world scenarios, especially in interconnected systems like graphs. For instance, an individual's risk of COVID-19 infection can be affected by the face covering practices of others in their contact network. Failure to account for these interdependencies can lead to flawed estimations of causal effects.

Recently, there have been many efforts aiming to tackle the problem of causal effect estimation under interference. Most existing studies addressing this problem [44, 45, 46, 145, 47, 146, 147, 148] assume the interference only occurs between pairs of units on ordinary graphs (as shown in Fig. 5.1(b)). While the conventional pairwise interactions in graphs are widely-used and applicable to a variety of settings, such as person-to-person physical contact or social networks, they fall short in capturing the intricacies of group interactions, where each interaction can involve more than just two individuals [149, 150, 151]. Hypergraphs can be introduced to address this limitation. Unlike ordinary edges which connect only two nodes, a hyperedge can connect an arbitrary number of nodes (as shown in Fig. 5.1(a)), reflecting the nature of group interactions. Consider a hypergraph example that individuals are connected via in-person social events, each mass gathering event can be represented as a hyperedge. In a hypergraph, high-order interference may exist. For instance, in a gathering event represented by a hyperedge, an individual's risk of COVID-19 infection can be influenced not only by direct first-order interference from others within the event, but also by indirect high-order interference resulting from the interactions among attendees, as shown in Fig. 5.1(c). It is important to handle the high-order interference that exists on hypergraphs.

To address this challenge, we propose a framework HyperSCI [22] for treatment effect estimation under high-order interference in hypergraphs. At its core, this framework controls for confounders and models high-order interference through representation learning. HyperSCI leverages a hypergraph neural network to effectively capture the interference patterns by learning interference representations and using an attention mechanism to model the



(a). Hypergraph (b). Ordinary graph

(c). First, second, and third-order interferences with u_1.

FIGURE 5.1. Hypergraph, ordinary graph, and interferences. (a) An example of a hypergraph; (b) An ordinary graph projected from this hypergraph; (c) Interferences with node u_1 from its neighbors on the hypergraph.

relative importance of each unit within each hyperedge. These hypergraph neural network technologies equip HyperSCI with both high accuracy and computational efficiency.

5.1 Problem Definition

DEFINITION 7. *(Hypergraph) A **hypergraph** $\mathcal{H} = \{\mathcal{V}, \mathcal{E}\}$ consists of a set of n nodes $\mathcal{V} = \{v_i\}_{i=1}^{n}$ and a set of m hyperedges $\mathcal{E} = \{\mathbf{e}_k\}_{k=1}^{m}$. Each hyperedge can connect any number of nodes.*

In the studied problem, the given observational data is denoted by $\{\mathbf{X}, \mathcal{H}, \mathbf{T}, \mathbf{Y}\}$. Here, $\mathbf{X} = \{\mathbf{x}_i\}_{i=1}^{n}$, $\mathbf{T} = \{t_i\}_{i=1}^{n}$ and $\mathbf{Y} = \{y_i\}_{i=1}^{n}$ represent node features, treatment assignments, and observed outcomes, respectively. $\mathbf{H} = \{h_{i,e}\} \in \mathbb{R}^{n \times m}$ is an incidence matrix for hypergraph $\mathcal{H}$. Here, $h_{i,e} = 1$ if node i is in hyperedge e, otherwise $h_{i,e} = 0$. The treatment assignment for each node is binary (i.e., $t_i \in \{0, 1\}$).

DEFINITION 8. *(Potential outcome) The **potential outcome** [152] of the unit i (denoted by y_i^1 or y_i^0) is defined as the outcome which would be realized for unit i under treatment $t_i = 1$ or $t_i = 0$. These potential outcomes can be obtained via a transformation function $Y_i^{T_i} = \Phi_Y(T_i, X_i, T_{-i}, X_{-i}, H)$. Here, Φ_Y is a (non-deterministic) function, i.e., $y_i^{t_i} = \Phi_Y(t_i, \mathbf{x}_i, \mathbf{T}_{-i}, \mathbf{X}_{-i}, \mathbf{H})$, where $(\cdot)_{-i}$ denotes all other nodes on $\mathcal{H}$ except i.*

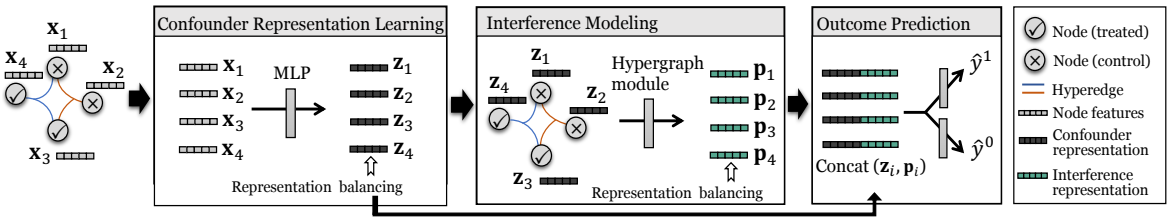


FIGURE 5.2. An illustration of HyperSCI, including three components: confounder representation learning, interference modeling, and outcome prediction.

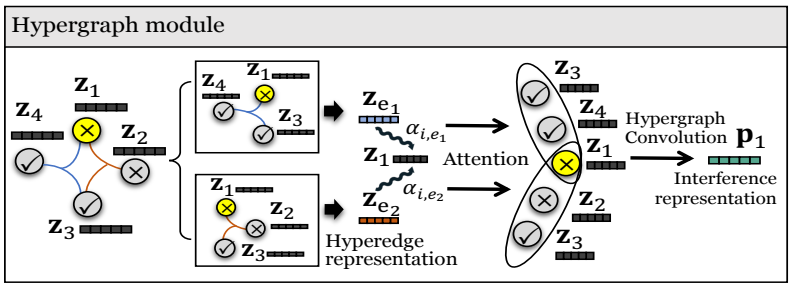


FIGURE 5.3. An illustration of the hypergraph module in HyperSCI. Here node v_1 (highlighted in yellow) is taken as an example.

This work aims to estimate ITE in a hypergraph. Based on the above definition, the ITE in the studied problem is defined as follows:

DEFINITION 9. *For each node i on the hypergraph $\mathcal{H}$, the **individual treatment effect** (ITE) is defined by the difference between potential outcomes corresponding to $t_i = 1$ and $t_i = 0$:*

$$\begin{aligned}\tau(\mathbf{x}_i, \mathbf{T}_{-i}, \mathbf{X}_{-i}, \mathbf{H}) &= \mathbb{E}[Y_i^1 - Y_i^0 | X_i = \mathbf{x}_i, T_{-i} = \mathbf{T}_{-i}, X_{-i} = \mathbf{X}_{-i}, H = \mathbf{H}] \\ &= \mathbb{E}[\Phi_Y(1, \mathbf{x}_i, \mathbf{T}_{-i}, \mathbf{X}_{-i}, \mathbf{H}) - \Phi_Y(0, \mathbf{x}_i, \mathbf{T}_{-i}, \mathbf{X}_{-i}, \mathbf{H})].\end{aligned} \tag{5.1}$$

5.2 Proposed Method

We propose the framework HyperSCI [22] to address the studied problem. As shown in Fig. 5.2, this framework mainly contains three components: confounder representation learning, interference modeling, and outcome prediction.

5.2.1 Confounder Representation Learning

HyperSCI learns representations of confounders by mapping the node features $\mathbf{x}_i$ into a latent space with a multilayer perceptron (MLP) module, i.e., $\mathbf{z}_i = \text{MLP}(\mathbf{x}_i)$. The confounder

representations for all the nodes are denoted by $\mathbf{Z} = \{\mathbf{z}_i\}_{i=1}^{n}$. Similar as [37], a Wasserstein-1 distance [153] based representation balancing method is used to minimize the distance between the representation distributions of the treatment group and control group.

5.2.2 Interference Modeling

An interference modeling module is developed to model the high-order interference among nodes in the hypergraph. More specifically, a function $\Psi(\cdot)$ is learned via a hypergraph neural network module to obtain the interference representations ($\mathbf{p}_i$) for each node i, i.e., $\mathbf{p}_i = \Psi(\mathbf{Z}, \mathbf{H}, \mathbf{T}_{-i}, t_i)$. The illustration of this module is shown in Fig. 5.3. This module is implemented based on a hypergraph convolutional network [149, 151] as well as a hypergraph attention mechanism [149, 154, 155].

To learn the interference representations for each node, the treatment and confounder representations are propagated through the hypergraph structure. A vanilla Laplacian matrix for the given hypergraph $\mathcal{H}$ can be calculated as:

$$\mathbf{L} = \mathbf{D}^{-1/2}\mathbf{H}\mathbf{B}^{-1}\mathbf{H}^{\top}\mathbf{D}^{-1/2}, \tag{5.2}$$

where $\mathbf{D} \in \mathbb{R}^{n\times n}$ is a diagonal matrix in which each element stands for the node degree (i.e., $\sum_{e=1}^{m} h_{i,e}$). $\mathbf{B} \in \mathbb{R}^{m\times m}$ is a diagonal matrix in which each element corresponds to the size of each hyperedge ($\sum_{i=1}^{n} h_{i,e}$). The hypergraph convolution operation is defined as:

$$\mathbf{P}^{(l+1)} = \text{LeakyReLU}\left(\mathbf{L}\mathbf{P}^{(l)}\mathbf{W}^{(l+1)}\right), \tag{5.3}$$

where $\mathbf{P}^{(l)}$ denotes the representations in the l-th layer of the hypergraph module. The input of the first layer is the confounder representation masked by the treatment assignment, i.e., $\mathbf{p}_i^{(0)} = t_i * \mathbf{z}_i$. Here, $*$ is element-wise multiplication. $\mathbf{W}^{(l+1)} \in \mathbb{R}^{d^{(l)}\times d^{(l+1)}}$ represents the parameter matrix in the (l+1)-th layer of the hypergraph module, where $d^{(l)}$ and $d^{(l+1)}$ are the dimensionality of the l-th layer and (l+1)-th layer, respectively.

While the hypergraph convolution layer allows for interference modeling through hyperedges, it lacks flexibility to consider the varying significance of interference on different nodes via different hyperedges. To address this, a hypergraph attention mechanism [149, 155, 154] is utilized to capture the intrinsic relationship between nodes and hyperedges. Specifically, the attention weights are learned for each node and its corresponding hyperedges, which allows for a better understanding of how certain individuals, such as those participating in group events, may have a greater influence on or be influenced by others in these groups within the context of a hypergraph, as seen in the COVID-19 example. More specifically, the attention

score between a node i and a hyperedge e is calculated as:

$$\alpha_{i,e} = \frac{\exp(\sigma(\text{sim}(\mathbf{z}_i\mathbf{W}_a, \mathbf{z}_e\mathbf{W}_a)))}{\sum_{k\in\mathcal{E}_i}\exp(\sigma(\text{sim}(\mathbf{z}_i\mathbf{W}_a, \mathbf{z}_k\mathbf{W}_a)))}, \tag{5.4}$$

where $\sigma(\cdot)$ is an activation function, $\mathcal{E}_i$ is the set of hyperedges which contain the node i. $\mathbf{z}_e$ is the representation for each hyperedge e, obtained by aggregating across the representations of its associated nodes. $\mathbf{W}_a$ denotes a parameter matrix to compute the node-hyperedge attention. $\text{sim}(\cdot)$ denotes a similarity function, which can be implemented as follows:

$$\text{sim}(\mathbf{x}_i, \mathbf{x}_j) = \mathbf{a}^\top[\mathbf{x}_i, \mathbf{x}_j]. \tag{5.5}$$

Here, $\mathbf{a}$ is a weight vector, $[\cdot,\cdot]$ is a concatenation operation. The attention scores are used to model the different significance of interference. More specifically, the original incidence matrix $\mathbf{H}$ of the hypergraph in Eq. 5.2 is replaced with an attention-involved matrix $\tilde{\mathbf{H}} = \{\tilde{h}_{i,e}\}$, where $\tilde{h}_{i,e} = \alpha_{i,e}h_{i,e}$.

5.2.3 Outcome Prediction

Based on the confounder representations and the interference representations, the potential outcomes are predicted by:

$$\hat{y}_i^1 = f_1([\mathbf{z}_i, \mathbf{p}_i]),\ \hat{y}_i^0 = f_0([\mathbf{z}_i, \mathbf{p}_i]), \tag{5.6}$$

where $f_1(\cdot)$ and $f_0(\cdot)$ are learnable functions which are trained to predict potential outcomes for treatment assignment 1 and 0, respectively. The ITE for each node i is then estimated by: $\hat{\tau}_i = \hat{y}_i^1 - \hat{y}_i^0$. The prediction for the observed outcome is obtained by $\hat{y}_i = \hat{y}_i^{t_i}$. The final loss function for HyperSCI is:

$$\mathcal{L} = \sum_{i=1}^{n}(y_i - \hat{y}_i)^2 + \alpha\mathcal{L}_b + \lambda\|\mathbf{\Theta}\|^2, \tag{5.7}$$

where the first term is the outcome prediction loss, which can be implemented by standard mean squared error. $\mathcal{L}_b$ is the representation balancing loss, as introduced in Section 4.2.3. $\mathbf{\Theta}$ denotes all the model parameters. α and λ are hyperparameters that control the weights for representation balancing and model regularization, respectively.

5.3 Experimental Evaluation

5.3.1 Datasets and Simulation

5.3.1.1 Simulation

We obtain the semi-synthetic data based on two publicly available hypergraph datasets (**Contact** [156, 157], **Goodreads** [158, 159]) and one large-scale proprietary web application dataset (**Microsoft Teams**). We do not account for the temporal information of each hyperedge in our experiments and leave this as a future research direction instead. In all three datasets, we discard extremely large hyperedges and keep those with no more than 50 nodes only. [1]

Outcome Simulation. Given the treatment allocations $\mathbf{T}$, node features $\mathbf{X}$, and the hypergraph structure $\mathbf{H}$, the potential outcome of an individual i can be simulated via

$$y_i = f_{y,0}(\mathbf{x}_i) + \overbrace{\gamma f_t(t_i, \mathbf{x}_i)}^{\text{individual treatment effect (ITE)}} + \underbrace{\beta f_s(\mathbf{T}, \mathbf{X}, \mathbf{H})}_{\text{hypergraph spillover effect}} + \epsilon_{y_i}, \tag{5.8}$$

where $f_{y,0}(\mathbf{x}_i)$ describes the outcome of instance i when $t_i = 0$ and without network interference, $f_t(\cdot)$ calculates the ITE of each instance, $f_s(\cdot)$ calculates the spillover effect, and ϵ_{y_i} denotes the random noise from a Gaussian distribution $\mathcal{N}(0, 1)$. We specify $f_{y,0}(\mathbf{x}_i)$ as a linear transformation of $\mathbf{x}_i$:

$$f_{y,0} = \mathbf{w}_0 \mathbf{x}_i, \tag{5.9}$$

where $\mathbf{w}_0 \sim \mathcal{N}(0, \mathbf{I}), \mathbf{w}_0 \in \mathbb{R}^d$. Then we control the individual treatment effect ($f_t(t_i, \mathbf{x}_i)$) and the hypergraph spillover effect ($f_s(\mathbf{T}, \mathbf{X}, \mathbf{H})$) under two different settings:

(1) **Linear.**

$$f_t(t_i, \mathbf{x}_i) = \begin{cases} \mathbf{w}_1 \mathbf{x}_i + \epsilon & \text{if } t_i = 1 \\ 0 & \text{if } t_i = 0 \end{cases} \tag{5.10}$$

Here $\mathbf{w}_1 \in \mathbb{R}^d$, and each element in $\mathbf{w}_1$ follows a Gaussian distribution. We generate f_s as:

$$f_s(\mathbf{T}, \mathbf{X}, \mathbf{H}) = \frac{1}{|\mathcal{E}_i|} \sum_{e \in \mathcal{E}_i} \sigma'(\frac{1}{|\mathcal{N}_e|} \sum_{j \in \mathcal{N}_e} t_j \times f_t(t_j, \mathbf{x}_j)). \tag{5.11}$$

Here, $\sigma'(\cdot)$ is a function on the aggregation over each hyperedge e that contains node i. $\mathcal{N}_e$ is the set of nodes in hyperedge e. We implement $\sigma'(\cdot)$ with an identity function by default.

[1] Note hyperedges with large size of nodes are usually less meaningful [157].

(2) **Quadratic.**

$$f_t(t_i, \mathbf{x}_i) = \begin{cases} \mathbf{x}_i^\top \mathbf{W}_t \mathbf{x}_i + \epsilon & \text{if } t_i = 1 \\ 0 & \text{if } t_i = 0 \end{cases} \tag{5.12}$$

Here $\mathbf{W}_t \in \mathbb{R}^{d \times d}$, and each element in $\mathbf{W}_t$ follows a Gaussian distribution. We generate f_s as:

$$f_s(\mathbf{T}, \mathbf{X}, \mathbf{H}) = \frac{1}{|\mathcal{E}_i|} \sum_{e \in \mathcal{E}_i} \sigma'(\frac{1}{|\mathcal{N}_e|^2}(\mathbf{T}_e * \mathbf{X}_e)\mathbf{W}_t(\mathbf{T}_e * \mathbf{X}_e)^\top). \tag{5.13}$$

Here $\mathbf{X}_e$ and $\mathbf{T}_e$ are the feature matrix and treatment assignment of nodes contained in hyperedge e, respectively. Here $*$ denotes element-wise multiplication.

5.3.1.2 Datasets

We follow the above process to generate potential outcomes on all three datasets. Additional details about each dataset are provided as the follows.

Contact. This dataset collects interactions recorded by wearable sensors among students at a high school [156, 157], and includes 327 nodes and 7,818 hyperedges. Each node represents a person, and each hyperedge stands for a group of individuals are in close physical proximity to each other. This contact hypergraph data allows us to simulate a hypothetical question: "how does one's face covering practice (treatment) causally affect their infection risk of an infectious disease (outcome)?". In each group contact, one may bring the virus to the surrounding environment, and thus affect other people's infection risk. Due to the lack of detailed information about each individual, apart from the potential outcome, we also generate the treatment (t_i) and the covariates ($\mathbf{x}_i$) as the follows:

$$\mathbf{x}_i \sim \mathcal{N}(0, \mathbf{I}), t_i \sim \text{Bernoulli}(\text{Sigmoid}(\mathbf{x}_i \mathbf{v}_t)), \tag{5.14}$$

where $\mathbf{I}$ is an $d \times d$ identity matrix, here we set $d = 50$. $\mathbf{v}_t$ is a d-dimensional vector where each element inside follows a Gaussian distribution. Eventually about $50\% \sim 60\%$ of the nodes are treated ($t_i = 1$) in our experiments.

GoodReads. This dataset collects book information from the book review website GoodReads[2], including the book title, authors, descriptions, reviews, and ratings [158, 159]. We take each book in the *Children* category as an instance. The bag-of-words of the book descriptions are used as the covariates of each book. Each hyperedge corresponds to each author and all books sharing the same author are in the same hyperedge. The real-world book ratings are considered as treatment assignments: for each node i, we define $t_i = 1$ if the rating score is larger than 3 and $t_i = 0$ otherwise. We aim to study the causal effect of the rating score

[2]https://www.goodreads.com/

on the sales of each book. The ratings of each author's books can establish this author's overall reputation, and thus influence the sales of other books from the same author. The final processed dataset includes 57,031 nodes (where 40% are treated) and 12,709 hyperedges. Note each book may have more than one author, and each author may have published multiple books.

Microsoft Teams. We sampled 91,391 anonymized employees of a multinational technology company and collected their aggregated telemetry data on Microsoft Teams[3]. Microsoft Teams is a workplace communication platform where users are allowed to create a group space (i.e., "team" or "channel") to enable public communication within each group. We are interested in how a user's usage of these group spaces causally affects their productivity. We process the treatment assignment into binary values by taking it as 1 if the employee has sent out at least one message in any of these group spaces during the first week of March, 2021; otherwise the treatment is assigned as 0. Each group space can be regarded as a hyperedge, where information can be shared via group discussions thus one's activeness on this platform may affect other individuals' outcomes in the same group. Employee demographics (e.g., office location, job description, work experience) were leveraged as the covariates.

5.3.2 Experiment Settings

Baselines. To investigate the effectiveness of our framework, we compare it with multiple state-of-the-art ITE estimation baselines. These baselines can be divided into the following categories:

- **No graph.** We compare the estimation results with traditional methods which do not consider graph data and spillover effects. These methods include outcome regression which is implemented by linear regression (**LR**), counterfactual regression (**CFR** [37]), causal effect variational autoencoder (**CEVAE** [38]). By comparing the proposed framework to these methods, we evaluate the effectiveness of modeling interference for ITE estimation.
- **No spillover effect in ordinary graphs.** Although assuming no spillover effect exists, the network deconfounder (**Netdeconf**) [41] captures latent confounders for ITE estimation by utilizing the network structure among instances.
- **Spillover effect in ordinary graphs.** We compare our framework with other ITE estimation baselines which can handle the pairwise spillover effect on ordinary graphs: a node representation learning based method [47] estimates ITE under network interference, including two variants: (a) **GNN + HSIC**, which is based

[3] https://www.microsoft.com/en-us/microsoft-teams

on graph neural network [160] and Hilbert Schmidt independence criterion (HSIC) [161], and (b) **GCN + HSIC**, which is based on GCN [162]. To utilize the baselines which handle ordinary graphs, we project the original hypergraph $\mathcal{H}$ to an ordinary graph $\mathcal{G} = \{\mathcal{V}, \mathcal{E}^p\}$ by setting $(v_i, v_j) \in \mathcal{E}^p$ if v_i and v_j are contained in at least one common hyperedge in $\mathcal{H}$. By comparing HyperSCI to the above baselines, we are able to evaluate the benefits of modeling high-order interferences on the original hypergraph.

Setup. We randomly partition all datasets into 60%-20%-20% training/validation/test splits. All the results are averaged over ten repeated executions. Unless otherwise specified, we set the hyperparameters as $\alpha = 0.001$, $\beta = 1.0$, $\gamma = 1.0$, $\lambda = 0.01$, the dimension for confounder representation and interference representation both as 64. We use ReLU as the activation function, and use an Adam optimizer. By default, the interference modeling component contains one hypergraph convolutional layer.

5.3.3 ITE Estimation Performance

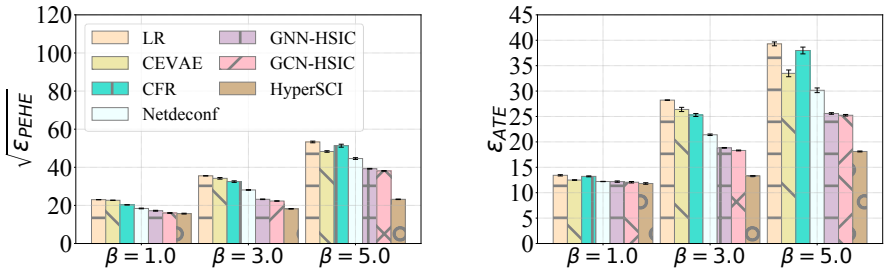
The performance of ITE estimation in hypergraph is shown in Table 5.1. From this table, we observe that HyperSCI outperforms all the baselines under different settings of outcome simulation function (in both linear and quadratic cases). As for the reasons, HyperSCI can leverage the structure information in hypergraph to model the high-order interference. In this way, it mitigates the influence of spillover effect on ITE estimation performance. Among baselines, some of them consider the pairwise network interference (GCN-HSIC and GNN-HSIC [47]), or use the graph structure to infer the hidden confounders in the ITE estimation problem (Netdeconf [41]). These methods perform better than those baselines (LR, CEVAE [38], CFR [37]) which cannot handle graph information. Furthermore, in the simulation, the hyperparameter β controls the level of hypergraph spillover effect in the outcome simulation. The ITE estimation results under different values of β are shown in Fig. 5.4. When β increases, the outcome is more strongly affected by interference, and larger performance gains can be observed from HyperSCI compared with the baselines. More details of experiments are shown in Appendix B.

5.3.4 Ablation Study

To investigate the effectiveness of different components in the proposed framework, we conduct ablation studies by considering the following variants: 1) we apply the proposed model HyperSCI on the projected graph (in a hypergraph structure) (denoted as **HyperSCI-P**); 2) we replace the hypergraph neural network module with a graph neural network module with the

TABLE 5.1. ITE estimation performance. “CT", “GR" and “MS" refer to Contact, GoodReads, and Microsoft Teams datasets, respectively.

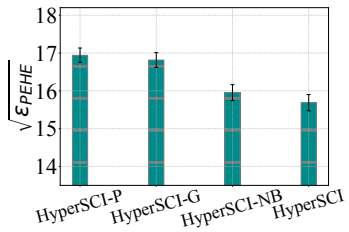
Data	Method	Linear		Quadratic	
		$\sqrt{\epsilon_{PEHE}}$	ϵ_{ATE}	$\sqrt{\epsilon_{PEHE}}$	ϵ_{ATE}
CT	LR	25.41 ±0.04	9.11 ±0.09	38.22 ±0.77	20.28 ±0.38
	CEVAE	22.88 ±1.07	8.29 ±0.69	35.28 ±0.75	18.22 ±0.76
	CFR	24.04 ±0.75	7.17 ±0.43	32.24 ±1.01	17.28 ±0.75
	Netdeconf	10.22 ±0.47	4.29 ±0.13	21.23 ±0.72	11.39 ±0.74
	GNN-HSIC	7.42 ±0.39	2.06 ±0.03	16.28 ±0.24	7.28 ±0.39
	GCN-HSIC	7.28 ±0.44	2.08 ±0.04	14.23 ±0.20	6.27 ±0.15
	HyperSCI	**3.45 ±0.27**	**1.39 ±0.03**	**9.20 ±0.09**	**2.24 ±0.07**
GR	LR	23.01 ±0.04	13.42 ±0.12	48.56 ±1.02	31.19 ±0.47
	CEVAE	22.69 ±0.03	12.49 ±0.06	45.21 ±3.10	29.22 ±0.44
	CFR	20.30 ±0.03	13.21 ±0.09	41.72 ±0.72	26.28 ±0.43
	Netdeconf	18.39 ±0.19	12.20 ±0.03	35.18 ±0.78	21.20 ±0.76
	GNN-HSIC	17.20 ±0.23	12.18 ±0.13	27.22 ±0.78	16.87 ±0.47
	GCN-HSIC	16.01 ±0.20	12.06 ±0.15	25.42 ±0.76	16.28 ±0.76
	HyperSCI	**15.68 ±0.21**	**11.81 ±0.15**	**19.23 ±0.44**	**13.33 ±0.27**
MS	LR	22.80 ±0.64	21.41± 0.74	414.17 ±3.94	192.80 ±2.97
	CEVAE	19.36 ±0.80	8.63 ±0.78	315.01 ±2.53	188.47 ±4.27
	CFR	25.23 ±0.01	18.28 ±0.02	392.56 ±4.33	189.75 ±4.80
	Netdeconf	11.11 ±0.01	9.22 ±0.03	241.02 ±2.32	147.29 ±1.04
	GNN-HSIC	9.38 ±0.44	6.91 ±0.38	114.28 ±3.62	81.21 ±2.53
	GCN-HSIC	8.27 ±0.41	6.60 ±0.48	109.57 ±3.85	77.75 ±3.93
	HyperSCI	**5.13 ±0.56**	**4.46 ±0.61**	**81.08 ±0.37**	**74.41 ±0.42**



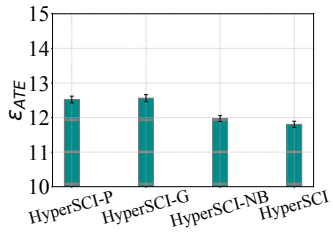
(a). $\sqrt{\epsilon_{PEHE}}$ (b). ϵ_{ATE}

FIGURE 5.4. ITE estimation performance under different values of β in linear setting on GoodReads.

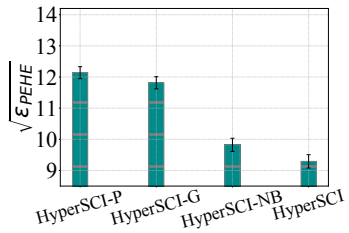
same number of layers, and then apply it on the projected graph (in an original graph structure) (**HyperSCI-G**). Notice that although both evaluated on the projected graph, **HyperSCI-G**



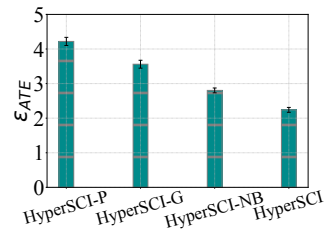
(A) GoodReads, $\sqrt{\epsilon_{PEHE}}$



(B) GoodReads, ϵ_{ATE}

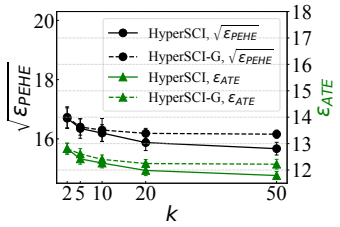


(C) Contact, $\sqrt{\epsilon_{PEHE}}$

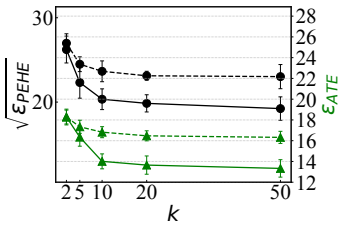


(D) Contact, ϵ_{ATE}

FIGURE 5.5. Ablation studies of different variants of our framework HyperSCI. Results (mean and standard error) are reported under the linear setting but similar patterns can be found under the quadratic setting and on all datasets.



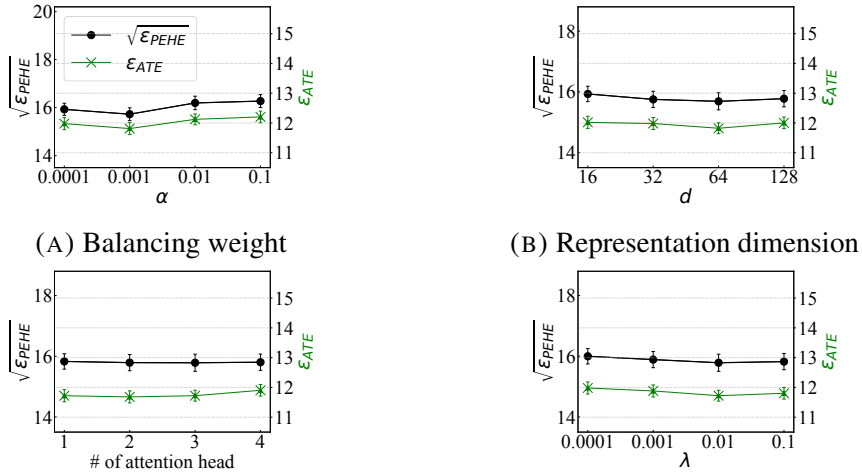
(A) GoodReads, Linear



(B) GoodReads, Quadratic

FIGURE 5.6. ITE estimation performance of HyperSCI/ HyperSCI-G on hypergraphs with hyperedge size no more than k.

handles ordinary graphs with its graph neural network module, while **HyperSCI-P** handles hypergraphs with its hypergraph neural network module; 3) we remove the balancing techniques in the framework (**HyperSCI-NB**). The ITE estimation results are reported in Fig. 5.5, where we notice significant performance gaps between **HyperSCI-P/HyperSCI-G** and HyperSCI, which imply the effectiveness of modeling the high-order relationships on hypergraphs. We also observe the ITE estimation performance degrades after removing the representation balancing modules, which indicates the effectiveness of the representation balancing techniques on mitigating the biases of ITE estimations.



(A) Balancing weight (B) Representation dimension

(C) # of attention head (D) Regularization weight

FIGURE 5.7. ITE estimation performance (mean and standard error) of the proposed framework HyperSCI under different parameters or model structures on GoodReads dataset.

5.3.5 A Closer Look at High-Order Interference

In addition to the overall ITE estimation performance, we take a closer look at high-order interference. We investigate how the proposed framework responds to hyperedges with difference sizes. More specifically, we remove the hyperedges with size larger than k, denote the modified hypergraph as $\mathcal{H}^{(k)}$, and vary the value of k. In Fig 5.6, we compare the ITE estimation performance of the proposed framework HyperSCI with its variant on the projected ordinary graph **HyperSCI-G**. We observe that: 1) When $k = 2$ (hyperedge size ≤ 2), the performance of HyperSCI-G is close to HyperSCI. Because when $k = 2$, graph convolution can be regarded as a special case of hypergraph convolution with small differences in the graph Laplacian matrix (as illustrated in [149]). Empirically this leads to a minor performance difference between HyperSCI-G and HyperSCI; 2) When k increases, the performance of ITE estimation from both methods are gradually improved, but such an improvement becomes less significant when k is larger. Besides, we notice HyperSCI consistently outperforms **HyperSCI-G** and such a difference becomes larger as k increases, indicating its efficacy on modeling high-order interference especially on large hyperedges.

5.3.6 Sensitivity Analysis

To evaluate the robustness of the proposed framework, we present the ITE estimation performance of HyperSCI under different settings of model hyper-parameters in Fig. 5.7. More specifically, we vary the value of the balancing weight from $\{0.0001, 0.001, 0.01, 0.1\}$, and

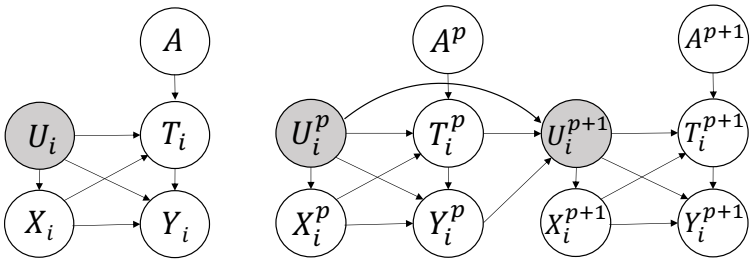
vary the representation dimension from $\{16, 32, 64, 128\}$. We also vary the number of attention head from $\{1, 2, 3, 4\}$, then change the parameter of regularization weight from $\{0.0001, 0.001, 0.01, 0.1\}$. As can be observed, our framework is generally robust to different hyper-parameter settings, but proper fine-tuning of these hyper-parameters is still beneficial for the ITE estimation performance.

CHAPTER 6

Causal Effect Estimation under Entangled Treatments

In causal effect estimation, the treatment assignment mechanism plays a key role as it determines the patterns of missing counterfactuals — the fundamental challenge of causal inference. Most existing observational studies for causal effect learning assume that the treatment is assigned individually for each unit. However, in many occasions, the treatments are pairwisely assigned for units which are connected in graph, i.e., the treatments of different units are entangled [163]. For example, in the causal question "What is the causal effect of close contact (*treatment*) on the spread of MRSA (*outcome*) in a room-sharing network?", the in-room contact in a room-sharing network is often not individually applied to each person. Instead, it is often happens between a pair of people. Neglecting the entangled treatments can impede the causal effect estimation. In this work, we study the problem of causal effect estimation with treatment entangled in a graph.

Despite a few explorations [163, 164] for entangled treatments, this problem still remains challenging due to the following challenges: 1) As discussed in [163], treatment entanglement increases the risk of misspecification of treatment effect estimator. If the entanglement through the graph is not considered, causal effect estimators tend to incorrectly attribute the observed treatment assignments to each unit's individual properties, and thus degrade the performance of causal effect estimation. To handle this entanglement problem, existing works [163, 164] assume that the treatment assignment is determined by a pre-determined function over the graph (e.g., the treatment can be the node degree on graph). However, in many occasions, this function is unknown. 2) Existing works [163, 164] rely on the unconfoundedness assumption [20] (or its weaker version) that there do not exist unobserved confounders. However, hidden confounders often exist in the real world and could lead to confounding biases. For example, a patient's behavior habits are hidden confounders which influence their physical contact and MRSA infection risk. 3) Existing works are often limited to a static setting. However, the graph, treatment, outcome, and unit covariates are naturally dynamic in many real-world scenarios. For example, the patient data is evolving over time; the causal association across different timestamps also brings more difficulties in learning causal effects.



(A) Static (B) Dynamic

FIGURE 6.1. The causal graph of the studied problem of entangled treatments in a static setting (A) and in a dynamic setting (B). The observable variables are shown in white while the unobserved ones are shown in grey.

To address the aforementioned challenges, we propose a novel framework NEAT to estimate causal effects under **N**etwork **E**nt**A**ngled **T**reatments. Specifically: 1) To handle the entangled treatment, for each node, we explicitly leverage its relevant graph topology to model the unknown treatment assignment with a learnable neural network module. 2) To tackle the hidden confounders, we take the graph structure w.r.t. each node as an *instrumental variable* (IV) [35]. IV has been widely used as an effective tool to eliminate the biases brought by hidden confounders in causal effect estimation. In the previous example, the room-sharing network is a valid IV if it is assumed to be independent of the patient's behavior habits, and its influence on the MRSA infection is fully mediated by the physical contact. A valid IV can provide randomization for the estimation of the causal effect of interest, and lead to an unbiased causal effect estimation even with hidden confounders. 3) To learn causal effects in a dynamic setting, we generalize the setting and develop our framework to handle the entangled treatment across multiple timestamps.

6.1 Problem Definition

The observational data is denoted by $\{\mathbf{X}, \mathbf{A}, \mathbf{T}, \mathbf{Y}\}^{1,\dots,P}$, which corresponds to the node features, graph adjacency matrices, treatment assignments, and observed outcomes, respectively, in P timestamps. We use $(\cdot)^p$ to denote the data in the p-th timestamp. When we focus on a static setting, or a single timestamp, we drop this superscript for notation simplicity. We assume there are N units with d_x covariates, with $\mathbf{X}^p = \{X_i^p\}_{i\in[N]}$, and for each unit i, $X_i \in \mathbb{R}^{d_x}$. The graph structure connecting these units at each timestamp is an $N \times N$ binary matrix $\mathbf{A}^p = \{A_{i,j}^p\}_{i,j\in[N]}$, where $A_{i,j}^p = 1$ when there is an edge from node i to node j, otherwise $A_{i,j}^p = 0$. The treatment is $\mathbf{T}^p = \{T_i^p\}_{i\in[N]}$. In most studies, treatment is assumed to be a binary value, but in this work, we allow it to be a d_t-size vector (e.g., a vector that describes patients' close contact patterns). The observed outcomes are denoted

by $\mathbf{Y}^p = \{Y_i^p\}_{i\in[N]}$. For each unit i at timestamp p, $Y_i^p \in \mathbb{R}$. In this section, we use bold letters (e.g., $\mathbf{X}^p$) to denote variables for all units, and use unbold letters (e.g., X_i^p) to denote variables for a single unit. For simplicity, we use the same notation for both variables and data. The causal graph for this study is shown in Fig. 6.1; in this case, not all the confounders can be directly observed or measured, thus they can often lead to biased treatment effect estimation. The hidden confounders are denoted by $\mathbf{U}^p = \{U_i\}^p_{i\in[N]}$. This work is based on the well-known Neyman-Rubin potential outcome framework [19]. We denote the potential outcomes under treatment $T = t$ as $\mathbf{Y}^p(t) = \{Y_i(t)\}^p_{i\in[N]}$. The entangled treatment is defined as follows:

DEFINITION 10. *(Entangled treatment) The treatment here can be a function $\mathcal{T}(\cdot)$ over the graph structure, the observed features, and the hidden confounders:*

$$T = \mathcal{T}(\mathbf{A}, X, U). \tag{6.1}$$

In a dynamic setting, the treatment is also a function over historical information

$$T^p = \mathcal{T}(\mathbf{A}^p, X^p, \mathbf{M}^p, U^p). \tag{6.2}$$

DEFINITION 11. *(Individual treatment effect under entangled treatments) Consider a baseline treatment as $T = t_0$, for a treatment $T = t$, the treatment effect conditioned on covariates X in a static setting is defined as:*

$$\tau(X) = \mathbb{E}[Y_i(t) - Y_i(t_0)|X]. \tag{6.3}$$

In a dynamic setting, we denote the historical information before timestamp p as $\mathbf{M}^p = \{\mathbf{X}, \mathbf{A}, \mathbf{T}, \mathbf{Y}\}^{1,\dots,p-1}$. When estimate causal effects at timestamp p, only the data no later than timestamp p can be used. We define the treatment effect at timestamp p as:

$$\tau(X^p, \mathbf{M}^p) = \mathbb{E}[Y_i^p(t) - Y_i^p(t_0)|X^p, \mathbf{M}^p]. \tag{6.4}$$

Similar as [165], we define the treatment effect for each unit i at timestamp p as $\tau_i^p = \tau(X_i^p, \mathbf{M}^p)$.

The studied problem in this work is then formally defined as:

DEFINITION 12. *(Causal effect estimation under entangled treatments) Given the observational data $\{\mathbf{X}, \mathbf{A}, \mathbf{T}, \mathbf{Y}\}^{1,\dots,P}$, we aim to estimate the treatment effect $\tau(X^p, \mathbf{M}^p)$ for each unit at each timestamp p with treatments entangled in the graph.*

6.2 Assumptions

An implicit assumption of this work is that the graph information of each node i can be represented as a variable A_i, and its samples in observational data are sufficient for us to capture the patterns it influences the treatment assignment. Similarly, we denote the historical information regarding unit i before timestamp p as M_i^p.

We assume that the outcome is generated by treatment, features, historical information, and hidden confounders as follows:

$$Y^p = \mathcal{Y}(T^p, X^p, M^p) + g(U^p), \tag{6.5}$$

where $\mathcal{Y}$ and g are (nonlinear) functions. We assume $E[g(U^p)] = 0$.

In this work, we take the graph structure as an instrumental variable (IV) for IV analysis. The following assumptions make the graph structure as a valid IV.

ASSUMPTION 2. *(Relevance) Given X^p, M^p for any random unit, the treatment is relevant to the graph structure, i.e., $A^p \not\perp\!\!\!\perp T^p | X^p, M^p$.*

ASSUMPTION 3. *(Exclusion restriction) For any random unit, the causal effect of A^p on Y^p is fully mediated by T^p, i.e., $Y^p(T, A) = Y^p(T, A')$, $\forall A \neq A'$. Here, $Y^p(T, A)$ denotes the potential outcome for treatment T and graph A at timestamp p.*

ASSUMPTION 4. *(Instrumental unconfoundedness) There is no unblocked backdoor path from A^p to Y^p, i.e., $A^p \perp\!\!\!\perp Y^p(A) | X^p, M^p$ for any random unit. Here, $Y^p(A)$ denotes the potential outcome for graph A at timestamp p.*

6.3 Proposed Method

In this section, we introduce the proposed framework NEAT. Fig. 6.2 shows an illustration of the proposed framework. Specifically, this framework contains three modules: node representation learning, entangled treatment modeling, and outcome prediction.

6.3.1 Overall Pipeline

The whole framework is designed in a classical two-stage IV study pipeline [166, 35]. Generally, in this pipeline, the first stage predicts the treatment with IVs, and the second stage estimates the potential outcomes based on the treatment predicted by the first stage. The key behind this design is that, as the IVs are unconfounded, the predicted treatment from the first

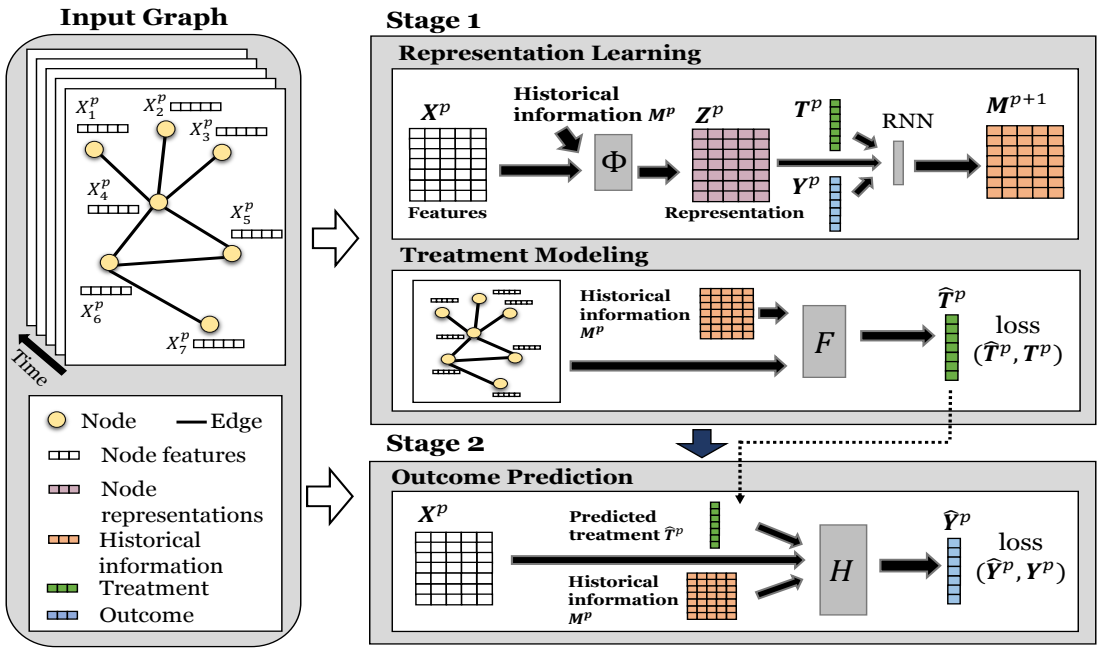


FIGURE 6.2. The proposed framework NEAT. It contains three components: node representation learning, entangled treatment modeling, and outcome prediction.

stage can provide more randomization, and thus it can help mitigate the confounding bias brought by hidden confounders.

In our framework NEAT, in the first stage, we train a treatment modeling module to predict treatment assignment for each node at each timestamp. In this module, we leverage the graph structure as IVs, and use them to capture the patterns of entangled treatment in the graph. Simultaneously, we learn a representation for each node to encode its properties, including its current features and historical information. In the second stage, we predict potential outcomes based on the original node features, the learned node representations, and the predicted treatment. In this two-stage IV framework, the biases brought by hidden confounders can then be effectively eliminated.

6.3.2 Node Representation Learning

In the real world, the treatment effects are often different for nodes with different properties. For example, the close contact may influence patients with different ages differently. To model such heterogeneity, we capture the properties of each node through node representation learning. For each node i, we learn a representation Z_i to encode its properties based on its

node features X_i:

$$Z_i = \phi(X_i). \tag{6.6}$$

Here, $\phi(\cdot)$ is implemented by a neural network module with learnable parameters.

Dynamic setting. In a time-evolving environment, as illustrated in Fig. 6.1 (b), the current properties of each node can be influenced by the historical data in previous timestamps. To capture the time-evolving properties and model the causal mechanism in a dynamic setting, for each node i, we embed the historical information before each timestamp p into a representation M_i^p with a recurrent neural network (RNN) [111, 167]. M_i^P is then incorporated into Z_i^p. At each timestamp, we update the historical embedding as:

$$M_i^p = \text{RNN}(M_i^{p-1}, T_i^{p-1}, Y_i^{p-1}, Z_i^{p-1}, X_i^{p-1}). \tag{6.7}$$

Here, we learn the representation for each node i at timestamp p with a transformation function $\Phi(\cdot)$:

$$Z_i^p = \Phi(M_i^p, X_i^p). \tag{6.8}$$

6.3.3 Entangled Treatment Modeling

The treatment function $\mathcal{T}(\cdot)$ in Eq. (6.1) or Eq. (6.2) is often not pre-determined. To better estimate treatment effects from observational data, we capture the treatment assignment patterns by training a module $F(\cdot)$ to model the conditional distribution of treatment T_i^p given $\mathbf{A}^p, X_i^p, M_i^p$. The treatment modeling module is trained in the first stage together with node representation learning:

$$\hat{T}_i^p = F(\mathbf{A}^p, X_i^p, M_i^p) = f(\mathbf{A}^p, \Phi(M_i^p, X_i^p)). \tag{6.9}$$

Treatment Entanglement. As the treatments of different units are entangled through the graph structure, to effectively capture the patterns of treatment assignment, we explicitly leverage the graph structure in the treatment modeling module. Specifically, we design this module $f(\cdot)$ based on graph neural networks (GNNs) [11, 13]. Here we use one-layer graph convoluntional network (GCN) [11] to predict the treatment as follows:

$$\hat{T}_i^p = \sigma(\hat{\mathbf{A}}^p([\mathbf{X}^p, \mathbf{Z}^p])\mathbf{W}_0), \tag{6.10}$$

where $\sigma(\cdot)$ is an activation function such as Softmax. $\hat{\mathbf{A}}^p$ is the normalized adjacency matrix calculated from the graph $\mathbf{A}^p$ beforehand with the renormalization trick [11]. Here $[\cdot, \cdot]$ stands for a concatenation operation. $\mathbf{W}_0$ denotes the parameters in GCNs.

Loss for treatment modeling. The loss for treatment prediction is denoted by $\mathcal{L}_t$. Generally, $\mathcal{L}_t$ is defined as:

$$\mathcal{L}_t = \sum_{p=1}^{P} \sum_{i=1}^{N} l_t(\hat{T}_i^p, T_i^p) = \sum_{p=1}^{P} \sum_{i=1}^{N} l_t(F(\mathbf{A}^p, X_i^p, M_i^p), T_i^p), \tag{6.11}$$

where $l_t(\cdot)$ is a loss term to measure the prediction error of treatment modeling. Noticeably, in this work, we do not restrict the data type of treatment. To handle different types of treatment, we design a different implementation for this module. More specifically, for discrete scalar treatments (e.g., whether a patient has frequent close contact), we implement treatment prediction $f(\cdot)$ as a classification model with the cross-entropy loss function; for continuous and high-dimensional treatments (e.g., a vector which describes the patient's contact patterns) we implement it as a prediction task with mean square error (MSE) loss.

6.3.4 Outcome Prediction

We train an outcome prediction module in the second stage, which predicts Y_i^p based on M_i^p, X_i^p and $\hat{T}_i^p$ with a learnable function $H(\cdot)$:

$$\hat{Y}_i^p = \int H(\hat{T}_i^p, M_i^p, X_i^p) dF(\hat{T}_i^p | \mathbf{A}^p, X_i^p, M_i^p). \tag{6.12}$$

We denote the loss function for outcome prediction by:

$$\mathcal{L}_y = \sum_{p=1}^{P} \sum_{i=1}^{N} l_y(\hat{Y}_i^p, Y_i^p), \tag{6.13}$$

where $l_y(\cdot)$ is a loss function (e.g., MSE) to measure the prediction error of the outcome. For each node i, the potential outcome w.r.t. treatment $T = t$ is predicted by $\hat{Y}_i(t) = H(t, M_i, X_i)$. We thereby estimate the treatment effect for each node i as:

$$\hat{\tau}_i = \hat{Y}_i(t) - \hat{Y}_i(t_0). \tag{6.14}$$

6.4 Experimental Evaluation

We validate the effectiveness of our proposed method with the following research questions: **RQ1:** How does the proposed framework perform under treatment entanglement compared with baselines? **RQ2:** How does the proposed framework perform under different levels of treatment entanglement and hidden confounders?

Baselines. 1) **Individual units.** These methods are based on the assumption that different units are independent, including S-Learner (**SL**) [168], causal forest (**CF**) [103], and counterfactual regression (**CFR**) [37]. 2) **Network deconfounding.** We use network deconfounder (**NetDeconf**) [41] and dynamic network deconfounder (**DNDC**) [21]. 3) **DeepIV.** This method [35] uses instrumental variables to mitigate the confounding biases. For each node i, we take the i-th row in the adjacency matrix as its IV.

Evaluation Metrics. We still adopt the two metrics introduced before: Rooted Precision in Estimation of Heterogeneous Effect (PEHE) [102] and Mean Absolute Error (ATE) [118] at each timestamp p. For all the experiments, we calculate the average values of these metrics over all timestamps.

6.4.1 Datasets and Simulation

In our experiment, we use four datasets with dynamic graph data, including synthetic, semi-synthetic, and real-world data. As it is notoriously hard to obtain the true causal models and counterfactuals from real world, on the first three datasets, we follow a regular practice to evaluate our method on data with simulated causal models. Nevertheless, we encourage our simulation to be as close to reality as possible, thus our synthetic and semi-synthetic datasets are based on graphs which are generated by real-world relational information and node features. Based on these graph data, we simulate the time-varying hidden confounders, treatment, and outcome in the following way:

6.4.1.1 Simulation

We describe the way we simulate the hidden confounders, observed node features, treatment, and potential outcomes as follows:

Hidden confounders. In a static setting, we simulate the hidden confounders as:

$$U_i \sim \mathcal{N}(0, \mu\mathbf{I}). \tag{6.15}$$

Here, $\mathbf{I}$ denotes an identity matrix of size d_u (i.e., the dimension of hidden confounders). We set $\mu = 20$ by default.

Features. If the node features are available in the dataset, we directly use them. Otherwise, we simulate them by:

$$X_i = \psi(U_i) + \epsilon_x, \tag{6.16}$$

where $\psi(\cdot)$ is a function $\mathbb{R}^{d_u} \to \mathbb{R}^{d_x}$. Here, d_x and d_u are the dimension of node features and hidden confounders, respectively. ϵ_x is a noise vector in Gaussian distribution.

Treatment. We simulate the treatment with function $\mathcal{T}$:

$$T_i = \texttt{BI}((1-\lambda)\Theta_{t,x}^\top X_i + \lambda \frac{1}{|\mathcal{N}_i|}\sum_{j\in\mathcal{N}_i}(\Theta_{t,x}^\top X_j) + \Theta_{t,u}^\top U_i + \epsilon_t), \tag{6.17}$$

where $\Theta_{t,x}, \Theta_{t,u}$ are parameter vectors with dimension d_x and d_u, respectively. Each parameter in $\Theta_{t,*}$ is in Gaussian distribution $\mathcal{N}(0, 0.5^2)$. $\mathcal{N}_i$ is the set of neighbors of node i in the graph. We use only one-hop neighbors by default. $\lambda \in [0,1]$ is the parameter that controls the strength of treatment entanglement, i.e., the larger λ is set, the stronger the graph influences the treatment assignments. $\texttt{BI}(\cdot)$ is a function that maps the input to a binary value. A regular implementation is to transform the input to a probability using a Sigmoid function, and then sample the output with Bernoulli distribution. Noticeably, we do not restrict the treatment to be a binary value. Continuous treatment can be simulated without the $\texttt{BI}(\cdot)$ function; and high-dimensional treatment with dimension d_t can be simulated by replacing the parameter vector $\Theta_{t,x}$ with a parameter matrix $\Theta_{t,x}$ with dimension $d_x \times d_t$ (similarly for $\Theta_{t,u}$). $\epsilon_t \sim \mathcal{N}(0, 0.01^2)$ is a random Gaussian noise.

Potential outcome. We simulate the potential outcomes as follows:

$$Y_i(t) = t \cdot \Theta_y^\top X_i + \Theta_0^\top X_i + \beta \Theta_u^\top U_i + \epsilon_y, \tag{6.18}$$

where Θ_y and Θ_0 are parameter vectors of dimension d_x, and Θ_u is of dimension d_u. $\beta \geq 0$ is a parameter that controls the strength of the hidden confounder. $\epsilon_y \sim \mathcal{N}(0, 0.1^2)$ is a noise.

Dynamic setting. In a dynamic setting, we simulate the historical data over time as:

$$M_i^p = \sum_{r=1}^{R}(W_u^r U_i^{R-r} + W_x^r X_i^{R-r} + W_t^r T_i^{R-r} + W_y^r Y_i^{R-r}), \tag{6.19}$$

$$U_i^p = \psi_u(M_i^p) + \epsilon_u \tag{6.20}$$

where R is the number of previous timestamps which influence the current one. We set $R = 3$ by default. Generally, the historical information at each timestamp encodes the previous hidden confounders, node features, treatments, and outcomes. Parameters W_u^r, W_x^r, W_t^r, and W_y^r control these four types of influence from timestamp $R - r$. We generate time-varying hidden confounders with a transformation over the historical information. Here, $\psi_u(\cdot)$ is a linear transformation function. $\epsilon_u \sim \mathcal{N}(0, \mathbf{I})$ is a Gaussian noise. We use the same way as Eq. (6.16) to simulate features. The treatments and outcomes are also generated similarly as above description in Eq. (6.17) and Eq. (6.18), but the historical information M_i^p is incorporated by concatenating it with X_i^p as input.

TABLE 6.1. Detailed statistics of the datasets.

Dataset	Random	Transaction	Social	MRSA
# of nodes	$30,000$	$186,509$	$52,406$	$11,044$
# of edges	$208,193$	$61,572$	$107,394$	$31,403$
# of features	32	21	16	8
# of timestamps	12	15	10	13

6.4.1.2 Datasets

We further introduce more details about each dataset. More details of data statistics are shown in Table 6.1, including the number of nodes, edges, features, and timestamps. More details can be found in Appendix C.

Random graph. This dataset contains synthetic graphs generated by the Erdös-Rényi (E-R) model [169] at each timestamp. We use NetworkX [170] to generate these graphs. Based on these graphs, we simulate other variables as described in Section 6.4.1.1.

Real-world graphs. We use two real-world dynamic graphs with each node representing a real person and each edge representing a certain type of connection between them. Based on the type of connection, these two datasets are referred as **Transaction** and **Social**, respectively. We use the covariates of people in these datasets as node features, and simulate the treatments and outcomes as described in Section 6.4.1.1.

MRSA. This dataset contains real-world hospital data for studying Methicillin-resistant Staphylococcus aureus (MRSA) infection. We construct a dynamic graph for the room-sharing relations between patients. At each timestamp, each node is a patient, and an edge exists between a pair of patients if and only if they have shared at least one room during this timestamp. The patient information such as medicine usage and length of stay are taken as node features. We investigate the causal effect of the number of in-room contacts (treatment) on MRSA infection test results (outcome). We consider there exist hidden confounders such as patients' behavior habits. In this dataset, we do not use any simulated data, and do not evaluate our causal effect estimation based on simulated counterfactuals. Instead, we use the domain knowledge regarding MRSA to confirm our findings.

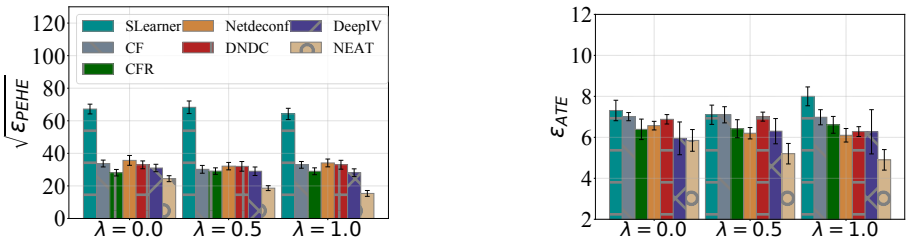
6.4.2 Performance of Different Methods

To demonstrate the effectiveness of the proposed method, in Table 6.2, we show the treatment effect estimation performance of our method and the baselines in both static and dynamic

TABLE 6.2. Performance of treatment effect estimation for different methods.

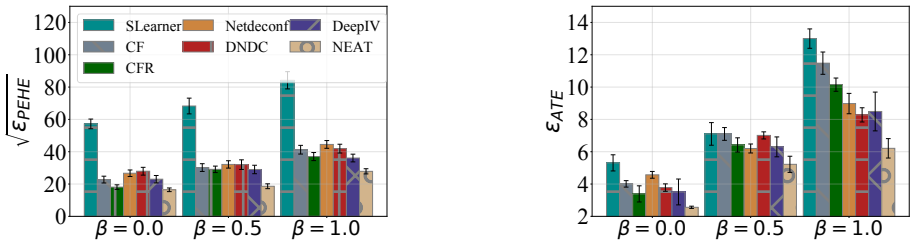
Method	Static					
	Random		Transaction		Social	
	$\sqrt{\epsilon_{PEHE}}$	ϵ_{ATE}	$\sqrt{\epsilon_{PEHE}}$	ϵ_{ATE}	$\sqrt{\epsilon_{PEHE}}$	ϵ_{ATE}
SL	67.2 ±3.0	7.3 ±0.5	40.9 ±1.4	7.1 ±0.3	48.3 ±2.5	9.2 ±0.7
CF	33.7 ±2.1	7.0 ±0.2	30.9 ±1.8	6.9 ±0.3	23.6 ±1.1	5.9 ±0.4
CFR	28.1 ±2.4	6.3 ±0.5	34.4 ±2.3	5.6 ±0.9	27.3 ±2.0	5.2 ±0.5
NetDeconf	35.6 ±3.0	6.2 ±0.3	28.6 ±2.0	5.8 ±0.7	30.5 ±2.7	6.3 ±0.4
DNDC	32.9 ±2.4	6.8 ±0.3	29.8 ±2.2	6.0 ±0.5	33.2 ±3.1	6.6 ±0.7
DeepIV	31.0 ±2.3	5.9 ±0.4	26.7 ±1.9	5.4 ±0.6	21.4 ±1.6	5.1 ±0.3
NEAT	**22.4 ±1.8**	**5.2 ±0.3**	**18.8 ±1.4**	**4.6 ±0.4**	**17.9 ±1.2**	**4.1 ±0.5**
Method	Dynamic					
	Random		Transaction		Social	
	$\sqrt{\epsilon_{PEHE}}$	ϵ_{ATE}	$\sqrt{\epsilon_{PEHE}}$	ϵ_{ATE}	$\sqrt{\epsilon_{PEHE}}$	ϵ_{ATE}
SL	69.4 ±3.1	7.7 ±0.4	55.8 ±1.8	8.4 ±0.6	45.3 ±1.4	6.5 ±0.3
CF	36.2 ±2.4	7.4 ±0.6	39.6 ±1.2	6.2 ±0.4	31.4 ±1.0	5.8 ±0.4
CFR	33.3 ±2.7	6.7 ±0.4	30.0 ±2.6	5.9 ±0.4	27.7 ±2.2	6.0 ±0.5
NetDeconf	34.0 ±2.5	6.8 ±0.7	29.4 ±1.5	6.1 ±0.5	32.9 ±2.2	5.8 ±0.8
DNDC	29.9 ±2.2	6.2 ±0.4	28.9 ±1.8	5.7 ±0.5	35.8 ±3.0	6.4 ±0.7
DeepIV	32.2 ±3.1	5.8 ±0.5	30.2 ±1.9	5.8 ±0.4	24.1 ±1.8	5.6 ±0.6
NEAT	**20.1 ±1.4**	**5.0 ±0.2**	**22.5 ±1.0**	**5.3 ±0.3**	**18.2 ±1.6**	**5.0 ±0.4**

settings. We observe that in both settings, the proposed method NEAT outperforms other baselines in different metrics. We attribute the improvement to two key factors: 1) We explicitly incorporate the graph structure to model the treatment assignment. During this process, we can better utilize the observational data for treatment effect estimation. Among the baselines, SL, CF, and CFR do not consider the graph which connects different units; NetDeconf and DNDC can leverage graph structure, but they use the graph as a proxy to infer the hidden confounders. These methods, however, do not fit in well in the problem setting studied in this work. 2) We utilize the graph structure as an instrumental variable to eliminate the confounding biases. Among the baselines, SL, CF, and CFR are based on the unconfoundedness assumption; NetDeconf and DNDC assume the hidden confounders can be inferred from the graph structure. These assumptions cannot be satisfied in our datasets. DeepIV also takes the graph information as an instrumental variable to handle hidden confounders, but its performance is impeded due to the lack of proper techniques to handle graph data.



(A) $\sqrt{\epsilon_{PEHE}}$ (B) ϵ_{ATE}

FIGURE 6.3. Treatment effect estimation performance under different levels of treatment entanglement on Random dataset.



(A) $\sqrt{\epsilon_{PEHE}}$ (B) ϵ_{ATE}

FIGURE 6.4. Treatment effect estimation performance under different levels of hidden confounders on Random dataset.

6.4.3 Performance under Different Levels of Treatment Entanglement and Confounders

To evaluate our method more comprehensively, we test it under different levels of treatment entanglement. In simulation, we control the treatment entanglement with parameter λ: the larger λ is set, the stronger the treatment assignment of each node is entangled with neighbors. Fig. 6.3 shows the causal effect estimation performance when we set λ as different values. Generally, we observe more obvious performance gain when λ is larger. This observation indicates that our method can well handle the entangled treatments by leveraging the graph structure. We only show the results on the Random dataset, but similar observations can also be found on other datasets.

We also evaluate our method under different levels of hidden confounders. In Fig. 6.4, we show the results when we change the strength of hidden confounders. Specifically, we change the strength by multiplying the hidden confounders in simulation with the parameter $\beta \geq 0$. From Fig. 6.4, it can be observed that compared with baselines, our method is more robust with hidden confounders. This is because we effectively utilize the graph as an instrumental variable to mitigate confounding biases.

TABLE 6.3. Estimated treatment effect of roommate number on MRSA infection in different populations of patients.

Population	T=0	T=1	T=2
All	0	0.025 ± 0.002	0.082 ± 0.004
General Surgery	0	0.016 ± 0.002	0.058 ± 0.003
Intensive Care	0	0.033 ± 0.003	0.119 ± 0.005
Gerontology	0	0.024 ± 0.002	0.082 ± 0.004

TABLE 6.4. Estimated treatment effect of hospital unit type on MRSA infection.

Hospital Unit Type	Estimated ATE
General Surgery	0 (baseline)
Intensive Care	0.135 ± 0.002
Gerontology	0.138 ± 0.004
Transitional Care	-0.042 ± 0.006
Internal Medicine	0.000 ± 0.002
Cardiology	0.072 ± 0.004
Orthopedic Surgery	0.000 ± 0.001
Gastroenterology	0.000 ± 0.001
Hematology and Oncology	-0.083 ± 0.005

6.4.4 Case Study on Real-world Hospital Data

Methicillin-resistant Staphylococcus aureus (MRSA) is a difficult-to-treat pathogen (owing to multi-drug resistance) that is known to spread efficiently within hospitals via contact. One important avenue of hospitalized patient-to-patient MRSA transmission is thought to be through contamination of hospital room surfaces and equipment [171]. In addition, patients may be more or less susceptible to acquiring MRSA given individual factors [172], and MRSA transmission rates may vary according to particular hospital units [173].

The MRSA dataset contains observational data including patient covariates, room-sharing information, and MRSA test record from a real-world hospital. We construct a room-sharing network, in which an edge connects two patients (nodes) if and only if they have appeared in at least one same room simultaneously. We use our method to investigate the following causal questions: (1) How does the number of in-room contacts causally influence the MRSA infection risk? (2) How do other treatments, such as the type of hospital unit (e.g. Cardiology, Internal Medicine, etc.) causally influence the MRSA infection risk? As the ground-truth

causal model is unknown, it is infeasible to evaluate our method on this dataset with the aforementioned metrics. Instead, we show some case studies and verify our key findings with domain knowledge.

For the first question, we map the number of in-room contacts into three levels of treatment. Here, treatments $0, 1, 2$ represent the roommate number from low to high. We take $T = 0$ as the control group, and calculate the treatment effect for $T = 1$ and $T = 2$ by comparing the estimated potential outcomes of them with the case of $T = 0$, respectively. Table 6.3 shows the estimated averaged treatment effect (ATE) of roommate number on MRSA infection over *all* the patients, and also shows the estimated conditional averaged treatment effect (CATE) conditioned on each subpopulation of patients in a specific group of rooms. From the results, we observe that: 1) In general, the increase in roommate number could result in an increase in MRSA infection risk. This observation holds in the whole population and different subpopulations. As MRSA is contagious through physical contact, this observation is consistent with domain knowledge. 2) The CATE of roommate number on MRSA infection is the strongest in Intensive Care and Gerontology. In Intensive Care, it is frequent for patients to share devices such as ventilators, which leads to a more severe risk of infection when the number of in-room contacts increases. Besides, most patients in Gerontology rooms are older adults with comorbidities associated with MRSA susceptibility (i.e., age >79, prior nursing home residence, antibiotic exposure, dementia, stroke, or diabetes), which brings a higher risk for acquiring MRSA from the environment with more physical contact [174].

For the second question, we take the hospital unit type as treatment, and show the estimated ATE of each hospital unit type on MRSA infection in Table 6.4. Here, we take General Surgery as the baseline treatment (control group). From Table 6.4, we observe that staying in Intensive Care and Gerontology rooms increases the MRSA infection risk most obviously. The reason might lie in the properties of these units (equipment sharing in the intensive care units, and more MRSA carriers in Gerontology). We also observe a relatively low treatment effect among beds in Transitional Care and Hematology/Oncology units. Most of these rooms are private (as opposed to other semi-private or 2-patient shared rooms), and may lead to less infection risk.

Part III

Improve Graph Machine Learning with Causality

CHAPTER 7

Overview of Part III

In Part III, we introduce our work in another direction: leveraging causality to facilitate graph machine learning in trustworthiness. We cover different aspects of trustworthiness in different chapters.

In Chapter 8, we investigate the fairness problem in graph learning methods. Specifically, we focus on counterfactual fairness, which measures fairness by comparing the model prediction for the same individual node with different values of sensitive features on the graph. We discuss the key difference between counterfactual fairness on i.i.d. data and graphs, and then propose a novel notion of graph counterfactual fairness. On top of that, we develop a framework which promotes graph counterfactual fairness and maintains good prediction performance simultaneously.

In Chapter 9, we study the explanation problem for black-box graph prediction models. Specifically, we focus on counterfactual explanation, which explains the given model by providing counterfactuals (i.e., a graph which is slightly different from the input but can achieve a different output). We tackle a series of challenges in this task, and develop a generative counterfactual explanation approach, which can be efficiently optimized and provide causality-consistent counterfactual explanations for input graphs.

In this part, we promote trustworthy graph machine learning from different views, and improve the potential of graph ML in a wider range of high-stakes applications.

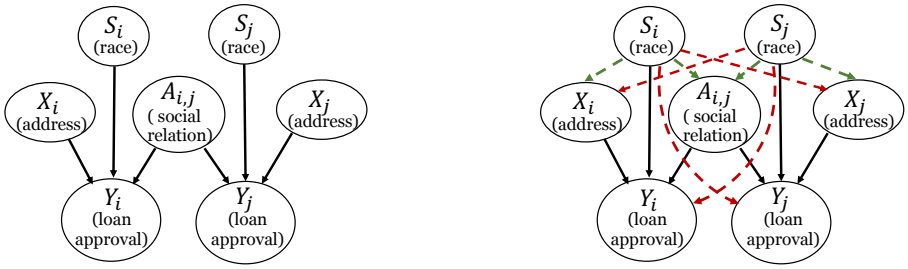
CHAPTER 8

Counterfactual Fairness in Node Representation Learning

Representation learning on graphs aims to map nodes into a latent embedding space. These node representations are often used to power downstream predictive tasks, and have become the new state-of-the-art in multiple real-world applications [175, 162, 13, 176]. However, these node representation learning approaches may overlook potential biases buried in the graph data, thus introducing algorithmic biases against subpopulations defined by certain sensitive attributes such as race, gender, and age. Consequently this may raise ethical and societal concerns, especially in high-stakes decision-making scenarios such as ranking of job applicants [177] and credit scoring [178]. For example, it would become a serious ethical issue if a bank's decision on the loan application was affected by the applicant's and their close contacts' race information.

To tackle the above problem, several approaches were proposed to assess and address the fairness of node representation learning on graphs. The majority of these methods aim to learn node representations which can elicit *statistically* fair predictions across the population [72, 100, 179, 180]. In addition, the concept of *counterfactual fairness* has been extended to graph-structured data recently [100, 181]. Different from the previous statistical notions, counterfactual fairness extends Pearl's causal structural models [110] and aims to encourage the predictions made from different versions of the same individual (a.k.a. *counterfactuals*) to be equal. For example, the prediction for one's loan application being approved should be the same regardless this applicant being Black or White.

This work falls under the umbrella of counterfactual fairness but focuses on addressing two critical limitations of existing studies [100, 181] of counterfactual fairness on graphs: 1) biases induced by **one's neighboring nodes** and 2) biases induced by **the causal relations from the sensitive attributes to other features as well as the graph structure**. We follow the previous loan application example to explain these limitations in details: i) As illustrated in Fig. 8.1(a), existing studies mostly focus on mitigating the causal influence from the sensitive attribute (race information S_i) of the i-th applicant on the prediction of the label (loan approval decision Y_i), but neglect the fact that the race information of the applicant's social contacts (S_j) can also causally affect the fairness of the prediction (as labeled using red dashed edges



(A) M' (B) M

FIGURE 8.1. Causal models generally used in existing works (M') and in this work (M). We use S_i, X_i, Y_i to denote the sensitive attribute, features, and label of any node i, and $A_{i,j} \in \{0, 1\}$ denotes the edge between node pair (i, j). Each arrow denotes a causal relation. The dashed lines denote the causal relations that the existing works do not consider.

in Fig. 8.1(b)). ii) On the other hand, existing methods may implicitly assume the sensitive attribute (S_i) has no causal effect on other variables such as node features (X_i) and the graph structure ($A_{i,j}$) so that they can safely simplify the counterfactual data generation mechanism as by just flipping the sensitive attribute values. However, we question the applicability of this assumption since such causal effect is ubiquitous in real-world scenarios. For example, one's race can causally influence their social relations as well as the residential neighborhood they live in (as labeled using green dashed edges in Fig. 8.1(b)).[1]

We argue that biases in model predictions can be induced by the aforementioned pathways. We propose a more comprehensive fairness notion on graphs – *graph counterfactual fairness*, which considers the potential biases regarding the sensitive attributes of each node and its neighboring nodes, as well as the biases led by the causal effect from sensitive attributes on other variables. With this notion, learning node representations towards graph counterfactual fairness is still challenging. It is because the causal relations among variables (as showed in Fig. 8.1(b)) are often required to obtain the counterfactuals, but these causal relations are often unknown in practice. Manually constructing the entire causal model requires extensive domain knowledge and human efforts, especially for large-scale graph data.

To address the above challenge, we propose a novel framework to learn **G**raph count**E**rfactually f**A**ir node **R**epresentations (**GEAR**). GEAR aims to learn node representations towards graph counterfactual fairness, and maintain high performance for downstream tasks such as node classification. Specifically, for each node, we minimize the discrepancy between the representations learned from the original data and the augmented counterfactuals with different sensitive attribute values.

[1]There may also exist causal relations between non-sensitive features and the graph structure, although we do not show them in Fig.8.1 for simplicity of illustration.

8.1 Problem Definition

Notations. Given a graph $\mathcal{G} = \{\mathcal{V}, \mathcal{E}, \mathbf{X}\}$, where $\mathcal{V}$ is the set of nodes, $\mathcal{E}$ is the set of edges, $\mathbf{X} = \{\mathbf{x}_i\}_{i=1}^n$ denotes the node features ($n = |\mathcal{V}|$), and $\mathbf{x}_i \in \mathbb{R}^d$ represents the features of node i. $\mathbf{A} \in \mathbb{R}^{n \times n}$ denotes the adjacency matrix of the graph $\mathcal{G}$, where $\mathbf{A}_{i,j} = 1$ if edge $i \to j$ exists, otherwise $\mathbf{A}_{i,j} = 0$. Without loss of generalization, we assume $\mathcal{G}$ is undirected and unweighted, but this work can be naturally extended to directed or weighted settings. Each node i has a sensitive attribute $s_i \in \{0, 1\}$ (we assume one single, binary sensitive attribute for simplicity, but our model can also be easily extended to multivariate or continuous sensitive attributes). $\mathbf{S} = \{s_i\}_{i=1}^n$, and s_i is included in $\mathbf{x}_i$. We denote the non-sensitive features as $\mathbf{X}^{\neg s} = \{\mathbf{x}_1^{\neg s}, ..., \mathbf{x}_n^{\neg s}\}$, where $\mathbf{x}_i^{\neg s} = \mathbf{x}_i \backslash s_i$.

Traditional node representation learning methods train an encoder $\Phi(\cdot) : \mathbb{R}^{n \times d} \times \mathbb{R}^{n \times n} \to \mathbb{R}^{n \times d'}$ to map each node to a latent representation. The learned representations for the n nodes are denoted by $\mathbf{Z} = \{\mathbf{z}_i\}_{i=1}^n$, where $\mathbf{z}_i = (\Phi(\mathbf{X}, \mathbf{A}))_i$, $\mathbf{z}_i \in \mathbb{R}^{d'}$ for any node i, and d' is the dimensionality of node representations. These representations can be used in various downstream tasks like node classification [182], link prediction [183], and graph classification [184]. $f(\cdot)$ denotes the downstream classifier/predictor. In the node classification task, let y_i denote the true label of the node i, $f(\cdot)$ takes the representation $\mathbf{z}_i$ as input, and outputs the predicted label $\hat{y}_i$.

Counterfactual fairness. Counterfactual fairness [185] is a fairness notion based on Pearl's structural causal model [110]. A *causal model* consists of a *causal graph* and *structural equations*. A causal graph is a directed acyclic graph (DAG), where each node represents a variable, and each directed edge represents a causal relationship. Structural equations describe these causal relations among variables. For variables Y, S, the value of the *counterfactual* "what would Y have been if S had been set to s?" is denoted by $Y_{S \leftarrow s}$. Based on a given causal model, a predictor $\hat{Y} = f(X)$ is *counterfactually fair* [185] if under any features $X = x$ and sensitive attribute $S = s$,

$$P(\hat{Y}_{S \leftarrow s} = y | X = x, S = s) = P(\hat{Y}_{S \leftarrow s'} = y | X = x, S = s), \tag{8.1}$$

for all y and $s' \neq s$. Here $\hat{Y}_{S \leftarrow s} = f(X_{S \leftarrow s}, s)$ denotes the prediction made on the counterfactual when S had been set to s. Intuitively, it aims to minimize the difference between predictions made on each individual and its counterfactuals with different sensitive attribute values. Ideally, the counterfactuals should be generated based on the ground truth causal model. Different from the statistical fairness notions such as equality of opportunity (EO) [186, 187] and demographic parity (DP) [188], counterfactual fairness aims to eliminate the biases led by the causal effect from the sensitive attribute on the observed variables used for model training. However, most existing works of counterfactual fairness focus on i.i.d. data.

Existing notion of counterfactual fairness on graph. Recent works [100, 181] have extended counterfactual fairness to graphs. Given a graph $X = \mathbf{X}, A = \mathbf{A}$, these works consider that an encoder $\Phi(\cdot)$ satisfies counterfactual fairness if for any node i:

$$(\Phi(X_{S_i=0}, A))_i = (\Phi(X_{S_i=1}, A))_i, \tag{8.2}$$

where $X_{S_i=0}$ and $X_{S_i=1}$ denote the node features after setting S_i as 0 and 1, respectively, while everything else does not change[2]. This notion considers fairness as minimizing the discrepancy between the representations of each node with different values of its sensitive attribute (while everything else is fixed). This notion has the following limitations: 1) it does not consider the potential biases led by the causal effect from the sensitive attribute of other nodes in the graph on the prediction of each node; 2) it implicitly assumes that the sensitive attribute has no causal effect on other features or the graph structure. In a nutshell, this fairness notion is more limited than the general counterfactual fairness notion.

Graph counterfactual fairness. To address the above limitations, in this work, we propose a novel fairness notion on graphs:

DEFINITION 13. *(Graph counterfactual fairness). An encoder $\Phi(\cdot)$ satisfies graph counterfactual fairness if for any node i:*

$$P((Z_i)_{S\leftarrow\mathbf{s}'}|X = \mathbf{X}, A = \mathbf{A}) = P((Z_i)_{S\leftarrow\mathbf{s}''}|X = \mathbf{X}, A = \mathbf{A}), \tag{8.3}$$

for all $\mathbf{s}' \neq \mathbf{s}''$, where $\mathbf{s}', \mathbf{s}'' \in \{0, 1\}^n$ are arbitrary sensitive attribute values of all nodes, $Z_i = (\Phi(X, A))_i$ denotes the node representations for node i. In other words, given a graph $X = \mathbf{X}, A = \mathbf{A}$, $\Phi(\cdot)$ should minimize the distribution discrepancy between the representations $(\Phi(X_{S\leftarrow\mathbf{s}'}, A_{S\leftarrow\mathbf{s}'}))_i$ and $(\Phi(X_{S\leftarrow\mathbf{s}''}, A_{S\leftarrow\mathbf{s}''}))_i$ for any node i.

Intuitively, this notion encourages the representations learned from the original graph and counterfactuals to be equal. The counterfactuals correspond to different cases when the sensitive attribute of the n nodes had been set to any values. For notation simplicity, in the following sections, we use $\mathbf{X}_{S\leftarrow\mathbf{s}'}$ to denote a specific value of the counterfactual "what would the node features have been if the sensitive attribute of the n nodes had been set by $\mathbf{s}'$, given the original data, i.e., node features $\mathbf{X}$ and graph structure $\mathbf{A}$?". We also use notation $\mathbf{A}_{S\leftarrow\mathbf{s}'}$ in a similar way.

In our work, we aim to develop a framework which learns node representations on graph towards graph counterfactual fairness, and maintains a good prediction performance simultaneously.

[2]In this section, we use italicized uppercase letters (e.g., S_i, X, A) to denote random variables, and use italicized lowercase letters (e.g., s_i), non-italicized bold lowercase/uppercase letters (e.g., $\mathbf{x}_i$ and $\mathbf{X}$) to denote specific realization of scalars or vectors/matrices, respectively.

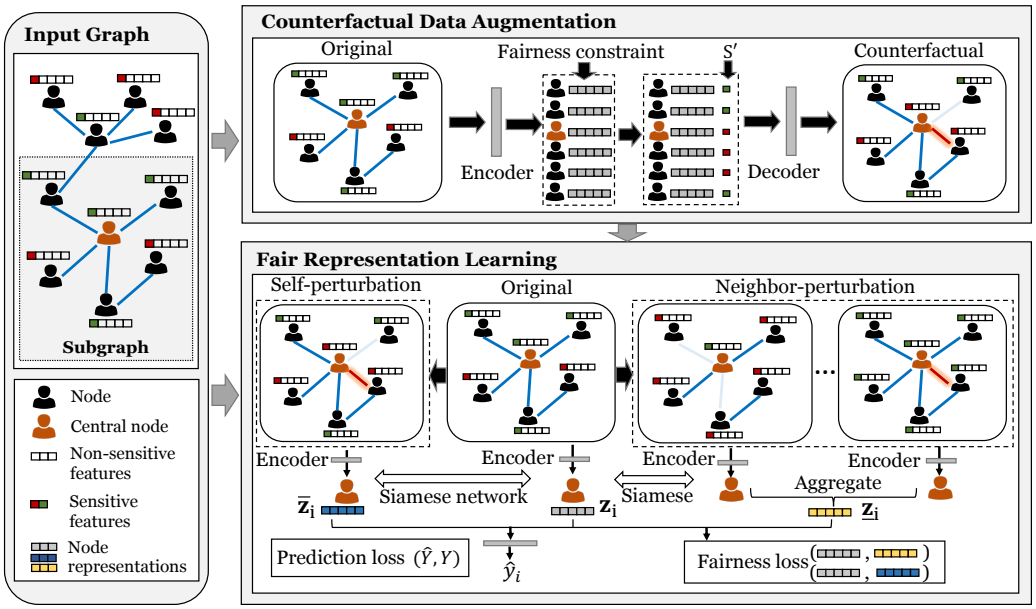


FIGURE 8.2. An illustration of the proposed framework GEAR.

8.2 Proposed Method

In this section, we propose a novel framework GEAR which aims to learn node representations for graph counterfactual fairness. As the illustration shown in Fig. 8.2, GEAR mainly includes three key components: 1) subgraph generation; 2) counterfactual data augmentation; 3) fair representation learning. In subgraph generation, GEAR extracts a context subgraph for each node, which contains the local graph structure including the node itself (central node) and its nearest neighbors with respect to precomputed importance scores. In counterfactual data augmentation, we generate counterfactuals in which the sensitive attribute of nodes in these subgraphs had been perturbed. Based on the augmented counterfactuals, the fair representation learning component leverages Siamese networks [189] to minimize the distance between the representations learned from the original data and the counterfactuals w.r.t. the same node.

8.2.1 Subgraph Generation

True causal models for graph data are often difficult to be completely obtained, especially for large-scale graphs. Based on a common observation [190, 191] that each node is mostly influenced by its nearest neighbors, we extract a subgraph $\mathcal{G}^{(i)}$ with node features $\mathbf{X}^{(i)}$ and adjacency matrix $\mathbf{A}^{(i)}$ for each node i. This subgraph extracts the context information of the central node i on $\mathcal{G}$, i.e., the subgraph of $\mathcal{G}$ which only contains the top k neighbors of node

i (including itself). These top-k neighbors are usually within several hops from the central node. Specifically, for each node i on the graph, we generate its context subgraph $\mathcal{G}^{(i)}$ with a subgraph generator $\texttt{Sub}(\cdot)$. Based on these context subgraphs, we learn the representations for their corresponding central nodes. This is based on a commonly used assumption [190] that each node has a low dependency with the nodes outside its context subgraph. Therefore, each subgraph is expected to be informative enough with respect to the graph structure relative to the central node for high-quality representation learning and counterfactual data augmentation afterwards.

Inspired by recent subgraph based node representation learning methods [191, 192], we first compute the importance scores for every node pair with personalized pagerank algorithm [193]. The importance scores can be calculated as: $\mathbf{R} = \alpha(\mathbf{I} - (1-\alpha)\bar{\mathbf{A}})$, where $\mathbf{R}$ is the importance score matrix, and each entry $\mathbf{R}_{i,j}$ describes how important node j is for node i, and $\mathbf{R}_{i,:}$ denotes the importance score vector for node i. α is a parameter in the range of $[0, 1]$, $\mathbf{I}$ is the identity matrix. $\bar{\mathbf{A}} = \mathbf{A}\mathbf{D}^{-1}$ denotes the column-normalized adjacency matrix, where $\mathbf{D}$ is the corresponding diagonal matrix with $\mathbf{D}_{i,i} = \sum_j \mathbf{A}_{i,j}$. We compute $\mathbf{R}$ in a preprocessing stage before model training for efficiency. With the importance scores, we use a $\texttt{TOP}(\cdot)$ operation to select the top-k important nodes $\mathcal{V}^{(i)}$ for each central node i, then formulate the context subgraph $\mathcal{G}^{(i)}$ as follows:

$$\mathcal{G}^{(i)} = \{\mathcal{V}^{(i)}, \mathcal{E}^{(i)}, \mathbf{X}^{(i)}\} = \{\mathbf{A}^{(i)}, \mathbf{X}^{(i)}\}, \tag{8.4}$$

$$\mathcal{V}^{(i)} = \texttt{TOP}(\mathbf{R}_{i,:}, k), \tag{8.5}$$

$$\mathbf{A}^{(i)} = \mathbf{A}_{\mathcal{V}^{(i)},\mathcal{V}^{(i)}}, \ \mathbf{X}^{(i)} = \mathbf{X}_{\mathcal{V}^{(i)},:}, \tag{8.6}$$

where the symbol : means all the indices. The above subgraph generation process (Eq. (8.4) to (8.6)) is defined as $\mathcal{G}^{(i)} = \texttt{Sub}(i, \mathcal{G}, k)$. Then the generated subgraphs are fed into encoders to learn representations of the central nodes.

8.2.2 Counterfactual Data Augmentation

To achieve graph counterfactual fairness, we pretrain a counterfactual data augmentation module before node representation learning. Here we consider a relatively simple but general causal model (as shown in Fig. 8.1(b)) to generate counterfactuals for each subgraph. Based on common observations [185], we assume that the sensitive attribute (e.g., race) is exogenous, i.e., it has no parent variables in the causal graph, and it would causally influence the other node features, the graph structure, and the labels. Based on the causal model we assume, once we intervene on the sensitive attribute, we need to model how the other variables change accordingly. To achieve this goal, we use a graph variational auto-encoder (GraphVAE) [12] based module, which takes each context subgraph as input and encodes each node in the

subgraph into a latent embedding $\mathbf{h}_i$, then a decoder reconstructs the original subgraph with the latent embeddings $\mathbf{H} = \{\mathbf{h}_1, ..., \mathbf{h}_k\}$ and the sensitive attribute values of the k nodes in this subgraph. The reconstruction loss $\mathcal{L}_r$ is as follows (we leave out the superscript $(\cdot)^{(i)}$ for notation simplicity):

$$\mathcal{L}_r = \mathbb{E}_{q(H|\mathbf{X},\mathbf{A})}[-\log(p(\mathbf{X}, \mathbf{A}|H, \mathbf{S}))] + \mathrm{KL}[q(H|\mathbf{X}, \mathbf{A})\|p(H)], \tag{8.7}$$

where $p(H)$ is a standard Normal prior distribution. We sample the embeddings $\mathbf{H}$ from $q(H|\mathbf{X}, \mathbf{A})$.

As the sensitive attribute is assumed to be exogenous, we can mitigate the causal effect from the sensitive attribute on the embeddings by removing the statistical dependency between them. To achieve this target, we use an adversarial learning method to learn embeddings which are invariant to different sensitive attribute values of each node and their neighbors. Specifically, we use a discriminator here to predict the summary of neighboring sensitive attribute values. Here we take the summary $\tilde{s}_i$ as the mean aggregation over all the nodes in the subgraph $\mathcal{G}^{(i)}$, i.e., $\tilde{s}_i = \frac{1}{|\mathcal{V}^{(i)}|}\sum_{j\in\mathcal{V}^{(i)}} s_j$. We divide the summary into B ranges to formulate it as a multivariate classification task for the discriminator $D(\cdot)$. We use a fairness constraint as follows: $\mathcal{L}_d = \sum_{b\in[B]} \mathbb{E}[\log(D(\mathbf{H}, b))]$, where the discriminator $D(\mathbf{H}, b)$ predicts the probability of whether the summary of sensitive attribute values is in range b. Based on the theoretic analysis in [194, 72], $\mathcal{L}_d$ is a regularizer to minimize the mutual information between the summary of sensitive attribute values and the embeddings. The final loss of the counterfactual data augmentation is: $\mathcal{L}_a = \mathcal{L}_r + \beta\mathcal{L}_d$, where β is a hyperparameter for the weight of fairness constraint. We use alternating stochastic gradient descent for optimization: 1) we minimize $\mathcal{L}_a$ by fixing the discriminator and updating parameters in other parts; 2) we minimize $-\mathcal{L}_a$ with respect to the discriminator while other parts fixed. To achieve graph counterfactual fairness, we expect the embeddings $\mathbf{H}$ can capture the latent variables which are informative of the input subgraph but not causally influenced by the sensitive attribute of the nodes in the subgraph. We pretrain the counterfactual data augmentation module to better disentangle different components of the framework. If more prior knowledge of the causal model is provided, we can incorporate it in counterfactual data augmentation, e.g., directly generate counterfactuals with a given causal model, and do not need to change other components in the framework.

Based on the above techniques, we conduct perturbations on the original subgraphs and obtain different types of counterfactuals. For each context subgraph $\mathcal{G}^{(i)}$, we generate two kinds of perturbations on it, including self-perturbation on the sensitive attribute of the central node, and neighbor-perturbation on the sensitive attribute of other nodes in the subgraph.

Self-perturbation. In the subgraph $\mathcal{G}^{(i)}$, we take its node embeddings, flip the sensitive attribute value of the central node s_i, then feed the embeddings and the perturbed sensitive attribute into the decoder of the pretrained counterfactual data augmentation module, and take the reconstructed subgraph as the corresponding counterfactual. The set containing the subgraphs after self-perturbation is denoted by $\bar{\mathcal{G}}^{(i)} = \{\mathcal{G}^{(i)}_{S_i \leftarrow 1-s_i}\}$.

Neighbor-perturbation. Similarly, in the subgraph $\mathcal{G}^{(i)}$, we randomly perturb the sensitive attribute values of any nodes except the central node, i.e., the nodes in the set $\mathcal{V}^{(i)}_{\neg i}$. After such perturbation, we generate a set of counterfactuals $\underline{\mathcal{G}}^{(i)} = \{\mathcal{G}^{(i)}_{S^{(i)}_{\neg i} \leftarrow \texttt{SMP}(\mathbf{S}^{(i)}_{\neg i})}\}$, where $\texttt{SMP}(\cdot)$ randomly samples specific values of the sensitive attribute out of the value space $\{0,1\}^{|\mathcal{V}^{(i)}|-1}$. We use a parameter C to denote the number of $\texttt{SMP}(\cdot)$ operations in neighbor-perturbation.

8.2.3 Fair Representation Learning

Based on the above counterfactual data augmentation, we learn fair representations which are expected to elicit the same predicted label across different counterfactuals w.r.t. the same node. To achieve this goal, we leverage Siamese networks [189] to encode the three kinds of subgraphs: original subgraphs $\mathcal{G}^{(i)}$, counterfactual subgraphs $\bar{\mathcal{G}}^{(i)}$ and $\underline{\mathcal{G}}^{(i)}$ for each central node i. For graph counterfactual fairness, we expect to learn the same representations for each central node from the three kinds of subgraphs. We train a subgraph encoder $\phi(\cdot)$ to generate the representations $\mathbf{z}_i$, $\bar{\mathbf{z}}_i$, $\underline{\mathbf{z}}_i$ for each central node i on these three kinds of subgraphs, respectively. Then we minimize the distance between the central node representations learned from the original subgraph and from the counterfactuals. We formulate the loss for graph counterfactual fairness as:

$$\mathcal{L}_f = \frac{1}{|\mathcal{V}|}\sum\nolimits_{i\in\mathcal{V}}((1-\lambda_s)d(\mathbf{z}_i, \bar{\mathbf{z}}_i) + \lambda_s d(\mathbf{z}_i, \underline{\mathbf{z}}_i)), \tag{8.8}$$

where $d(\cdot)$ is a distance metric such as cosine distance. $\lambda_s \in [0,1]$ is a hyperparameter which controls the weight of neighbor-perturbation. From the original subgraph and the counterfactuals, we obtain the node representations in the following way:

$$\mathbf{z}_i = (\phi(\mathbf{X}^{(i)}, \mathbf{A}^{(i)}))_i, \tag{8.9}$$

$$\bar{\mathbf{z}}_i = \texttt{AGG}(\{(\phi(\mathbf{X}^{(i)}_{S_i \leftarrow 1-s_i}, \mathbf{A}^{(i)}_{S_i \leftarrow 1-s_i}))_i\}), \tag{8.10}$$

$$\underline{\mathbf{z}}_i = \texttt{AGG}(\{(\phi(\mathbf{X}^{(i)}_{S^{(i)}_{\neg i} \leftarrow \texttt{SMP}(\mathbf{S}^{(i)}_{\neg i})}, \mathbf{A}^{(i)}_{S^{(i)}_{\neg i} \leftarrow \texttt{SMP}(\mathbf{S}^{(i)}_{\neg i})}))_i\}), \tag{8.11}$$

where $\phi(\cdot) : \mathbb{R}^{k\times d} \times \mathbb{R}^{k\times k} \rightarrow \mathbb{R}^{k\times d'}$ takes each subgraph as input, and embeds each node on the input subgraph into a latent representation. We take the representations of each central node i learned from the original data as $\mathbf{z}_i$, and we use $\mathbf{Z} = \{\mathbf{z}_i\}_{i=1}^n$ for downstream tasks. For the sampled counterfactual subgraphs in $\bar{\mathcal{G}}^{(i)}$ and $\underline{\mathcal{G}}^{(i)}$, we use an aggregator (e.g., mean

TABLE 8.1. Detailed statistics of the datasets.

Dataset	Synthetic	Bail	Credit
$\lvert\mathcal{V}\rvert$	$2,000$	$18,876$	$30,000$
$\lvert\mathcal{E}\rvert$	$4,120$	$311,870$	$137,377$
Feature dimension	26	18	13
Average degree	5.120	34.044	10.158
# of intra-group edges	$2,379$	$162,821$	$120,750$
# of inter-group edges	$1,741$	$149,049$	$16,627$

aggregator) $\texttt{AGG}(\cdot)$ to aggregate the representations of each central node i, and obtain the final representations $\bar{\mathbf{z}}_i$ and $\underline{\mathbf{z}}_i$.

To encode useful information of node features and graph structure into the representations, we use labels as supervision. We use the task of node classification as an example, but our framework can be naturally extended to other kinds of tasks on graph data such as link prediction. We denote the class labels as $\mathbf{Y} = \{y_1, ..., y_n\}$ for the n nodes. The prediction loss can be formulated as:

$$\mathcal{L}_p = \frac{1}{n} \sum\nolimits_{i \in [n]} l(f(\mathbf{z}_i), y_i), \tag{8.12}$$

where $l(\cdot)$ is the loss function (e.g., cross-entropy) which measures the prediction error, $f(\cdot)$ makes predictions for downstream tasks with the representations, i.e., $\hat{y}_i = f(\mathbf{z}_i)$. Finally, the overall loss function for fair representation learning is:

$$\mathcal{L} = \mathcal{L}_p + \lambda \mathcal{L}_f + \mu \|\theta\|^2, \tag{8.13}$$

where θ is the set of model parameters, λ and μ are hyperparameters controlling the weight of the graph counterfactual fairness constraint, and L_2 norm regularization, respectively.

8.3 Experimental Evaluation

We evaluate the proposed method on both synthetic and real-world graphs. The detailed statistics of these datasets are shown in Table 8.1, including the number of nodes, the number of edges, the dimension of features, the average degree, and the number of intra-group and inter-group edges with respect to the sensitive attribute.

TABLE 8.2. Comparison of the performance of node representation learning methods with respect to prediction and fairness.

Dataset	Method	Prediction Performance		Fairness			
		Accuracy (↑)	F1-score (↑)	$\triangle_{EO}$ (↓)	$\triangle_{DP}$ (↓)	δ_{CF} (↓)	R^2 (↓)
Synthetic	GCN	0.686 ± 0.015	0.687 ± 0.020	0.050 ± 0.030	0.060 ± 0.033	0.101 ± 0.030	0.085 ± 0.050
	GraphSAGE	0.712 ± 0.012	0.714 ± 0.021	0.049 ± 0.036	0.053 ± 0.042	0.172 ± 0.056	0.011 ± 0.011
	GIN	0.682 ± 0.021	0.691 ± 0.022	0.077 ± 0.053	0.081 ± 0.055	0.301 ± 0.080	0.011 ± 0.009
	C-ENC	0.665 ± 0.023	0.671 ± 0.031	0.030 ± 0.024	0.048 ± 0.026	0.633 ± 0.013	0.085 ± 0.016
	FairGNN	0.668 ± 0.020	0.672 ± 0.026	$\mathbf{0.025 \pm 0.021}$	$\mathbf{0.042 \pm 0.033}$	0.678 ± 0.014	0.091 ± 0.021
	NIFTY-GCN	0.618 ± 0.035	0.640 ± 0.037	0.172 ± 0.110	0.199 ± 0.106	0.208 ± 0.090	0.105 ± 0.081
	NIFTY-SAGE	0.664 ± 0.041	0.682 ± 0.073	0.031 ± 0.027	0.048 ± 0.027	0.147 ± 0.071	0.008 ± 0.005
	GEAR	$\mathbf{0.718 \pm 0.018}$	$\mathbf{0.724 \pm 0.022}$	0.052 ± 0.038	0.064 ± 0.038	$\mathbf{0.002 \pm 0.002}$	$\mathbf{0.007 \pm 0.006}$
Bail	GCN	0.838 ± 0.017	0.782 ± 0.023	0.023 ± 0.019	0.075 ± 0.014	0.132 ± 0.059	0.075 ± 0.028
	GraphSAGE	$\mathbf{0.854 \pm 0.026}$	$\mathbf{0.804 \pm 0.032}$	0.039 ± 0.022	0.086 ± 0.039	0.088 ± 0.047	0.069 ± 0.011
	GIN	0.731 ± 0.058	0.656 ± 0.084	0.041 ± 0.023	0.065 ± 0.034	0.143 ± 0.069	0.047 ± 0.036
	C-ENC	0.842 ± 0.047	0.792 ± 0.014	0.038 ± 0.022	0.069 ± 0.020	0.040 ± 0.025	0.078 ± 0.024
	FairGNN	0.835 ± 0.024	0.784 ± 0.021	0.046 ± 0.013	0.074 ± 0.026	0.042 ± 0.032	0.086 ± 0.016
	NIFTY-GCN	0.752 ± 0.065	0.669 ± 0.050	0.019 ± 0.015	$\mathbf{0.036 \pm 0.022}$	0.031 ± 0.017	0.025 ± 0.018
	NIFTY-SAGE	0.823 ± 0.048	0.723 ± 0.103	$\mathbf{0.014 \pm 0.006}$	0.047 ± 0.015	0.013 ± 0.011	0.044 ± 0.020
	GEAR	0.852 ± 0.026	0.800 ± 0.031	0.019 ± 0.023	0.058 ± 0.017	$\mathbf{0.003 \pm 0.002}$	$\mathbf{0.038 \pm 0.012}$
Credit	GCN	0.698 ± 0.028	0.794 ± 0.027	0.087 ± 0.035	0.108 ± 0.031	0.042 ± 0.029	0.022 ± 0.014
	GraphSAGE	0.739 ± 0.009	0.821 ± 0.008	0.094 ± 0.033	0.109 ± 0.030	0.062 ± 0.036	0.014 ± 0.004
	GIN	0.713 ± 0.018	0.805 ± 0.016	0.121 ± 0.042	0.130 ± 0.037	0.123 ± 0.060	0.025 ± 0.012
	C-ENC	0.695 ± 0.011	0.786 ± 0.012	0.098 ± 0.025	0.104 ± 0.042	0.100 ± 0.024	0.048 ± 0.012
	FairGNN	0.683 ± 0.053	0.780 ± 0.042	0.175 ± 0.035	0.187 ± 0.036	0.105 ± 0.053	0.056 ± 0.018
	NIFTY-GCN	0.697 ± 0.007	0.792 ± 0.007	0.097 ± 0.024	0.106 ± 0.021	0.004 ± 0.004	0.017 ± 0.003
	NIFTY-SAGE	0.751 ± 0.023	0.833 ± 0.020	$\mathbf{0.075 \pm 0.021}$	$\mathbf{0.094 \pm 0.019}$	0.004 ± 0.003	0.011 ± 0.003
	GEAR	$\mathbf{0.755 \pm 0.011}$	$\mathbf{0.835 \pm 0.008}$	0.086 ± 0.018	0.104 ± 0.013	$\mathbf{0.001 \pm 0.001}$	$\mathbf{0.010 \pm 0.003}$

8.3.1 Datasets

A synthetic dataset and two real-world datasets are used in the experiments. In the synthetic dataset, we create a causal model with which we can fully manipulate the data generation process. More specifically, in this synthetic dataset, we generate the features, latent embeddings, graph structure, and labels as below:

$$S_i \sim \text{Bernoulli}(p),\ Z_i \sim \mathcal{N}(0, \mathbf{I}),\ X_i = \mathcal{S}(Z_i) + S_i\mathbf{v}, \tag{8.14}$$

$$P(A_{i,j} = 1) = \sigma(\cos(Z_i, Z_j) + a\mathbf{1}(S_i = S_j)), Y_i = \mathcal{B}(\mathbf{w}Z_i + w_s\frac{\sum_{j\in\mathcal{N}_i} S_j}{|\mathcal{N}_i|}), \tag{8.15}$$

where we sample the sensitive attribute with Bernoulli distribution, where $p = 0.4$ is the probability of $S_i = 1$. We sample latent embeddings $Z_i \in \mathbb{R}^{d_z}$ from a Gaussian distribution, where $d_z = 50$. Z_i influences the node features and the graph structure for each node i, and $\mathcal{S}(\cdot)$ denotes a sampling operation which randomly selects $d = 25$ dimensions out of the latent embeddings to form the observed features X_i. $\mathbf{v} \in \mathbb{R}^d, \mathbf{v} \sim \mathcal{N}(0, \mathbf{I})$ controls the influence of the sensitive attribute on other features. We simulate the probability of each edge (i, j) based on the cosine similarity between Z_i and Z_j, as well as whether their sensitive attribute values are equal. Here $\mathbf{1}(\cdot)$ is an indicator function which outputs 1 when the input statement is true

and 0 otherwise. We set parameter $a = 0.01$. Then we sum up the above similarity between (Z_i, Z_j) and the indicator function's output of $(S_i = S_j)$, and map it into a range of $[0, 1]$ with a Sigmoid function $\sigma(\cdot)$ to compute the link probability between (i, j). $\mathbf{w} \in \mathbb{R}^{d_z}$ contains parameters sampled from Normal distribution. We average each node's and their one-hop neighbors' sensitive attribute values and use it into label generation with weight $w_s = 0.5$. In $\mathcal{B}(\cdot)$, we map Y_i into a binary value. Specifically, we first compute the mean value of Y_i over all nodes, and set $Y_i = 1$ if it is larger than the mean value, otherwise $Y_i = 0$.

As for the real-world graphs, we use: 1) *Bail* [100]: This graph contains the data of defendants who got released on bail at the U.S state courts. In this graph, each node represents a defendant, each edge between a pair of nodes represents their similarity of criminal records and demographics. We use the defendants' race as the sensitive attribute. The task is to classify defendants into bail (not tend to commit a violent crime if released) or no bail. 2) *Credit defaulter* [100]: This graph contains people's default payment information. In this graph, each node represents an individual, each edge between a pair of nodes represents the similarity of their spending and payment patterns. We use their age as the sensitive attribute, and the task is to predict that their default ways of payment is credit card or not.

To evaluate the graph counterfactual fairness of the proposed method, we need to generate the ground-truth counterfactuals with the perturbations on different nodes' sensitive attribute. On the synthetic dataset, the counterfactuals can be generated based on the predefined causal model. On the real-world graphs, the ground-truth causal models are unknown, so we use a simple causal model and fit the observed data, and use the learned parameters in the fitted causal model to generate the counterfactuals for the whole graph. More specifically, we first use a Naïve Bayes model to learn $P(X_i|S_i)$, and then update the counterfactual features by $(\mathbf{x}_i)_{S_i \leftarrow 1-s_i} = \mathbb{E}[X_i|S_i = 1 - s_i] - \mathbb{E}[X_i|S_i = s_i] + \mathbf{x}_i$. We use $(\cdot)^{CF}$ to denote any counterfactuals. Then we generate the counterfactual graph edge for each (i, j) based on the following rules:

$$P(A_{i,j}^{CF} = 1) = \sigma(\cos(X_i^{CF} \backslash S_i^{CF}, X_j^{CF} \backslash S_j^{CF}) + \gamma \mathbf{1}(S_i^{CF}, S_j^{CF})), \tag{8.16}$$

where $\cos(\cdot)$ is cosine similarity. We use $\sigma(\cdot)$ to map its input into a range $[0, 1]$ to compute the link probability for any node pairs in the counterfactual graph. We fit the data with this causal model and learn the parameter γ. For evaluation, we use the counterfactual data generated by the causal model and learned parameters.

As discussed in [195], there might be multiple possible causal models in the real-world data, so we have tried different causal models to fit the real-world data. Due to the space limit, we only show the results based on the causal model as described above, but the observations over all the experiments are generally consistent.

TABLE 8.3. Comparison of the performance of different variants of GEAR.

Dataset	Method	Prediction Performance		Fairness			
		Accuracy (↑)	F1-score (↑)	$\triangle_{EO}$ (↓)	$\triangle_{DP}$ (↓)	δ_{CF} (↓)	R^2 (↓)
Synthetic	-NS	0.722 ± 0.023	0.726 ± 0.025	0.061 ± 0.044	0.071 ± 0.024	0.005 ± 0.002	0.011 ± 0.006
	-NN	0.725 ± 0.026	$\mathbf{0.727 \pm 0.016}$	0.066 ± 0.048	0.086 ± 0.033	0.008 ± 0.004	0.016 ± 0.005
	-NP	$\mathbf{0.729 \pm 0.022}$	$\mathbf{0.727 \pm 0.027}$	0.094 ± 0.051	0.116 ± 0.063	0.012 ± 0.005	0.023 ± 0.018
	-NC	0.720 ± 0.019	0.725 ± 0.018	0.058 ± 0.042	0.069 ± 0.028	0.006 ± 0.003	0.012 ± 0.004
	GEAR	0.718 ± 0.018	0.724 ± 0.022	$\mathbf{0.052 \pm 0.038}$	$\mathbf{0.064 \pm 0.038}$	$\mathbf{0.002 \pm 0.002}$	$\mathbf{0.007 \pm 0.006}$
Bail	-NS	0.854 ± 0.020	0.802 ± 0.014	0.027 ± 0.024	0.066 ± 0.020	0.014 ± 0.007	0.056 ± 0.018
	-NN	0.855 ± 0.024	$\mathbf{0.804 \pm 0.024}$	0.032 ± 0.027	0.068 ± 0.023	0.022 ± 0.009	0.058 ± 0.016
	-NP	$\mathbf{0.860 \pm 0.022}$	$\mathbf{0.804 \pm 0.031}$	0.041 ± 0.028	0.073 ± 0.028	0.027 ± 0.010	0.064 ± 0.019
	-NC	0.853 ± 0.024	0.801 ± 0.019	0.025 ± 0.027	0.064 ± 0.014	0.007 ± 0.004	0.053 ± 0.014
	GEAR	0.852 ± 0.026	0.800 ± 0.031	$\mathbf{0.019 \pm 0.023}$	$\mathbf{0.058 \pm 0.017}$	$\mathbf{0.003 \pm 0.002}$	$\mathbf{0.049 \pm 0.012}$
Credit	-NS	0.749 ± 0.014	0.831 ± 0.024	0.089 ± 0.018	0.109 ± 0.038	0.016 ± 0.027	0.012 ± 0.005
	-NN	0.751 ± 0.012	0.832 ± 0.018	0.092 ± 0.034	0.114 ± 0.043	0.020 ± 0.047	0.013 ± 0.004
	-NP	0.753 ± 0.018	$\mathbf{0.836 \pm 0.017}$	0.099 ± 0.043	0.122 ± 0.049	0.028 ± 0.054	0.016 ± 0.007
	-NC	0.749 ± 0.015	0.830 ± 0.011	0.088 ± 0.012	0.106 ± 0.011	0.004 ± 0.002	0.013 ± 0.004
	GEAR	$\mathbf{0.755 \pm 0.011}$	0.835 ± 0.008	$\mathbf{0.086 \pm 0.018}$	$\mathbf{0.104 \pm 0.013}$	$\mathbf{0.001 \pm 0.001}$	$\mathbf{0.010 \pm 0.003}$

8.3.2 Experiment Settings

Metrics. We evaluate the proposed framework with respect to two aspects: prediction performance and fairness. To evaluate the prediction performance, we use the widely-used node classification metrics: accuracy and F1-score. To measure the fairness of the representations, we first use two metrics which are commonly used in statistical fairness: $\triangle_{SP} = |P(\hat{Y}_i|S_i = 0) - P(\hat{Y}_i|S_i = 1)|$, and $\triangle_{EO} = |P(\hat{Y}_i|Y_i = 1, S_i = 0) - P(\hat{Y}_i|Y_i = 1, S_i = 1)|$.

To evaluate graph counterfactual fairness, we design a metric δ_{CF}:

$$\delta_{CF} = |P((\hat{Y}_i)_{S \leftarrow \mathbf{s}'}|X = \mathbf{X}, A = \mathbf{A}) - P((\hat{Y}_i)_{S \leftarrow \mathbf{s}''}|X = \mathbf{X}, A = \mathbf{A})|, \tag{8.17}$$

where $\mathbf{s}', \mathbf{s}'' \in \{0, 1\}^n$ are arbitrary values of sensitive attribute of all nodes. As there are too many different counterfactuals (e.g., there are 2^n cases for a graph with n nodes), it is difficult to evaluate the difference of predictions under all these counterfactuals. Therefore, we evaluate the graph counterfactual fairness of the proposed model in the following way: on each dataset, we control the rate of sensitive subgroup population and randomly perturb the sensitive attribute of all nodes. More specifically, we randomly select $0\%, 50\%, 100\%$ nodes, and set their sensitive attribute values to be 1, while set the sensitive attribute of other nodes to be 0. With such perturbations, we generate counterfactual data for the whole graph with different ratios of sensitive subgroup, based on the causal model described in Section 8.3.1. Intuitively, these perturbations implicitly control the distribution of the sensitive attribute in each node's neighborhood, and we take the averaged ratio of nodes which flip their predicted labels as an estimation for δ_{CF}. Besides, we also compute the R-square $R^2(\hat{Y}_i, \tilde{S}_i)$ to measure how well a linear regression predictor for $\hat{Y}_i$ can be explained by the summary of

the neighboring sensitive attribute values for any node i. Here we use the mean aggregator over the sensitive attribute values of all one-hop neighbors and each node i itself to compute the sensitive attribute summary $\tilde{S}_i$. This R-square metric can reflect the statistical dependency between $\hat{Y}_i$ and $\tilde{S}_i$.

Baselines. We compare the proposed framework with several state-of-the-art node representation learning methods. We divide them into two categories: 1) node representation learning methods without fairness constraints: these methods only aim to encode useful information from the input graph and improve the prediction performance in downstream tasks. We use graph convoluntional network (**GCN**) [162], **GraphSAGE** [190], and Graph Isomorphism Network (**GIN**) [176] as baselines; 2) fair representation learning methods on graphs: these methods target on learning fair node representation on graphs. Among them, **C-ENC** [72] and **FairGNN** [179] enforces fairness with an adversarial discriminator to predict the sensitive attribute; **NIFTY** [100] enforces fairness by maximizing the similarity of representations learned from the original graph and their augmented counterfactual graphs, where the sensitive attribute values of all nodes are flipped, while other parts remain unchanged. We use two variants of it with GCN or GraphSAGE encoders, denoted by **NIFTY-GCN** and **NIFTY-SAGE**, respectively.

8.3.3 Prediction Performance and Fairness

The performance of prediction and fairness is shown in Table 8.2. The best results are shown in **bold**, and the runner-up results are underlined. Generally speaking, we have the following observations: 1) The proposed model GEAR shows comparable prediction performance with the state-of-the-art node representation learning methods, and it outperforms all the fair node representation learning methods in prediction; 2) The proposed model outperforms all the other fair node representation learning methods in δ_{CF} and R^2. These two fairness metrics explicitly consider the causal/statistical relation between the neighboring sensitive attribute and the model prediction, thus this observation validates the effectiveness of our framework in mitigating the biases from neighbors. Besides, GEAR also performs well in other fairness metrics $\triangle_{EO}$ and $\triangle_{DP}$. The baseline NIFTY also has good performance in graph counterfactual fairness, because NIFTY also generates counterfactuals during training. Although NIFTY does not explicitly consider the causal effect from neighbors' sensitive attributes on each node, its counterfactuals still implicitly promote graph counterfactual fairness. However, our method still outperforms all these fairness methods mainly for two reasons: a) GEAR generates multiple versions of counterfactuals with self-perturbation and neighbor-perturbation. It has better coverage of the space of possible counterfactuals, while NIFTY only generates one counterfactual by flipping all nodes' sensitive attribute values, here

the influence from the neighbors' sensitive attribute may counteract with each other; b) GEAR generates counterfactuals which include changes in both features and graph structure after modifying the sensitive attribute, rather than simply changing the sensitive attribute. More specifically, the counterfactual augmentation component in GEAR removes biases caused by misusing the descendants of the sensitive attribute in node representation learning.

8.3.4 Ablation Study

In the ablation study, we compare different variants of GEAR to verify the effectiveness of different components. We first remove self-perturbations, and denote this variant as GEAR-NS. Next, we remove neighbor-perturbations, denoted by GEAR-NN. We then remove all the perturbations, and denote this variant as GEAR-NP. We remove the counterfactual data augmentation module, just flip the sensitive attribute values, and denote this variant as GEAR-NC. The model performance of these variants is shown in Table 8.3. We observe that all the variants perform worse than GEAR with respect to fairness. These results validate the effectiveness of different components in GEAR for learning fair node representations.

CHAPTER 9

Counterfactual Explanation for Graph Machine Learning Models

Motivation. To facilitate explainability in opaque ML models, explainable artificial intelligence (XAI) has recently attracted significant attention. A special class of XAI – *counterfactual explanation* (CFE) [196] promotes model explainability by answering the question: "*how should the input features X be slightly perturbed to new features X' to obtain a desired label for a specific instance?*". The original instance whose prediction needs to be explained is called an *explainee instance*, and the generated instances after perturbation are referred to as "counterfactual explanations". Different from traditional CFE studies [196, 197, 198, 199] on tabular or image data, recently, CFE on graphs is also an emerging field in many domains with graph structure data such as molecular analysis [200] and professional networking [201].

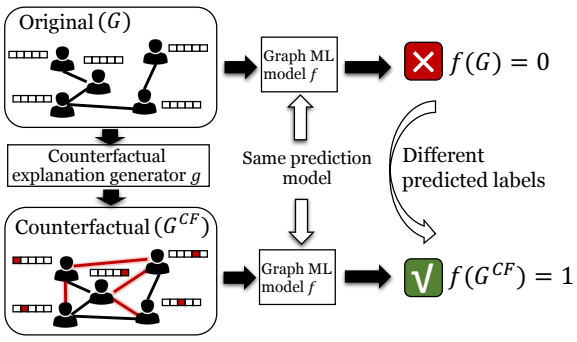


FIGURE 9.1. An example of CFE on graphs.

For example, consider a grant application prediction [202] model with each input instance as a graph representing a research team's collaboration network, where each node represents a team member, and each edge signifies a collaboration relationship between them. Team leaders can improve their teams for next application by changing the original graph according to the counterfactual with a desired predicted label (application being granted). If the counterfactual is more dense than the original, the team leader may then encourage more team collaborations. Generally, CFE promotes human interpretation through the comparison between X and X'. To this end, we investigate the problem of generating counterfactual explanations on graphs. As shown in Fig. 9.1, given a prediction model f on graphs, for a graph instance G, we aim to generate counterfactuals (e.g., G^{CF}) which are slightly different from G w.r.t. their node features or graph structures to elicit a desired model prediction. Specifically, we focus on graph-level prediction without any assumptions of the prediction model type and its model access, i.e., f can be a black box with unknown structure.

Recently, a few studies [25, 200, 203, 204, 24, 205] explore to extend CFEs into graphs. However, this problem still remains a daunting task due to the following key challenges: 1) **Optimization:** Different from traditional data, the space of perturbation operations on graphs (e.g., add/remove nodes/edges) is discrete, disorganized, and vast, which brings difficulties for optimization in CFE generation. Most existing methods [200, 204] search for graph counterfactuals by enumerating all the possible perturbation operations on the current graph. However, such enumeration on graphs is of high complexity, and it is also challenging to solve an optimization problem in such a complex search space. Few graph CFE methods which enable gradient-based optimization either rely on domain knowledge [200] or assumptions [203] about the prediction model to facilitate optimization. However, these knowledge and assumptions limit their applications in different scenarios. 2) **Generalization:** The discrete and disorganized nature of graphs also brings challenges for the generalization of CFE methods on unseen graphs, as it is hard to sequentialize the process of graph CFE generation and then generalize it. Most existing CFE methods on graphs [25, 24] solve an optimization problem for each explainee graph separately. These methods, however, cannot be generalized to new graphs. 3) **Causality:** It is challenging to generate counterfactuals that are consistent with the underlying causality. Specifically, causal relations may exist among different node features and the graph structure. In the aforementioned example, for each team, after establishing more collaborations, the team culture may be causally influenced. Incorporating causality can generate more realistic and feasible counterfactuals [198], but most existing CFE methods either cannot handle causality, or require too much prior knowledge about the causal relations in data.

To address the aforementioned challenges, in this work, we propose a novel framework — generative **C**ounterfactua**L** **E**xpl**A**nation gene**R**ator for graphs (CLEAR). At a high level, CLEAR is a generative, model-agnostic CFE generation framework for graph prediction models. For any explainee graph instance, CLEAR aims to generate counterfactuals with slight perturbations on the explainee graph to elicit a desired predicted label, and the counterfactuals are encouraged to be in line with the underlying causality. More specifically, to facilitate the optimization of the CFE generator, we map each graph into a latent representation space, and output the counterfactuals as a probabilistic fully-connected graph with node features and graph structure similar as the explainee graph. In this way, the framework is differentiable and enables gradient-based optimization. To promote the generalization of the CFE generator on unseen graphs, we propose a generative way to construct the counterfactuals. After training the CFE generator, it can be efficiently deployed to generate (multiple) counterfactuals on unseen graphs, rather than retraining from scratch. To generate more realistic counterfactuals without explicit prior knowledge of the causal relations, inspired by the recent progress in nonlinear independent component analysis (ICA) [206] and its connection with causality [15],

we make an exploration to promote causality in counterfactuals by leveraging an auxiliary variable to better identify the latent causal relations.

9.1 Problem Definition

Preliminaries. A graph $G = (X, A)$ is specified with its node feature X and adjacency matrix A. We have a graph prediction model $f : \mathcal{G} \to \mathcal{Y}$, where $\mathcal{G}$ and $\mathcal{Y}$ represent the space of graphs and labels, respectively. In this work, we assume that we can access the prediction of f for any input graph, but we do not assume the access to any knowledge of the prediction model itself. For a graph $G \in \mathcal{G}$, we denote the output of the prediction model as $Y = f(G)$. A counterfactual $G^{CF} = (X^{CF}, A^{CF})$ is expected to be similar as the original explainee graph G, but the predicted label for G^{CF} made by f (i.e., $Y^{CF} = f(G^{CF})$) should be different from $f(G)$. With a desired label Y^* (here $Y^* \neq Y$), the counterfactual G^{CF} is considered to be *valid* if and only if $Y^* = Y^{CF}$.

Suppose we have a set of graphs sampled from the space $\mathcal{G}$, and different graphs may have different numbers of nodes and edges. A counterfactual explanation generator can generate counterfactuals for any input graph G w.r.t. its desired predicted label Y^*. As aforementioned, most existing CFE methods on graphs have limitations in three aspects: optimization, generalization, and causality. Next, we provide more background on causality. The causal relations between different variables (e.g., node features, degree, etc.) in the data can be described with a structural causal model (SCM):

DEFINITION 14. *(Causality in CFE) For an explainee graph G, a counterfactual G^{CF} satisfies causality if the change from G to G^{CF} is consistent with the underlying structural causal model.*

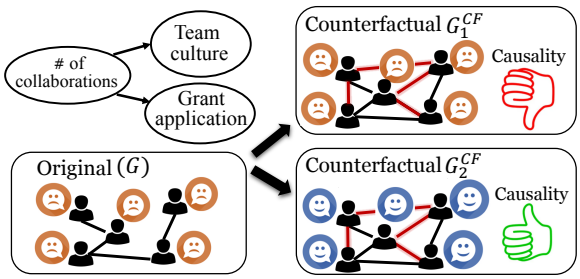


FIGURE 9.2. An example of causality in CFE.

Example: In the aforementioned grant application example, for a research team that has been rejected for an application before (as the graph G in Fig. 9.2), to get the next application approved, a valid counterfactual may suggest this team to improve the number of collaborations between team members. Based on real-world observations, we assume that an additional causal relation exists: the number of team collaborations causally affects the team culture. For example, for the same team, if more collaborations had been

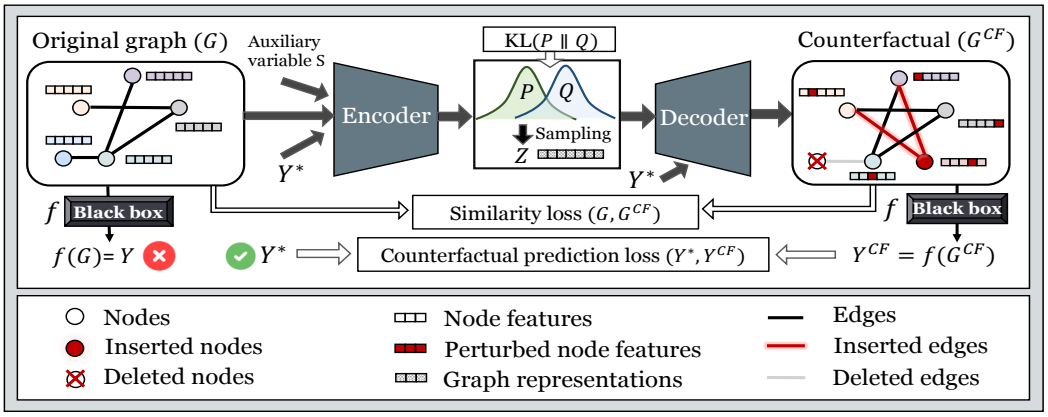


FIGURE 9.3. An illustration of the proposed framework CLEAR.

established, then the team culture should have been improved in terms of better member engagement and respect for diversity. The SCM is illustrated in Fig. 9.2, where we leave out the exogenous variables for simplicity. Although the team culture usually does not affect the result of the grant application, the counterfactuals with team culture changed correspondingly when the number of collaborations changes are more consistent with the ground-truth SCM. G_2^{CF} in Fig. 9.2 shows an example of such counterfactuals. In contrast, if a counterfactual improves a team's number of collaborations alone without improving the team culture (see G_1^{CF} in Fig. 9.2), then it violates the causality. As discussed in [198], traditional CFE methods often optimize on a single instance, and are prone to perturb different features independently, thus they often fail to satisfy the causality.

9.2 Proposed Method

In this section, we describe a novel generative framework — CLEAR, which addresses the problem of counterfactual explanation generation for graphs. First, we introduce its backbone CLEAR-VAE to enable optimization on graphs and generalization on unseen graph instances. This backbone is based on a graph variational auto-encoder (VAE) [207] mechanism. On top of CLEAR-VAE, we then promote the causality of CFEs with an auxiliary variable.

9.2.1 CLEAR-VAE: Backbone of Graph Generative Counterfactual Explanations

Different from most existing methods [25, 24] for CFE generation on graphs which focus on a single graph instance, CLEAR is based on a generative backbone CLEAR-VAE which can efficiently generate CFEs for different graphs after training, even for the graphs that do

not appear in the training data. As shown in Fig. 9.3, CLEAR-VAE follows a traditional graph VAE [207] architecture with an encoder and a decoder. The encoder maps each original graph $G = (X, A)$ into a latent space as a representation Z, then the decoder generates a counterfactual G^{CF} based on the latent representation Z. Following the VAE mechanism as in [207, 208] and its recent application in CFE generation for tabular data [198, 209], the optimization objective is based on the evidence lower bound (ELBO), which is a lower bound for the log likelihood $\ln P(G^{CF}|Y^*, G)$. Here, $P(G^{CF}|Y^*, G)$ is the probability of the generated counterfactual G^{CF} conditioned on an input explainee graph G and a desired label Y^*. The ELBO for CLEAR-VAE can be derived as follows:

$$\ln P(G^{CF}|Y^*, G) \geq \mathbb{E}_Q[\ln P(G^{CF}|Z, Y^*, G)] - \text{KL}(Q(Z|G, Y^*)\|P(Z|G, Y^*)), \quad (9.1)$$

where Q refers to the approximate posterior distribution $Q(Z|G, Y^*)$, and $\text{KL}(\cdot||\cdot)$ means the Kullback-Leibler (KL) divergence. The first term $P(G^{CF}|Z, Y^*, G)$ denotes the probability of the generated counterfactual conditioned on the representation Z and the desired label Y^*. Due to the lack of ground-truth counterfactuals, it is hard to directly optimize this term. But inspired by [198], maximizing this term can be considered as generating valid graph counterfactuals w.r.t. the desired label Y^*, thus we replace this term with $-\mathbb{E}_Q[d(G, G^{CF}) + \alpha \cdot l(f(G^{CF}), Y^*)]$, where $d(\cdot, \cdot)$ is a similarity loss, which is a distance metric to measure the difference between G and G^{CF}, $l(\cdot)$ is the counterfactual prediction loss to measure the difference between the predicted label $f(G^{CF})$ and the desired label Y^*. In summary, these two terms encourage the model to output counterfactuals that are similar to the input graph but elicit the desired predicted label. α is a hyperparameter to control the weight of the counterfactual prediction loss. Overall, the loss function of CLEAR-VAE is:

$$\mathcal{L} = \mathbb{E}_Q[d(G, G^{CF}) + \alpha \cdot l(f(G^{CF}), Y^*)] + \text{KL}(Q(Z|G, Y^*)\|P(Z|G, Y^*)). \quad (9.2)$$

Encoder. In the encoder, the input includes node features X and graph structure A of the explainee graph G, as well as the desired label Y^*, the output is latent representation Z. The encoder learns the distribution $Q(Z|G, Y^*)$. We use a Gaussian distribution $P(Z|G, Y^*) = \mathcal{N}(\mu_z(Y^*), \text{diag}(\sigma_z^2(Y^*)))$ as prior, and enforce the learned distribution $Q(Z|G, Y^*)$ to be close to the prior by minimizing their KL divergence. Here, $\mu_z(Y^*)$ and $\text{diag}(\sigma_z^2(Y^*))$ are mean and diagonal covariance of the prior distribution learned by a neural network module. Z is sampled from the learned distribution $Q(Z|G, Y^*)$ with the widely-used reparameterization trick [208].

Decoder. In the decoder, the input includes Z and Y^*, while the output is the counterfactual $G^{CF} = (X^{CF}, A^{CF})$. Different counterfactuals can be generated for one explainee graph by sampling Z from $Q(Z|G, Y^*)$ for multiple times. The adjacency matrix is often discrete, and

typically assumed to include only binary values ($A_{(i,j)} = 1$ if edge from node i to node j exists, otherwise $A_{(i,j)} = 0$). To facilitate optimization, inspired by recent graph generative models [25, 207], our decoder outputs a probabilistic adjacency matrix $\hat{A}^{CF}$ with elements in range $[0, 1]$, and then generates a binary adjacency matrix A^{CF} by sampling from Bernoulli distribution with probabilities in $\hat{A}^{CF}$. We calculate the similarity loss in Eq. (9.2) as:

$$d(G, G^{CF}) = d_A(A, \hat{A}^{CF}) + \beta \cdot d_X(X, X^{CF}), \tag{9.3}$$

where d_A and d_X are metrics to measure the distance between two graphs w.r.t. their graph structures and node features, respectively. β controls the weight for the similarity loss w.r.t. node features. More details of model implementation are in Appendix D.

9.2.2 CLEAR: Improving the Causality in Counterfactual Explanations

To further incorporate the causality in the generated CFEs, most existing studies [198, 210, 211] leverage certain prior knowledge (e.g., a given path diagram which depicts the causal relations among variables) of the SCM. However, it is often difficult to obtain sufficient prior knowledge of the SCM in real-world data, especially for graph data. In this work, we do not assume the access to the prior knowledge of SCM, and only assume that the observational data is available. However, the key challenge, as shown in [210], is that it is impossible to *identify* the ground-truth SCM from observational data without additional assumptions w.r.t. the structural equations and the exogenous variables, because different SCMs may result in the same observed data distribution. Considering that different SCMs can generate different counterfactuals, the identifiability of SCM is an obstacle of promoting causality in CFE. Fortunately, enlightened by recent progress in nonlinear independent component analysis (ICA) [206, 212], we make an initial exploration to promote causality in CFE by improving the identifiability of the latent variables in our CFE generator with the help of an auxiliary observed variable. This CFE generator is denoted by CLEAR.

In nonlinear independent component analysis (ICA) [206, 212, 213, 214], it is assumed that the observed data, e.g., X, is generated from a smooth and invertible nonlinear transformation of independent latent variables (referred to as *sources*) Z. Identifying the sources and the transformation are the key goals in nonlinear ICA. Similarly, traditional VAE models also assume that the observed features X are generated by a set of latent variables Z. However, traditional VAEs cannot be directly used for nonlinear ICA as they lack identifiability, i.e., we can find different Z that lead to the same observed data distribution $p(X)$. Recent studies [206] have shown that the identifiability of VAE models can be improved with an auxiliary observed variable S (e.g., a time index or class label), which enables us to use VAE for nonlinear ICA problem. As discussed in [213, 214], a SCM can be considered as a nonlinear

ICA model if the exogenous variables in the SCM are considered as the sources in nonlinear ICA. Similar connections can be built between the structural equations in SCM and the transformations in ICA. Such connections shed light on improving the identifiability of the underlying SCM without much explicit prior knowledge of the SCM.

With this idea, based on the backbone CLEAR-VAE, improves the causality in counterfactuals by promoting identifiability with an observed auxiliary variable S. Intuitively, we expect the graph VAE can capture the exogenous variables of the SCM in its representations Z, and approximate the data generation process from the exogenous variables to the observed data, which is consistent with the SCM. Here, for each graph, the auxiliary variable S can provide additional information for CLEAR to better identify the exogenous variables in the SCM, and thus can elicit counterfactuals with better causality. To achieve this goal, following the previous work of nonlinear ICA [206, 212], we make the following assumption:

ASSUMPTION 5. *We assume that the prior on the latent variables $P(Z|S)$ is conditionally factorial.*

With this assumption, the original data can be stratified by different values of S, and each separated data stratum can be considered to be generated by the ground-truth SCM under certain constraints (e.g., the range of values that the exogenous variables can take). When the constraints become more restricted, the space of possible SCMs that can generate the same data distribution in each data stratum shrinks. In this way, the identification of the ground-truth SCM can be easier if we leverage the auxiliary variable S. Here, with the auxiliary variable S, we infer the ELBO of CLEAR:

THEOREM 2. *The evidence lower bound (ELBO) to optimize the framework CLEAR is:*

$$\ln P(G^{CF}|S,Y^*,G) \geq \mathbb{E}_Q[\ln P(G^{CF}|Z,S,Y^*,G)] - \mathit{KL}(Q(Z|G,S,Y^*)\|P(Z|G,S,Y^*)). \tag{9.4}$$

The detailed proof is shown in Appendix D.

Loss function of CLEAR. Based on the above ELBO, the final loss function of CLEAR is:

$$\mathcal{L} = \mathbb{E}_Q[d(G,G^{CF}) + \alpha \cdot l(f(G^{CF}),Y^*)] + \text{KL}(Q(Z|G,S,Y^*)\|P(Z|G,S,Y^*)). \tag{9.5}$$

Encoder and Decoder. The encoder takes the input G, S, and Y^*, and outputs Z as the latent representation. We use a Gaussian prior $P(Z|G,S,Y^*) = \mathcal{N}(\mu_z(S,Y^*), \text{diag}(\sigma_Z^2(S,Y^*))$ with its mean and diagonal covariance learned by neural network, and we encourage the learned approximate posterior $Q(Z|G,S,Y^*)$ to approach the prior by minimizing their KL divergence. Similar to the backbone CLEAR-VAE, the decoder takes the inputs Z and Y^* to

generate one or multiple counterfactuals G^{CF} for each explainee graph. More implementation details are in Appendix D.

9.3 Experimental Evaluation

In our experiments, we will answer the following research questions: **RQ1:** How does CLEAR perform compared to state-of-the-art baselines? **RQ2:** How do different components in CLEAR contribute to the performance? **RQ3:** How can the generated CFEs promote model explainability? **RQ4:** How does CLEAR perform under different settings of hyperparameters?

Baselines. We use the following baselines for comparison: 1) **Random**: For each explainee graph, it randomly perturbs the graph structure for at most T steps.2) **EG-IST**: For each explainee graph, it randomly inserts edges into it for at most T steps. 3) **EG-RM**: For each explainee graph, it randomly removes edges for at most T steps. 4) **GNNExplainer**: GNNExplainer [24] is proposed to identify the most important subgraphs for prediction. We apply it for CFE generation by removing the important subgraphs identified by GNNExplainer. 5) **CF-GNNExplainer**: CF-GNNExplainer [25] is proposed for generating counterfactual ego networks in node classification tasks. We adapt CF-GNNExplainer for graph classification. 6) **MEG**: MEG [200] is a reinforcement learning based CFE generation method.

Evaluation Metrics. We use the following metrics in evaluation:

1) **Validity:** the proportion of counterfactuals which obtain the desired labels.

$$\text{Validity} = \frac{1}{N}\sum\nolimits_{i\in[N]} \frac{1}{N^{CF}} \sum\nolimits_{j\in[N^{CF}]} |\mathbf{1}(f(\mathbf{G}^{CF}_{(i,j)}) = y^*_i)|, \tag{9.6}$$

where N is the number of graph instances, N^{CF} is the number of counterfactuals generated for each graph. $\mathbf{G}^{CF}_{(i,j)} = (\mathbf{X}^{CF}_{(i,j)}, \mathbf{A}^{CF}_{(i,j)})$ denotes the j-th counterfactual generated for the i-th graph instance. Here, y^*_i is the realization of Y^* for the i-th graph. $\mathbf{1}(\cdot)$ is an indicator function which outputs 1 when the input condition is true, otherwise it outputs 0.

2) **Proximity:** the similarity between the generated counterfactuals and the input graph.

$$\text{Proximity}_X = \frac{1}{N}\sum_{i\in[N]} \frac{1}{N^{CF}} \sum_{j\in[N^{CF}]} \text{sim}_X(\mathbf{X}_{(i)}, \mathbf{X}^{CF}_{(i,j)}),\ \text{Proximity}_A = \frac{1}{N}\sum_{i\in[N]} \frac{1}{N^{CF}} \sum_{j\in[N^{CF}]} \text{sim}_A(\mathbf{A}_{(i)}, \mathbf{A}^{CF}_{(i,j)}), \tag{9.7}$$

where we use cosine similarity for $\text{sim}_X(\cdot)$, and accuracy for $\text{sim}_A(\cdot)$.

3) **Causality:** Similarly as [198], we measure the causality by reporting the ratio of counterfactuals which satisfy the causal constraints corresponding to R.

4) **Time:** the average time cost (seconds) of generating a counterfactual for a single graph.

9.3.1 Datasets and Simulation

We evaluate our method on three datasets, including a synthetic dataset and two datasets with real-world graphs. **(1) Community.** This dataset contains synthetic graphs generated by the Erdös-Rényi (E-R) model [215]. In this dataset, each graph consists of two 10-node communities. The label Y is determined by the average node degree in the first community (denoted by $\deg_1(A)$). According to the causal model (in Appendix D), when $\deg_1(A)$ increases (decreases), the average node degree in the second community $\deg_2(A)$ should decrease (increase) correspondingly. We take this causal relation $\deg_1(A) \rightarrow \deg_2(A)$ as our causal relation of interest, and denote it as R. Correspondingly, we define a causal constraint for later evaluation of causality: "$(\deg_1(A^{CF}) > \deg_1(A)) \Rightarrow (\deg_2(A^{CF}) < \deg_2(A))$" OR "$(\deg_1(A^{CF}) < \deg_1(A)) \Rightarrow (\deg_2(A^{CF}) > \deg_2(A))$". **(2) Ogbg-molhiv.** In this dataset, each graph stands for a molecule, where each node represents an atom, and each edge is a chemical bond. As the ground-truth causal model is unavailable, we simulate the label Y and causal relation of interest R. **(3) IMDB-M.** This dataset contains movie collaboration networks from IMDB. In each graph, each node represents an actor or an actress, and each edge represents the collaboration between two actors or actresses in the same movie. Similarly as the above datasets, we simulate the label Y and causal relation of interest R, and define causal constraints corresponding to R. It is worth mentioning that the causal relation of interest R in the three datasets covers different types of causal relations respectively: i) causal relations between variables in graph structure A; ii) between variables in node features X; iii) between variables in A and in X. Thereby we comprehensively evaluate the performance of CLEAR in leveraging different modalities (node features and graph structure) of graphs to fit in different types of causal relations. More details about datasets are in Appendix D.

9.3.2 Performance of Different Methods

To evaluate our framework CLEAR, we compare its CFE generation performance against the state-of-the-art baselines. From the results in Table 9.1, we summarize the main observations as follows:

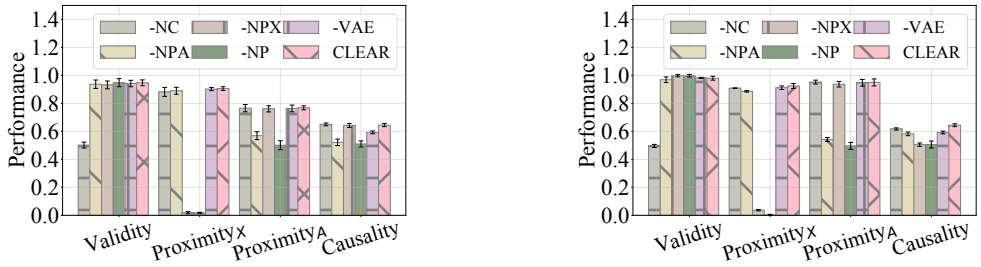
- **Validity and proximity.** Our framework CLEAR achieves good performance in validity and proximity. This observation validates the effectiveness of our method in achieving the basic target of CFE generation. a) In validity, CLEAR obviously outperforms all baselines on most datasets. Random, EG-IST, and EG-RM perform the worst due to their random nature; GNNExplainer can only remove edges and nodes, which also limits its validity; CF-GNNExplainer and MEG perform well as their optimization is designed for CFE generation; b) In Proximity$_A$, CLEAR outperforms

TABLE 9.1. The performance (mean $\pm$ standard deviation over ten repeated executions) of different methods of CFEs on graphs. The best results are in bold, and the runner-up results are underlined.

Datasets	Methods	Validity (↑)	Proximity$_X$(↑)	Proximity$_A$ (↑)	Causality (↑)	Time (↓)
Community	Random	0.53 ± 0.05	N/A	$\underline{0.77 \pm 0.02}$	$\underline{0.52 \pm 0.06}$	0.20 ± 0.01
	EG-IST	0.53 ± 0.05	N/A	0.66 ± 0.03	0.13 ± 0.06	0.27 ± 0.03
	EG-RMV	0.55 ± 0.04	N/A	$\mathbf{0.85 \pm 0.01}$	0.03 ± 0.02	$\underline{0.15 \pm 0.01}$
	GNNExplainer	0.52 ± 0.06	N/A	0.71 ± 0.01	0.05 ± 0.00	2.87 ± 0.08
	CF-GNNExplainer	$\underline{0.90 \pm 0.04}$	N/A	0.72 ± 0.00	0.14 ± 0.02	25.14 ± 1.22
	MEG	0.88 ± 0.04	N/A	0.71 ± 0.01	0.10 ± 0.03	27.29 ± 1.32
	CLEAR(ours)	$\mathbf{0.94 \pm 0.02}$	$\mathbf{0.91 \pm 0.01}$	0.77 ± 0.00	$\mathbf{0.65 \pm 0.03}$	$\mathbf{0.01 \pm 0.01}$
Ogbg-molhiv	Random	0.48 ± 0.09	N/A	0.87 ± 0.02	0.46 ± 0.1	0.17 ± 0.02
	EG-IST	0.48 ± 0.09	N/A	0.83 ± 0.03	0.46 ± 0.09	0.19 ± 0.04
	EG-RM	0.483 ± 0.09	N/A	$\mathbf{0.96 \pm 0.01}$	0.47 ± 0.09	$\underline{0.17 \pm 0.04}$
	GNNExplainer	0.50 ± 0.01	N/A	0.92 ± 0.00	0.48 ± 0.10	2.78 ± 0.10
	CF-GNNExplainer	$\underline{0.54 \pm 0.02}$	N/A	0.92 ± 0.01	0.49 ± 0.02	27.93 ± 1.20
	MEG	0.49 ± 0.03	N/A	0.93 ± 0.01	$\underline{0.50 \pm 0.10}$	22.39 ± 2.20
	CLEAR(ours)	$\mathbf{0.98 \pm 0.01}$	$\mathbf{0.92 \pm 0.02}$	$\underline{0.95 \pm 0.01}$	$\mathbf{0.64 \pm 0.02}$	$\mathbf{0.01 \pm 0.00}$
IMDB-M	Random	0.50 ± 0.04	N/A	0.67 ± 0.01	0.43 ± 0.08	0.19 ± 0.01
	EG-IST	0.56 ± 0.12	N/A	0.67 ± 0.06	0.45 ± 0.07	$\underline{0.16 \pm 0.03}$
	EG-RM	0.45 ± 0.11	N/A	$\mathbf{0.75 \pm 0.03}$	$\underline{0.53 \pm 0.08}$	0.18 ± 0.02
	GNNExplainer	0.43 ± 0.10	N/A	0.62 ± 0.02	0.50 ± 0.02	2.46 ± 0.50
	CF-GNNExplainer	$\underline{0.95 \pm 0.02}$	N/A	0.74 ± 0.02	0.51 ± 0.02	22.21 ± 1.42
	MEG	0.90 ± 0.02	N/A	0.72 ± 0.02	0.51 ± 0.02	24.12 ± 1.08
	CLEAR(ours)	$\mathbf{0.96 \pm 0.01}$	$\mathbf{0.99 \pm 0.00}$	$\underline{0.75 \pm 0.01}$	$\mathbf{0.73 \pm 0.01}$	$\mathbf{0.01 \pm 0.00}$

all non-random baselines. EG-RM performs the best in Proximity$_A$ because most graphs are very sparse, thus only removing edges can change the graph relatively less than other methods. As the baselines either cannot perturb node features, or their perturbation approach on node features cannot fit well in our setting, we do not compare Proximity$_X$ with them.

- **Time.** CLEAR significantly outperforms all baselines in time efficiency. Most of the baselines generate CFEs in an iterative way, and MEG needs to enumerate all perturbations at each step. GNNExplainer and CF-GNNExplainer optimize on every single instance, which limits their generalization. All the above reasons erode their time efficiency. While in our framework, the generative mechanism enables efficient CFE generation and generalization on unseen graphs, thus brings substantial improvement in time efficiency.
- **Causality.** CLEAR dramatically outperforms all baselines in causality. We contribute the superiority of our framework w.r.t. causality in two key factors: a) different from some baselines (e.g., GNNExplainer) optimized on each single graph, our framework can better capture the causal relations among different variables in data by leveraging the data distribution of the training set; b) our framework utilizes



(A) Community (B) Ogbg-molhiv

FIGURE 9.4. Ablation studies for CLEAR.

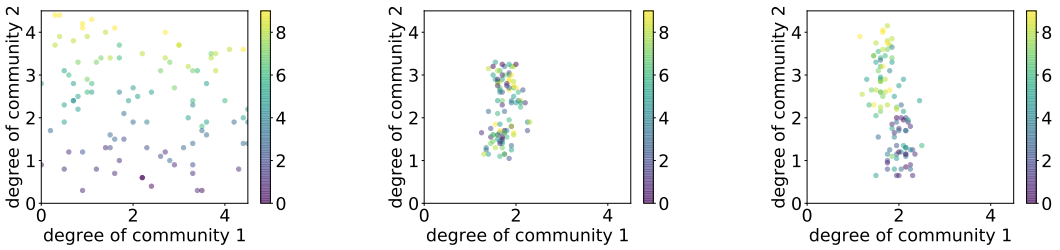
the auxiliary variable to better identify the underlying causal model and promote causality.

9.3.3 Ablation Study

To evaluate the effectiveness of different components in CLEAR, we conduct an ablation study with the following variants: 1) **CLEAR-NC**. In this variant, we remove the counterfactual prediction loss; 2) **CLEAR-NPA**, we remove the similarity loss w.r.t. graph structure; 3) **CLEAR-NPX**, we remove the similarity loss w.r.t. node features; 4) **CLEAR-NP**, we remove all the similarity loss; 5) **CLEAR-VAE**, the backbone of our framework. As shown in Fig. 9.4, we have the following observations: 1) The validity of CLEAR-NC degrades dramatically due to the lack of counterfactual prediction loss; 2) The performance w.r.t. proximity is worse in CLEAR-NPA, CLEAR-NPX, and CLEAR-NP as the similarity loss is removed. Besides, removing the similarity loss can also hurt the performance of causality when the variables in the causal relation of interest R are involved. For example, in Community, CLEAR-NPA performs much worse in causality (as R in Community involves node degree in graph structure), while in Ogbg-molhiv, the performance in causality of CLEAR-NPX is eroded (as R on Ogbg-molhiv involves node features); 3) The performance w.r.t. causality is impeded in CLEAR-VAE. This observation validates the effectiveness of the auxiliary variable for promoting causality. Similar observations can also be found in the ablation study on the IMDB-M dataset, which is shown in Appendix D.

9.3.4 Explainability through CFEs

To investigate how CFE on graphs promote model explainability, we take a closer look in the generated counterfactuals. Due to the space limit, we only show our investigation on the Community dataset. Fig. 9.5(a) shows the distribution of two variables: the average node degree in the first community and in the second community in the original dataset, i.e., $\deg_1(A)$



(A) Original data (B) CFEs from CLEAR-VAE (C) CFEs from CLEAR

FIGURE 9.5. Explainability through CFEs on Community dataset.

and $\deg_2(A)$. Fig. 9.5(b) shows the distribution of these two variables in counterfactuals generated by CLEAR-VAE. We observe that these counterfactuals are distributed close to the decision boundary, i.e., $\deg_1(A) = \text{ADG}_1$, where ADG_1 is a constant around 2. This is because the counterfactuals are enforced to change their predicted labels with perturbation as slight as possible. Fig. 9.5(c) shows the distribution of these two variables $\deg_1(A)$ and $\deg_2(A)$ in counterfactuals generated by CLEAR. Different colors denote different values of the auxiliary variable S. Notice that based on the causal model (in Appendix D), the exogenous variables are distributed in a narrow range when the value of S is fixed, thus the same color also indicates similar values of exogenous variables. We observe that compared with the color distribution in Fig. 9.5(b), the color distribution in Fig. 9.5(c) is more consistent with Fig. 9.5(a). This indicates that compared with CLEAR-VAE, CLEAR can better capture the values of exogenous variables, and thus the counterfactuals generated by CLEAR are more consistent with the underlying causal model. To better illustrate the explainability provided by CFE, we further conduct case studies to compare the original graphs and their counterfactuals in Appendix D.

Part IV

Summary and Future Work

CHAPTER 10

Summary and Future Work

In this dissertation, we investigate the interplay between causal inference and graph machine learning, mainly including two directions: leveraging graph ML to facilitate causal inference, and leveraging causal inference for trustworthy graph ML. Based on the prior work in these directions, we focus on different research topics: 1) causal inference on dynamic graphs with hidden confounders; 2) causal inference on hypergraph with interference; 3) causal inference under entangled treatments; 4) counterfactual fairness for node representation learning, and 5) counterfactual explanation for graph ML models. We introduce the motivation, problem definition, proposed method, and evaluation results for each topic. These studies have made contributions toward improving the fields of causal inference and graph learning, thereby enhancing their potential for real-world applications.

In the future, we can further explore these areas and extend to a wider range of research. Beyond my current study on graph ML and causal inference [21, 22, 216], a bigger picture behind is causality-based graph mining, which aims to identify the underlying causal relations buried in various types of graph data, and use them to facilitate future graph mining. Besides, most of my studies focus on a subarea of graph ML and causal effect estimation [21, 22]. In a higher level, there remain various interesting research topics in bridging the gap between causal inference and machine learning, including multiple types of data (e.g., natural language, video, and images) from different sources. The interdisciplinary study of causal inference and machine learning has been widely considered as a major milestone in AI, as it allows AI models to make accurate predictions based on causal relations rather than just correlations. In general, the combination of causal inference and machine learning sheds light on capturing the essential foundation of human cognition and artificial intelligence. I believe it would continuously provide guidance to next-generation AI, covering various applications (e.g., bio-medicine, recommender system, epidemiological study, economic analysis, and human-involved AI). Continuous progress in these areas can make a vital contribution to building trustworthy machine learning algorithms. Applying them to different tasks can have a significant positive impact on future human life in reality.

References

[1] Feng Xie and David Levinson. 'Measuring the structure of road networks'. In: *Geographical analysis* 39.3 (2007), pp. 336–356.

[2] J Clyde Mitchell. 'Social networks'. In: *Annual review of anthropology* 3.1 (1974), pp. 279–299.

[3] Maurice A Hitchcock et al. 'Professional networks: the influence of colleagues on the academic success of faculty.' In: *Academic Medicine: Journal of the Association of American Medical Colleges* 70.12 (1995), pp. 1108–1116.

[4] Frank Schweitzer et al. 'Economic networks: The new challenges'. In: *science* 325.5939 (2009), pp. 422–425.

[5] Wolfgang Huber et al. 'Graphs in molecular biology'. In: *BMC bioinformatics* 8.6 (2007), pp. 1–14.

[6] Yu Rong et al. 'Dropedge: Towards deep graph convolutional networks on node classification'. In: *arXiv preprint* (2019).

[7] Linyuan Lü and Tao Zhou. 'Link prediction in complex networks: A survey'. In: *Physica A: statistical mechanics and its applications* 390.6 (2011), pp. 1150–1170.

[8] Muhan Zhang et al. 'An end-to-end deep learning architecture for graph classification'. In: *Proceedings of the AAAI conference on artificial intelligence*. Vol. 32. 1. 2018.

[9] Santo Fortunato. 'Community detection in graphs'. In: *Physics reports* 486.3-5 (2010), pp. 75–174.

[10] Satu Elisa Schaeffer. 'Graph clustering'. In: *Computer science review* 1.1 (2007), pp. 27–64.

[11] Thomas N Kipf and Max Welling. 'Semi-supervised classification with graph convolutional networks'. In: *International Conference on Learning Representations*. 2017.

[12] Thomas N Kipf and Max Welling. 'Variational graph auto-encoders'. In: *arXiv preprint* (2016).

[13] Petar Veličković et al. 'Graph attention networks'. In: *ICLR* (2018).

[14] Seongjun Yun et al. 'Graph transformer networks'. In: *Advances in neural information processing systems* 32 (2019).

[15] Judea Pearl. *Causality*. Cambridge university press, 2009.

[16] Guido W Imbens and Donald B Rubin. *Causal inference in statistics, social, and biomedical sciences*. 2015.

[17] Harald O Stolberg, Geoffrey Norman and Isabelle Trop. 'Randomized controlled trials'. In: *AJR Am J Roentgenol* 183.6 (2004), pp. 1539–44.

[18] Cory E Goldstein et al. 'Ethical issues in pragmatic randomized controlled trials: a review of the recent literature identifies gaps in ethical argumentation'. In: *BMC Medical Ethics* (2018).

[19] Donald B Rubin. 'Causal inference using potential outcomes: Design, modeling, decisions'. In: *Journal of the American Statistical Association* 100.469 (2005), pp. 322–331.

[20] Donald B Rubin. 'Bayesian inference for causal effects'. In: *Handbook of Statistics* 25 (2005).

[21] Jing Ma et al. 'Deconfounding with Networked Observational Data in a Dynamic Environment'. In: *ACM International Conference on Web Search and Data Mining*. 2021.

[22] Jing Ma et al. 'Learning Causal Effects on Hypergraphs'. In: *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 2022.

[23] Yushun Dong et al. 'Fairness in Graph Mining: A Survey'. In: *arXiv preprint* (2022).

[24] Rex Ying et al. 'Gnnexplainer: Generating explanations for graph neural networks'. In: *NeurIPS* (2019).

[25] Ana Lucic et al. 'CF-GNNExplainer: Counterfactual Explanations for Graph Neural Networks'. In: *arXiv preprint* (2021).

[26] Peter Spirtes et al. *Causation, prediction, and search*. MIT press, 2000.

[27] Pater Spirtes et al. 'Constructing Bayesian network models of gene expression networks from microarray data'. In: (2000).

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

[29] Xun Zheng et al. 'Dags with no tears: Continuous optimization for structure learning'. In: *Advances in neural information processing systems* 31 (2018).

[30] Peter C Austin. 'An introduction to propensity score methods for reducing the effects of confounding in observational studies'. In: *Multivariate Behavioral Research* (2011).

[31] Xing Sam Gu and Paul R Rosenbaum. 'Comparison of multivariate matching methods: Structures, distances, and algorithms'. In: *Journal of Computational and Graphical Statistics* 2.4 (1993), pp. 405–420.

[32] Keisuke Hirano, Guido W Imbens and Geert Ridder. 'Efficient estimation of average treatment effects using the estimated propensity score'. In: *Econometrica* 71.4 (2003), pp. 1161–1189.

[33] Kosuke Imai and Marc Ratkovic. ‘Covariate balancing propensity score’. In: *Journal of the Royal Statistical Society: Series B: Statistical Methodology* (2014), pp. 243–263.

[34] Joshua D Angrist, Guido W Imbens and Donald B Rubin. ‘Identification of causal effects using instrumental variables’. In: *Journal of the American statistical Association* (1996).

[35] Jason Hartford et al. ‘Deep IV: A flexible approach for counterfactual prediction’. In: *International Conference on Machine Learning*. PMLR. 2017, pp. 1414–1423.

[36] Andrew Goodman-Bacon. ‘Difference-in-differences with variation in treatment timing’. In: *Journal of Econometrics* 225.2 (2021), pp. 254–277.

[37] Uri Shalit, Fredrik D Johansson and David Sontag. ‘Estimating individual treatment effect: generalization bounds and algorithms’. In: *International Conference on Machine Learning*. 2017.

[38] Christos Louizos et al. ‘Causal effect inference with deep latent-variable models’. In: *Advances in Neural Information Processing Systems*. 2017.

[39] Ishita Dasgupta et al. ‘Causal reasoning from meta-reinforcement learning’. In: *arXiv preprint* (2019).

[40] Samuel J Gershman. ‘Reinforcement learning and causal models’. In: *The Oxford handbook of causal reasoning* 1 (2017), p. 295.

[41] Ruocheng Guo, Jundong Li and Huan Liu. ‘Learning individual causal effects from networked observational data’. In: *International Conference on Web Search and Data Mining*. 2020.

[42] Ruocheng Guo et al. ‘IGNITE: A Minimax Game Toward Learning Individual Treatment Effects from Networked Observational Data’. In: *International Joint Conference on Artificial Intelligence*. 2020.

[43] Zhixuan Chu, Stephen L Rathbun and Sheng Li. ‘Graph Infomax Adversarial Learning for Treatment Effect Estimation with Networked Observational Data’. In: *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 2021.

[44] Peter M Aronow and Cyrus Samii. ‘Estimating average causal effects under general interference, with application to a social network experiment’. In: *The Annals of Applied Statistics* (2017).

[45] Guillaume Basse and Avi Feller. ‘Analyzing two-stage experiments in the presence of interference’. In: *Journal of the American Statistical Association* (2018).

[46] Kosuke Imai, Zhichao Jiang and Anup Malani. ‘Causal inference with interference and noncompliance in two-stage randomized experiments’. In: *Journal of the American Statistical Association* (2020).

[47] Yunpu Ma and Volker Tresp. 'Causal Inference under Networked Interference and Intervention Policy Enhancement'. In: *International Conference on Artificial Intelligence and Statistics*. 2021.

[48] Vineeth Rakesh et al. 'Linked causal variational autoencoder for inferring paired spillover effects'. In: *International Conference on Information and Knowledge Management*. 2018.

[49] Shonosuke Harada and Hisashi Kashima. 'Graphite: Estimating individual effects of graph-structured treatments'. In: *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*. 2021, pp. 659–668.

[50] Jean Kaddour et al. 'Causal effect inference for structured treatments'. In: *Advances in Neural Information Processing Systems* 34 (2021), pp. 24841–24854.

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

[52] Peter Spirtes and Kun Zhang. 'Causal discovery and inference: concepts and recent methodological advances'. In: *Applied informatics*. Vol. 3. 1. SpringerOpen. 2016, pp. 1–28.

[53] Yunzhu Li et al. 'Causal discovery in physical systems from videos'. In: *Advances in Neural Information Processing Systems* 33 (2020), pp. 9180–9192.

[54] Sindy Löwe et al. 'Amortized causal discovery: Learning to infer causal graphs from time-series data'. In: *Conference on Causal Learning and Reasoning*. PMLR. 2022, pp. 509–525.

[55] Dongjie Wang et al. 'Hierarchical Graph Neural Networks for Causal Discovery and Root Cause Localization'. In: *arXiv preprint* (2023).

[56] Yue Yu et al. 'Dag-gnn: Dag structure learning with graph neural networks'. In: *International Conference on Machine Learning*. 2019.

[57] Haoyang Li et al. 'Out-of-distribution generalization on graphs: A survey'. In: *arXiv preprint* (2022).

[58] Tong Zhao et al. 'Data augmentation for graph neural networks'. In: *Proceedings of the aaai conference on artificial intelligence*. Vol. 35. 12. 2021, pp. 11015–11023.

[59] Yuning You et al. 'Graph contrastive learning with augmentations'. In: *Advances in neural information processing systems* 33 (2020), pp. 5812–5823.

[60] Haoyang Li et al. 'Learning invariant graph representations for out-of-distribution generalization'. In: *Advances in Neural Information Processing Systems*. 2022.

[61] Ying-Xin Wu et al. 'Discovering invariant rationales for graph neural networks'. In: *arXiv preprint* (2022).

[62] Jianxin Ma et al. 'Disentangled graph convolutional networks'. In: *International conference on machine learning*. PMLR. 2019, pp. 4212–4221.

[63] Yanbei Liu et al. ‘Independence promoted graph disentangled networks’. In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 34. 04. 2020, pp. 4916–4923.

[64] Man Wu et al. ‘Domain-adversarial graph neural networks for text classification’. In: *2019 IEEE International Conference on Data Mining (ICDM)*. IEEE. 2019, pp. 648–657.

[65] Fuli Feng et al. ‘Graph adversarial training: Dynamically regularizing based on graph structure’. In: *IEEE Transactions on Knowledge and Data Engineering* 33.6 (2019), pp. 2493–2504.

[66] Weihua Hu et al. ‘Strategies for pre-training graph neural networks’. In: *arXiv preprint* (2019).

[67] Hongrui Liu et al. ‘Confidence may cheat: Self-training on graph neural networks under distribution shift’. In: *Proceedings of the ACM Web Conference 2022*. 2022, pp. 1248–1258.

[68] Shaohua Fan et al. ‘Generalizing graph neural networks on out-of-distribution graphs’. In: *arXiv preprint* (2021).

[69] Yongduo Sui et al. ‘Causal attention for interpretable and generalizable graph classification’. In: *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. 2022, pp. 1696–1705.

[70] Beatrice Bevilacqua, Yangze Zhou and Bruno Ribeiro. ‘Size-invariant graph representations for graph classification extrapolations’. In: *International Conference on Machine Learning*. PMLR. 2021, pp. 837–851.

[71] Yushun Dong et al. ‘Fairness in graph mining: A survey’. In: *IEEE Transactions on Knowledge and Data Engineering* (2023).

[72] Avishek Bose and William Hamilton. ‘Compositional fairness constraints for graph embeddings’. In: *International Conference on Machine Learning*. PMLR. 2019, pp. 715–724.

[73] Xu Zhang et al. ‘A Multi-view Confidence-calibrated Framework for Fair and Stable Graph Representation Learning’. In: *2021 IEEE International Conference on Data Mining (ICDM)*. IEEE. 2021, pp. 1493–1498.

[74] Maarten Buyl and Tijl De Bie. ‘Debayes: a bayesian method for debiasing network embeddings’. In: *International Conference on Machine Learning*. PMLR. 2020, pp. 1220–1229.

[75] Yushun Dong et al. ‘Individual fairness for graph neural networks: A ranking based approach’. In: *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*. 2021, pp. 300–310.

[76] Wei Fan et al. ‘Fair graph auto-encoder for unbiased graph representations with wasserstein distance’. In: *2021 IEEE International Conference on Data Mining (ICDM)*. IEEE. 2021, pp. 1054–1059.

[77] Preethi Lahoti, Krishna P Gummadi and Gerhard Weikum. ‘Operationalizing individual fairness with pairwise fair representations’. In: *arXiv preprint* (2019).

[78] Xianfeng Tang et al. ‘Investigating and mitigating degree-related biases in graph convoltuional networks’. In: *Proceedings of the 29th ACM International Conference on Information & Knowledge Management*. 2020, pp. 1435–1444.

[79] Jian Kang et al. ‘Rawlsgcn: Towards rawlsian difference principle on graph convolutional network’. In: *Proceedings of the ACM Web Conference 2022*. 2022, pp. 1214–1225.

[80] Himan Abdollahpouri, Robin Burke and Bamshad Mobasher. ‘Controlling popularity bias in learning-to-rank recommendation’. In: *Proceedings of the eleventh ACM conference on recommender systems*. 2017, pp. 42–46.

[81] Toshihiro Kamishima et al. ‘Efficiency Improvement of Neutrality-Enhanced Recommendation.’ In: *Decisions@ RecSys*. Citeseer. 2013, pp. 1–8.

[82] Mengting Wan et al. ‘Addressing marketing bias in product recommendations’. In: *Proceedings of the 13th international conference on web search and data mining*. 2020, pp. 618–626.

[83] Hao Yuan et al. ‘Explainability in graph neural networks: A taxonomic survey’. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2022).

[84] Hao Yuan et al. ‘Xgnn: Towards model-level explanations of graph neural networks’. In: *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 2020, pp. 430–438.

[85] Federico Baldassarre and Hossein Azizpour. ‘Explainability techniques for graph convolutional networks’. In: *arXiv preprint* (2019).

[86] Phillip E Pope et al. ‘Explainability methods for graph convolutional neural networks’. In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2019, pp. 10772–10781.

[87] Dongsheng Luo et al. ‘Parameterized explainer for graph neural network’. In: *Advances in neural information processing systems* 33 (2020), pp. 19620–19631.

[88] Thorben Funke, Megha Khosla and Avishek Anand. ‘Hard masking for explaining graph neural networks’. In: (2021).

[89] Xiang Wang et al. ‘Causal screening to interpret graph neural networks’. In: (2021).

[90] Robert Schwarzenberg et al. ‘Layerwise relevance visualization in convolutional text graph classifiers’. In: *arXiv preprint* (2019).

[91] Thomas Schnake et al. 'Higher-order explanations of graph neural networks via relevant walks'. In: *IEEE transactions on pattern analysis and machine intelligence* 44.11 (2021), pp. 7581–7596.

[92] Qiang Huang et al. 'Graphlime: Local interpretable model explanations for graph neural networks'. In: *IEEE Transactions on Knowledge and Data Engineering* (2022).

[93] Minh Vu and My T Thai. 'Pgm-explainer: Probabilistic graphical model explanations for graph neural networks'. In: *Advances in neural information processing systems* 33 (2020), pp. 12225–12235.

[94] Yongduo Sui et al. 'Deconfounded training for graph neural networks'. In: *arXiv preprint* (2021).

[95] Tao Zhang, Hao-Ran Shan and Max A Little. 'Causal GraphSAGE: A robust graph method for classification based on causal sampling'. In: *Pattern Recognition* 128 (2022), p. 108696.

[96] Wanyu Lin, Hao Lan and Baochun Li. 'Generative causal explanations for graph neural networks'. In: *International Conference on Machine Learning*. PMLR. 2021, pp. 6666–6679.

[97] Jing Ma et al. 'CLEAR: Generative Counterfactual Explanations on Graphs'. In: *Neural Information Processing Systems*. 2022.

[98] Chaveevan Pechsiri and Rapepun Piriyakul. 'Explanation knowledge graph construction through causality extraction from texts'. In: *Journal of computer science and technology* 25.5 (2010), pp. 1055–1070.

[99] Xiang Wang et al. 'Reinforced causal explainer for graph neural networks'. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2022).

[100] Chirag Agarwal, Himabindu Lakkaraju and Marinka Zitnik. 'Towards a unified framework for fair and stable graph representation learning'. In: *Uncertainty in Artificial Intelligence*. 2021, pp. 2114–2124.

[101] Jing Ma et al. 'Learning fair node representations with graph counterfactual fairness'. In: *Proceedings of the Fifteenth ACM International Conference on Web Search and Data Mining*. 2022.

[102] Jennifer L Hill. 'Bayesian nonparametric modeling for causal inference'. In: *Journal of Computational and Graphical Statistics* 20.1 (2011).

[103] Stefan Wager and Susan Athey. 'Estimation and inference of heterogeneous treatment effects using random forests'. In: *Journal of the American Statistical Association* 113.523 (2018).

[104] Yixin Wang and David M Blei. 'The blessings of multiple causes'. In: *arXiv preprint* (2018).

[105] Ruocheng Guo et al. 'Ignite: A minimax game toward learning individual treatment effects from networked observational data'. In: *Proceedings of the Twenty-Ninth*

International Conference on International Joint Conferences on Artificial Intelligence. 2021, pp. 4534–4540.

[106] Sepp Hochreiter and Jürgen Schmidhuber. 'Long short-term memory'. In: *Neural computation* 9.8 (1997), pp. 1735–1780.

[107] Larry R Medsker and LC Jain. 'Recurrent neural networks'. In: *Design and Applications* (2001).

[108] Jersey Neyman. 'Sur les applications de la théorie des probabilités aux experiences agricoles: Essai des principes'. In: *Roczniki Nauk Rolniczych* 10 (1923).

[109] Paul R Rosenbaum and Donald B Rubin. 'The central role of the propensity score in observational studies for causal effects'. In: *Biometrika* (1983).

[110] Judea Pearl et al. 'Causal inference in statistics: an overview'. In: *Statistics surveys* (2009).

[111] Kyunghyun Cho et al. 'Learning phrase representations using RNN encoder-decoder for statistical machine translation'. In: *arXiv preprint* (2014).

[112] Minh-Thang Luong, Hieu Pham and Christopher D Manning. 'Effective approaches to attention-based neural machine translation'. In: *arXiv preprint* (2015).

[113] Ashish Vaswani et al. 'Attention is all you need'. In: *Advances in Neural Information Processing Systems*. 2017.

[114] Yaroslav Ganin et al. 'Domain-adversarial training of neural networks'. In: *The Journal of Machine Learning Research* 17.1 (2016).

[115] Jonathan K Pritchard, Matthew Stephens and Peter Donnelly. 'Inference of population structure using multilocus genotype data'. In: *Genetics* 155.2 (2000).

[116] Victor Veitch, Dhanya Sridhar and David M Blei. 'Using text embeddings for causal inference'. In: *arXiv preprint* (2019).

[117] Terence C Mills and Terence C Mills. *Time series techniques for economists*. 1991.

[118] Cort J Willmott and Kenji Matsuura. 'Advantages of the mean absolute error (MAE) over the root mean square error (RMSE) in assessing average model performance'. In: *Climate Research* 30.1 (2005).

[119] Leo Breiman. 'Random forests'. In: *Machine Learning* 45.1 (2001).

[120] Ruocheng Guo, Jundong Li and Huan Liu. 'Learning individual causal effects from networked observational data'. In: *ACM International Conference on Web Search and Data Mining*. 2020.

[121] Ludger Rüschendorf. 'The Wasserstein distance and approximation theorems'. In: *Probability Theory and Related Fields* 70.1 (1985).

[122] Sijia Li et al. 'The impact of COVID-19 epidemic declaration on psychological consequences: a study on active Weibo users'. In: *International journal of environmental research and public health* (2020).

[123] John L Romano. ‘Politics of Prevention: Reflections From the COVID-19 Pandemic’. In: *Journal of Prevention and Health Promotion* (2020).

[124] John Daniel. ‘Education and the COVID-19 pandemic’. In: *Prospects* (2020).

[125] Betty Pfefferbaum and Carol S North. ‘Mental health and the Covid-19 pandemic’. In: *New England Journal of Medicine* (2020).

[126] Shabir Ahmad Lone and Aijaz Ahmad. ‘COVID-19 pandemic–an African perspective’. In: *Emerging microbes & infections* (2020).

[127] M Mofijur et al. ‘Impact of COVID-19 on the social, economic, environmental and energy domains: Lessons learnt from a global pandemic’. In: *Sustainable production and consumption* (2021).

[128] Mirko Manchia et al. ‘The impact of the prolonged COVID-19 pandemic on stress resilience and mental health: A critical review across waves’. In: *European Neuropsychopharmacology* (2022).

[129] Thomas Hale et al. ‘A global panel database of pandemic policies (Oxford COVID-19 Government Response Tracker)’. In: *Nature Human Behaviour* (2021).

[130] Solomon Hsiang et al. ‘The effect of large-scale anti-contagion policies on the COVID-19 pandemic’. In: *Nature* (2020).

[131] Kavita Shah Arora, Jaclyn T Mauch and Kelly Smith Gibson. ‘Labor and delivery visitor policies during the COVID-19 pandemic: balancing risks and benefits’. In: *Jama* (2020).

[132] Andrea Galimberti et al. ‘Rethinking urban and food policies to improve citizens safety after COVID-19 pandemic’. In: *Frontiers in Nutrition* (2020).

[133] Fernando A Wilson and Jim P Stimpson. ‘US policies increase vulnerability of immigrant communities to the COVID-19 pandemic’. In: *Annals of global health* (2020).

[134] Edward Kong and Daniel Prinz. ‘Disentangling policy effects using proxy data: Which shutdown policies affected unemployment during the COVID-19 pandemic?’ In: *Journal of Public Economics* (2020).

[135] Daniel T Halperin et al. ‘Revisiting COVID-19 policies: 10 evidence-based recommendations for where to go from here’. In: *BMC public health* (2021).

[136] Aaron Miller et al. ‘Correlation between universal BCG vaccination policy and reduced mortality for COVID-19’. In: *MedRxiv* (2020).

[137] Jiwei Jia et al. ‘Modeling the control of COVID-19: impact of policy interventions and meteorological factors’. In: *arXiv preprint* (2020).

[138] Zeynep Ertem, Ozgur M Araz and Mayteé Cruz-Aponte. ‘A decision analytic approach for social distancing policies during early stages of COVID-19 pandemic’. In: *Decision Support Systems* (2021).

[139] Andrea Riccardo Migone. 'The influence of national policy characteristics on COVID-19 containment policies: A comparative analysis'. In: *Policy Design and Practice* (2020).

[140] Bert George et al. 'A guide to benchmarking COVID-19 performance data'. In: *Public Administration Review* (2020).

[141] Thomas C Chalmers et al. 'A method for assessing the quality of a randomized control trial'. In: *Controlled clinical trials* (1981).

[142] Clement Adebamowo et al. 'Randomised controlled trials for Ebola: practical and ethical issues'. In: *The Lancet* (2014).

[143] Alberto Abadie. 'Semiparametric difference-in-differences estimators'. In: *The Review of Economic Studies* (2005).

[144] Timo Mitze, Reinhold Kosfeld, Johannes Rode et al. 'Face masks considerably reduce Covid-19 cases in Germany-A synthetic control method approach'. In: (2020).

[145] Ron Kohavi et al. 'Online controlled experiments at large scale'. In: *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 2013.

[146] Eric J Tchetgen Tchetgen and Tyler J VanderWeele. 'On causal inference in the presence of interference'. In: *Statistical methods in medical research* (2012).

[147] Johan Ugander et al. 'Graph cluster randomization: Network exposure to multiple universes'. In: *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 2013.

[148] Yuan Yuan, Kristen Altenburger and Farshad Kooti. 'Causal Network Motifs: Identifying Heterogeneous Spillover Effects in A/B Tests'. In: *the Web Conference*. 2021.

[149] Song Bai, Feihu Zhang and Philip HS Torr. 'Hypergraph convolution and hypergraph attention'. In: *Pattern Recognition* 110 (2021), p. 107637.

[150] Yifan Feng et al. 'Hypergraph neural networks'. In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 33. 01. 2019, pp. 3558–3565.

[151] Naganand Yadati et al. 'Hypergcn: Hypergraph convolutional networks for semi-supervised classification'. In: *arXiv preprint* 22 (2018).

[152] Donald B Rubin. 'Randomization analysis of experimental data: The Fisher randomization test comment'. In: *Journal of the American Statistical Association* 75.371 (1980), pp. 591–593.

[153] Cédric Villani et al. *Optimal transport: old and new*. Vol. 338. Springer, 2009.

[154] Ruochi Zhang, Yuesong Zou and Jian Ma. 'Hyper-SAGNN: a self-attention based graph neural network for hypergraphs'. In: *arXiv preprint* (2019).

[155] Kaize Ding et al. 'Be more with less: Hypergraph attention networks for inductive text classification'. In: *arXiv preprint* (2020).

[156] Rossana Mastrandrea, Julie Fournet and Alain Barrat. 'Contact patterns in a high school: a comparison between data collected using wearable sensors, contact diaries and friendship surveys'. In: *PloS one* 10.9 (2015), e0136497.

[157] Austin R Benson et al. 'Simplicial closure and higher-order link prediction'. In: *Proceedings of the National Academy of Sciences* 115.48 (2018), E11221–E11230.

[158] Mengting Wan and Julian McAuley. 'Item recommendation on monotonic behavior chains'. In: *Proceedings of the 12th ACM conference on recommender systems*. 2018, pp. 86–94.

[159] Mengting Wan et al. 'Fine-Grained Spoiler Detection from Large-Scale Review Corpora'. In: *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*. 2019, pp. 2605–2610.

[160] Christopher Morris et al. 'Weisfeiler and leman go neural: Higher-order graph neural networks'. In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 33. 01. 2019, pp. 4602–4609.

[161] Arthur Gretton et al. 'Measuring statistical dependence with Hilbert-Schmidt norms'. In: *International conference on algorithmic learning theory*. Springer. 2005, pp. 63–77.

[162] Thomas N Kipf and Max Welling. 'Semi-supervised classification with graph convolutional networks'. In: *arXiv preprint* (2016).

[163] Panos Toulis, Alexander Volfovsky and Edoardo M Airoldi. 'Propensity score methodology in the presence of network entanglement between treatments'. In: *arXiv preprint* (2018).

[164] Panos Toulis, Alexander Volfovsky and Edoardo M Airoldi. 'Estimating causal effects when treatments are entangled by network dynamics'. In: (2021).

[165] Uri Shalit, Fredrik D Johansson and David Sontag. 'Estimating individual treatment effect: generalization bounds and algorithms'. In: *Proceedings of the 34th International Conference on Machine Learning-Volume 70*. JMLR. org. 2017, pp. 3076–3085.

[166] Joshua D Angrist and Jörn-Steffen Pischke. *Mostly harmless econometrics: An empiricist's companion*. Princeton university press, 2009.

[167] Hojjat Salehinejad et al. 'Recent advances in recurrent neural networks'. In: *arXiv preprint* (2017).

[168] Sören R Künzel et al. 'Metalearners for estimating heterogeneous treatment effects using machine learning'. In: *Proceedings of the national academy of sciences* 116.10 (2019), pp. 4156–4165.

[169] Paul Erdős, Alfréd Rényi et al. 'On the evolution of random graphs'. In: *Publ. Math. Inst. Hung. Acad. Sci* (1960).

[170] Aric Hagberg, Pieter Swart and Daniel S Chult. *Exploring network structure, dynamics, and function using NetworkX*. Tech. rep. Los Alamos National Lab.(LANL), Los Alamos, NM (United States), 2008.

[171] Carlene A Muto et al. 'SHEA guideline for preventing nosocomial transmission of multidrug-resistant strains of Staphylococcus aureus and enterococcus'. In: *Infection Control & Hospital Epidemiology* 24.5 (2003), pp. 362–386.

[172] Erica S Shenoy et al. 'Natural history of colonization with methicillin-resistant Staphylococcus aureus (MRSA) and vancomycin-resistant Enterococcus (VRE): a systematic review'. In: *BMC infectious diseases* 14.1 (2014), pp. 1–13.

[173] Jan Ohst et al. 'The network positions of methicillin resistant Staphylococcus aureus affected units in a regional healthcare system'. In: *EPJ Data Science* 3 (2014), pp. 1–15.

[174] Andrew F Shorr et al. 'A risk score for identifying methicillin-resistant Staphylococcus aureus in patients presenting to the hospital with pneumonia'. In: *BMC infectious diseases* 13.1 (2013), pp. 1–7.

[175] Zonghan Wu et al. 'A comprehensive survey on graph neural networks'. In: *IEEE transactions on neural networks and learning systems* 32.1 (2020), pp. 4–24.

[176] Keyulu Xu et al. 'How Powerful are Graph Neural Networks?' In: *ICLR*. 2018.

[177] Ninareh Mehrabi et al. 'A survey on bias and fairness in machine learning'. In: *ACM CSUR* (2021).

[178] Harjit Singh Sekhon, Sanjit Kumar Roy and James Devlin. 'Perceptions of fairness in financial services: an analysis of distribution channels'. In: *International Journal of Bank Marketing* (2016).

[179] Enyan Dai and Suhang Wang. 'FairGNN: Eliminating the Discrimination in Graph Neural Networks with Limited Sensitive Attribute Information'. In: *arXiv preprint* (2020).

[180] Peizhao Li et al. 'On dyadic fairness: Exploring and mitigating bias in graph connections'. In: *ICLR*. 2020.

[181] Chirag Agarwal, Marinka Zitnik and Himabindu Lakkaraju. 'Towards a Rigorous Theoretical Analysis and Evaluation of GNN Explanations'. In: *arXiv preprint* (2021).

[182] Smriti Bhagat, Graham Cormode and S Muthukrishnan. 'Node classification in social networks'. In: *Social network data analytics*. 2011, pp. 115–148.

[183] David Liben-Nowell and Jon Kleinberg. 'The link-prediction problem for social networks'. In: *JASIST* (2007).

[184] Rex Ying et al. 'Hierarchical graph representation learning with differentiable pooling'. In: *NeurIPS*. 2018.

[185] Matt J Kusner et al. 'Counterfactual Fairness'. In: *Advances in Neural Information Processing Systems* (2017).

[186] Moritz Hardt, Eric Price and Nati Srebro. 'Equality of opportunity in supervised learning'. In: *NeurIPS*. 2016.

[187] Muhammad Bilal Zafar et al. 'Fairness beyond disparate treatment & disparate impact: Learning classification without disparate mistreatment'. In: *WWW*. 2017.

[188] Rich Zemel et al. 'Learning fair representations'. In: *ICML*. 2013.

[189] Jane Bromley et al. 'Signature verification using a "siamese" time delay neural network'. In: *International Journal of Pattern Recognition and Artificial Intelligence* (1993).

[190] William L Hamilton, Rex Ying and Jure Leskovec. 'Inductive representation learning on large graphs'. In: *NeurIPS*. 2017.

[191] Yizhu Jiao et al. 'Sub-graph contrast for scalable self-supervised graph representation learning'. In: *IEEE ICDM*. 2020.

[192] Jiawei Zhang et al. 'Graph-bert: Only attention is needed for learning graph representations'. In: *arXiv preprint* (2020).

[193] Glen Jeh and Jennifer Widom. 'Scaling personalized web search'. In: *IW3C2*. 2003.

[194] Yuhang Song et al. 'Novel human-object interaction detection via adversarial domain generalization'. In: *arXiv preprint* (2020).

[195] Chris Russell et al. 'When worlds collide: integrating different counterfactual assumptions in fairness'. In: *NeurIPS*. 2017.

[196] Sandra Wachter, Brent Mittelstadt and Chris Russell. 'Counterfactual explanations without opening the black box: Automated decisions and the GDPR'. In: *Harv. JL & Tech.* (2017).

[197] Sahil Verma, John Dickerson and Keegan Hines. 'Counterfactual explanations for machine learning: A review'. In: *arXiv preprint* (2020).

[198] Divyat Mahajan, Chenhao Tan and Amit Sharma. 'Preserving causal constraints in counterfactual explanations for machine learning classifiers'. In: *arXiv preprint* (2019).

[199] Saumitra Mishra et al. 'A Survey on the Robustness of Feature Importance and Counterfactual Explanations'. In: *arXiv preprint* (2021).

[200] Danilo Numeroso and Davide Bacciu. 'MEG: Generating Molecular Counterfactual Explanations for Deep Graph Networks'. In: *arXiv preprint* (2021).

[201] Hans-Georg Wolff and Klaus Moser. 'Effects of networking on career success: a longitudinal study.' In: *Journal of applied psychology* 94.1 (2009), p. 196.

[202] URL: https://www.kaggle.com/c/unimelb.

[203] Mohit Bajaj et al. 'Robust counterfactual explanations on graph neural networks'. In: *NeurIPS* (2021).

[204] Carlo Abrate and Francesco Bonchi. ‘Counterfactual graphs for explainable classification of brain networks’. In: *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*. 2021, pp. 2495–2504.

[205] Hao Yuan et al. ‘Explainability in graph neural networks: A taxonomic survey’. In: *arXiv preprint* (2020).

[206] Ilyes Khemakhem et al. ‘Variational autoencoders and nonlinear ica: A unifying framework’. In: 2020.

[207] Martin Simonovsky and Nikos Komodakis. ‘Graphvae: Towards generation of small graphs using variational autoencoders’. In: *ICANN*. 2018.

[208] Diederik P Kingma and Max Welling. ‘Auto-encoding variational bayes’. In: *arXiv preprint* (2013).

[209] Martin Pawelczyk, Klaus Broelemann and Gjergji Kasneci. ‘Learning model-agnostic counterfactual explanations for tabular data’. In: *WWW*. 2020.

[210] Amir-Hossein Karimi et al. ‘Algorithmic recourse under imperfect causal knowledge: a probabilistic approach’. In: *arXiv preprint* (2020).

[211] Amir-Hossein Karimi, Bernhard Schölkopf and Isabel Valera. ‘Algorithmic recourse: from counterfactual explanations to interventions’. In: *ACM FAccT*. 2021.

[212] Aapo Hyvarinen, Hiroaki Sasaki and Richard Turner. ‘Nonlinear ICA using auxiliary variables and generalized contrastive learning’. In: *AISTATS*. 2019.

[213] Shohei Shimizu et al. ‘A linear non-Gaussian acyclic model for causal discovery.’ In: *Journal of Machine Learning Research* (2006).

[214] Ricardo Pio Monti, Kun Zhang and Aapo Hyvärinen. ‘Causal discovery with general non-linear relationships using non-linear ica’. In: *Uncertainty in Artificial Intelligence*. 2020.

[215] P Erdős and A Rényi. ‘On random graphs I. Publicationes Mathematicae (Debrecen)’. In: (1959).

[216] Jing Ma and Jundong Li. ‘Learning causality with graphs’. In: *AI Magazine* 43.4 (2022), pp. 365–375.

[217] Yuhao Kang et al. ‘Multiscale Dynamic Human Mobility Flow Dataset in the U.S. during the COVID-19 Epidemic’. In: *Scientific Data* (2020).

[218] Samuel A Assefa et al. ‘Generating synthetic data in finance: opportunities, challenges and pitfalls’. In: *Proceedings of the First ACM International Conference on AI in Finance*. 2020, pp. 1–8.

[219] Daniel Borrajo, Manuela Veloso and Sameena Shah. ‘Simulating and classifying behavior in adversarial environments based on action-state traces: An application to money laundering’. In: *Proceedings of the First ACM International Conference on AI in Finance*. 2020, pp. 1–8.

[220] Manlio De Domenico et al. ‘The anatomy of a scientific rumor’. In: *Scientific reports* 3.1 (2013), pp. 1–9.

[221] Zhenqin Wu et al. ‘MoleculeNet: a benchmark for molecular machine learning’. In: *Chemical science* (2018).

[222] Greg Landrum et al. ‘RDKit: Open-source cheminformatics’. In: (2006).

[223] Berk Ustun, Alexander Spangher and Yang Liu. ‘Actionable recourse in linear classification’. In: *ACM FAccT*. 2019.

[224] Riccardo Guidotti et al. ‘Local rule-based explanations of black box decision systems’. In: *arXiv preprint* (2018).

[225] Ramaravind K Mothilal, Amit Sharma and Chenhao Tan. ‘Explaining machine learning classifiers through diverse counterfactual explanations’. In: *ACM FAccT*. 2020.

[226] Amit Dhurandhar et al. ‘Explanations based on the missing: Towards contrastive explanations with pertinent negatives’. In: *arXiv preprint* (2018).

APPENDIX A

Details for Chapter 4

A1 Proof of Theory

Before the formal proof of Theorem 1, as there are some common assumptions used in most works as well as ours for ITE estimation, we first present them under our setting:

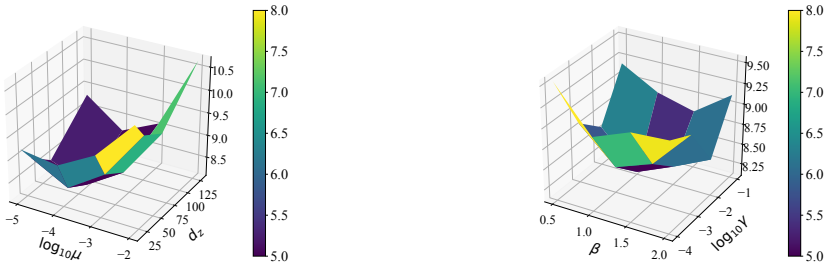
ASSUMPTION 6. *(Consistency). If the treatment history is* $\mathbf{C}^{\leq t}$*, then the observed outcome* $\mathbf{Y}^{\leq t}$ *equals to the potential outcome under treatment* $\mathbf{C}^{\leq t}$.

ASSUMPTION 7. *(Positivity). If the probability* $p(\mathbf{z}^t) \neq 0$*, then the probability of any treatment assignment* $\mathbf{c}^t$ *at time stamp* t *is in the range of* $(0, 1)$*, i.e.,* $0 < p(\mathbf{c}^t|\mathbf{z}^t) < 1$.

ASSUMPTION 8. *(SUTVA). The potential outcomes for any instance are not influenced by the treatment assignment of other instances, and, for each instance, there are no different forms or versions of each treatment level, which lead to different potential outcomes.*

In most existing works, the identification of ITE is based on above three assumptions, along with the strong ignorability assumption. In this work, we relax the strong ignorability assumption and allow the existence of the hidden confounders which could be captured from the time-evolving networked observational data. Based on above premise, we study on the identification of ITE in such data:

Theorem 1. (Identification of ITE) If we recover $p(\mathbf{z}^t|\mathbf{x}^t, \mathbf{H}^t, \mathbf{A}^t)$, $p(y^t|\mathbf{z}^t, c^t)$, then the proposed DNDC can recover the ITE under the causal graph in Fig. 4.1.



(A) μ and d_z (B) β and γ

FIGURE A.1. $\sqrt{\epsilon_{PEHE}}$ with different values of learning rate μ, embedding size d_z, β and γ.

PROOF. Under these above assumptions, we can prove the identification of ITE:

$$\tau^t \overset{(i)}{=} \mathbb{E}_y[y_1^t - y_0^t|\mathbf{x}^t, \mathbf{H}^t, \mathbf{A}^t] \tag{A.1}$$

$$\overset{(ii)}{=} \mathbb{E}_z[\mathbb{E}_y[y_1^t - y_0^t|\mathbf{x}^t, \mathbf{z}^t, \mathbf{H}^t, \mathbf{A}^t]|\mathbf{x}^t, \mathbf{H}^t, \mathbf{A}^t] \tag{A.2}$$

$$\overset{(iii)}{=} \mathbb{E}_z[\mathbb{E}_y[y_1^t - y_0^t|\mathbf{z}^t]|\mathbf{x}^t, \mathbf{H}^t, \mathbf{A}^t] \tag{A.3}$$

$$\overset{(iv)}{=} \mathbb{E}_z[\mathbb{E}_y[y_1^t|\mathbf{z}^t, c^t = 1] - \mathbb{E}_y[y_0^t|\mathbf{z}^t, c^t = 0]|\mathbf{x}^t, \mathbf{H}^t, \mathbf{A}^t] \tag{A.4}$$

$$\overset{(v)}{=} \mathbb{E}_z[\mathbb{E}_y[y_F^t|\mathbf{z}^t, c^t = 1] - \mathbb{E}_y[y_F^t|\mathbf{z}^t, c^t = 0]|\mathbf{x}^t, \mathbf{H}^t, \mathbf{A}^t], \tag{A.5}$$

where $\tau^t = \tau^t(\mathbf{x}^t, \mathbf{H}^t, \mathbf{A}^t)$, we drop the instance index i for simplification. The equation (i) is the definition of ITE in our setting, equation (ii) is a straightforward expectation over $p(\mathbf{z}^t|\mathbf{x}^t, \mathbf{H}^t, \mathbf{A}^t)$, equation (iii) can be inferred from the structure of the causal graph shown in Fig. 4.1, and the SUTVA assumption is implicitly used in the causal graph, equation (iv) is based on the assumption that $\mathbf{z}^t$ contains all the hidden confounders, as well as the positivity assumption, and equation (v) can be inferred from the consistency assumption. Thus, if our framework can correctly model $p(\mathbf{z}^t|\mathbf{x}^t, \mathbf{H}^t, \mathbf{A}^t)$ and $p(y^t|\mathbf{z}^t, c^t)$, then the ITEs can be identified under the causal graph in Fig. 4.1. □

A2 More Experiments for DNDC

A2.1 Hyperparameter Study

To investigate how the values of model hyperparameters affect the performance of DNDC, we assess its performance under different hyperparameter settings, including the learning rate $\mu \in \{10^{-5}, 10^{-4}, 10^{-3}, 10^{-2}\}$, representation dimension $d_z \in \{16, 32, 64, 128\}$, $\beta \in \{0.5, 1.0, 1.5, 2.0\}$, and $\gamma \in \{10^{-4}, 10^{-3}, 10^{-2}, 10^{-1}\}$. Leaving out all the similar observations, we only show the ITE estimation performance $\sqrt{\epsilon_{PEHE}}$ with different values of

learning rates μ, embedding size d_z, β and γ on Flickr in Fig. A.1, where we observe that, the model achieves the best performance when μ is around 10^{-4}, $d_z \in [32, 64]$, $\beta = 1.0$ and $\gamma = 0.01$. Similar observations can be observed on other datasets, as well as ϵ_{ATE}. Generally speaking, our model is not very sensitive to the model parameters in a wide range.

A3 Data and Analysis for Covid-19 Related Information

In this section, we describe how we prepare data for assessing the causal impact of COVID-19 related policies on outbreak dynamics. Some preliminary data analyses are also presented to show the potential capability of capturing the (unobserved) confounders.

A3.1 Observational Data

In general, two basic types of information are indispensable in the causal inference study, i.e., treatment and outcome. Specifically, for treatment, we collect COVID-19 related policies that have been enacted by different counties in the United States throughout 2020; for outcome, we use the numbers of confirmed cases and death cases of different counties throughout 2020. To control the influence of unobserved confounders, we also collect data regarding the covariates of different counties and their relations. In particular, two types of networks among counties are used in our study. In total, after filtering the counties without sufficient data, our data corpus includes 391 counties in the United States. The locations of these selected counties in our corpus are shown in Fig. A.2. We then introduce how we collect and preprocess the data as follows.

Treatment — COVID-19 related policies. We collect the COVID-19 related policies that are in force in the USA throughout 2020 from the Department of Health & Human Services[1]. A total number of 60 policy types are included, along with descriptions and start/end dates. The collected policies include both state-level and county-level ones. For state-level policies, we assume that the policy applies to all counties in the state; for county-level policies, they are considered as only applying to the corresponding county. In order to better analyze the effect of these policies, we perform the following preprocessing: (1) *Policy filtering.* We remove the policy types that are adopted in less than 10% of the counties in our corpus, as we do not have sufficient observational data with respect to them. (2) *Policy categorization.* Based on the goal and some key element of each policy, we group them into three categories: *reduce contacts through social distancing* (henceforth referred to as social distancing), *minimize damage to the economy through reopening* (reopening), and *reduce airborne transmission through mask*

[1] https://catalog.data.gov/dataset/covid-19-state-and-county-policy-orders

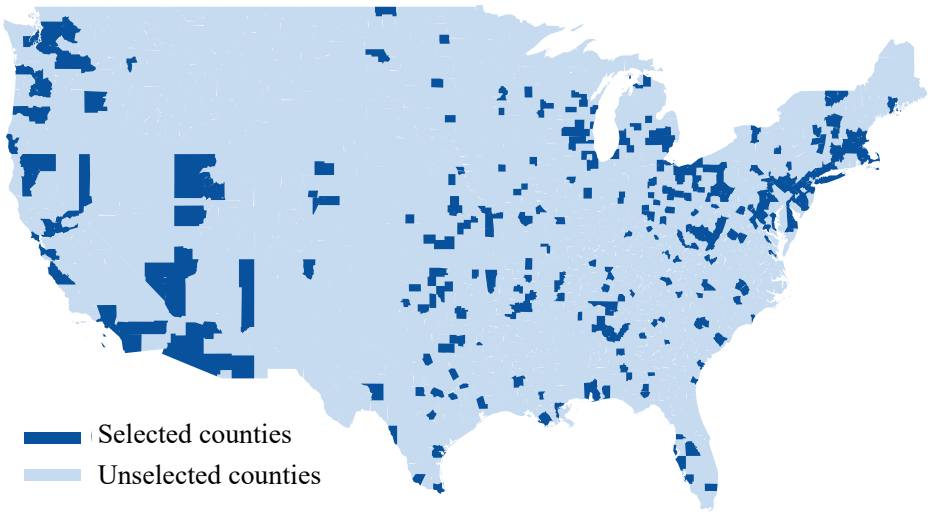


FIGURE A.2. Geolocation of the selected counties in our corpus.

requirements (mask requirements). For each category, Fig. A.3 shows the top-10 policy types adopted by the largest number of counties, and the proportion of counties adopting them throughout 2020.

Outcome — numbers of confirmed and death cases. The daily numbers of confirmed and death cases are collected across these 391 counties from Johns Hopkins Coronavirus Resource Center[2] from January to December, 2020. In our experiments, we regard these numbers as outcomes of each county.

Covariates — popularity of keywords on Google. Unobserved confounders which causally affect the policies and outbreak dynamics are hard to capture. Hence, we use some proxy variables such as covariates of counties to infer these confounders. More specifically, we consider the web search of COVID-19 related keywords (e.g., coronavirus, mask, quarantine, etc.) by residents in these counties as covariates. We collect such web search data from Google Trends[3]. In this process, we first select a set of COVID-19 related keywords, and then obtain their *popularity score* based on the corresponding proportion in the total searches in each county from Google Trend. A higher popularity score implies that a larger proportion in this county has a high vigilance of COVID-19. In total, 19 different keywords are selected, and we obtain a 19-dimensional covariate vector for each county on each day from February to December 2020.

Networks — distance network and mobility flow network. Previous works [120, 42] have shown that network structure among instances can reflect some unobserved confounders.

[2]https://coronavirus.jhu.edu/map.html
[3]https://trends.google.com/trends/?geo=US

Therefore, in this work, two networks including the geographical distance network and mobility flow network among the selected 391 counties are collected as another kind of proxies for confounders. *1) Geographical distance network.* Geographical distance network among counties can be utilized to capture confounders. Intuitively, counties that are geographically closer are more likely to have similar confounders [120] such as residents' vigilance, because they tend to have similar cultural background and social climate. We construct a weighted network among the 391 counties based on the geographical distance from County Distance Database[4], where nodes represent counties, and edge weights are calculated from the corresponding distances between county pairs. Specifically, we select the county pairs with distance less than a threshold $\tau = 100$ kilometers, and set the weight as $1/d(i,j)$ for the edge between the i-th and j-th counties with distance $d(i,j)$. *2) Mobility flow network.* The mobility flow network among counties can also be adopted to capture confounders, as counties with large mobility flow are more likely to have similar confounders [120, 42] (such as residents' vigilance) as they have more communications. We construct an temporal mobility flow network with weighted edges among the 391 counties based on COVID19 USFlows [217], which tracks anonymous GPS pings based on mobile applications. In this temporal network, the total daily volume of mobility flow is aggregated at different scales (e.g., census tract, county, and state) w.r.t. the timeline spanning from February to December in 2020. Each node denotes a county, and the weight of each edge is calculated from the mobility flow volume between the corresponding pair of counties. Specifically, we set the weight as $\frac{\log flow(i,j)}{\max_{i,j} \log flow(i,j)}$ for the edge between the i-th and j-th counties with mobility flow $flow(i,j)$.

A3.2 Preliminary Data Analysis

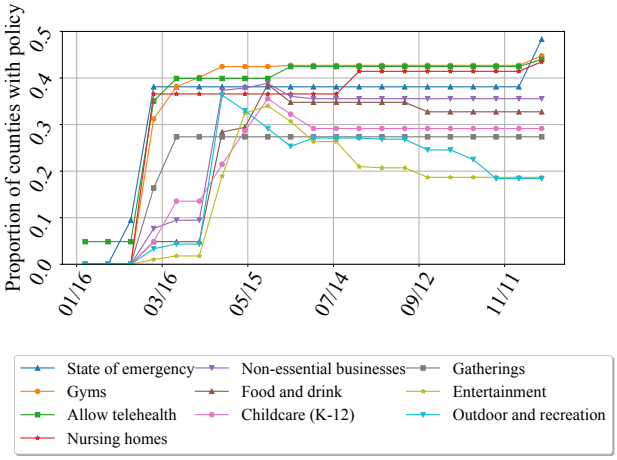
To explore whether the covariates and networks have the potential to capture the (unobserved) confounders, we conduct preliminary data analysis to explore their dependencies with COVID-19 outbreak dynamics (i.e., the number of confirmed/death cases). Due to the space limit, we only show the analysis on the cumulative confirmed cases number of ten counties from Virginia (VA) and Massachusetts (MA) as examples. Similar observations can also be found in other counties, as well as the number of death cases.

Relations between covariates and outbreak dynamics. As proxy variables of unobserved confounders, covariates of counties could have dependencies with the COVID-19 outbreak dynamics (outcome). For example, counties with relatively higher similarity of covariates may also have higher similarity in the number of confirmed cases. If such dependencies exist, it implies that the covariates of counties could be potentially helpful in capturing the unobserved

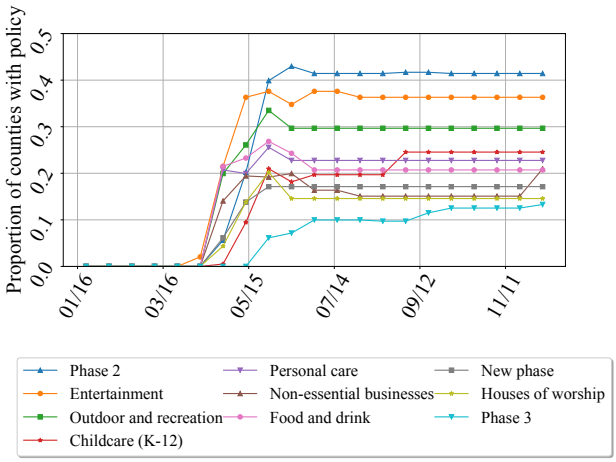
[4] https://www.nber.org/research/data/county-distance-database

confounders. In this regard, we explore whether such dependency exists in our collected covariates of counties, i.e., popularity of COVID-19 related search keywords of counties on Google US. We first compute the bivariate correlation between the daily cumulative confirmed case number series of the chosen counties and all 391 counties in 2020. Then for each of the ten counties, we rank its bivariate correlation values with the 391 counties in an ascending order. The average bivariate correlation value over every 10 percentile of the ranking is reported in Fig. (A.4a). Due to space constraints, we explain the process here only for one of the 19 keywords we used, "social distance". Similar observations can also be drawn based on other keywords. For each county, we represent the daily popularity of "social distance" on Google US as a time series spanning from February to December 2020. The bivariate correlation between counties based on their daily popularity is then calculated. Following the same ranking order in each row of Fig. (A.4a), we report the average bivariate correlation value of their daily popularity over every 10 percentile in Fig. (A.4b) in exponential scale. The results implies that most county pairs with higher bivariate correlation w.r.t. outbreak dynamics tend to have higher keyword popularity similarity. This reveals that our collected covariates have the potential to help capture the confounders.

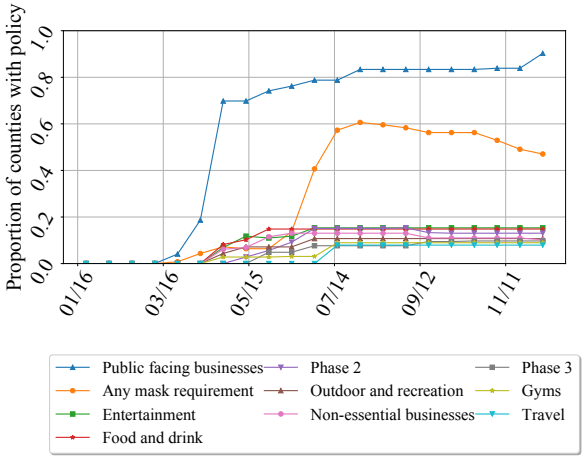
Relations between networks and outbreak dynamics. Networks could also have dependencies with COVID-19 outbreak dynamics, e.g., county pairs with relatively shorter distance or larger mobility flow volume may also have higher similarity in the number of confirmed cases. Similar to the relationship between covariates and outbreak dynamics, such dependencies imply the potential utility of networks in capturing the confounders. Consequently, here we explore whether such dependencies exist in networks among counties. Following the same ranking order in each row of Fig. (A.4a), we also report the corresponding value of their distance and mobility flow (aggregated between Feb. and Dec. in 2020) averaged over every 10 percentile (in log scale) in Fig. (A.4c) and Fig. (A.4d), respectively. We have the following conclusions: 1) most county pairs with relatively lower bivariate correlation w.r.t. the outbreak dynamics are more likely to have larger distance than those with high correlation; 2) most county pairs with relatively higher bivariate correlation w.r.t. the outbreak dynamics tend to have larger human mobility flow volume between them. These imply the potential of the collected networks to capture the confounders.



(A) Social distancing

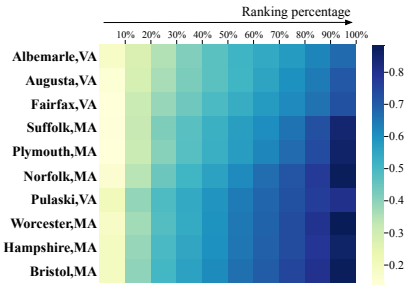


(B) Reopening

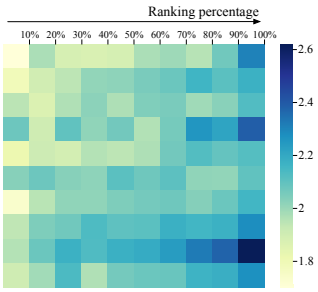


(C) Mask requirement

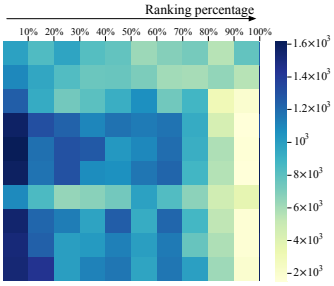
FIGURE A.3. Proportion of counties with policy types in each category over the course of 2020.



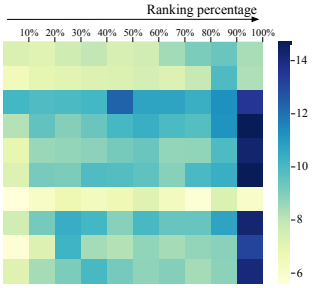
(A) Correlation between confirmed case numbers in different counties.



(B) Correlation between keyword popularity in different counties.



(C) Geographical distance between counties.



(D) Mobility flow volume between counties.

FIGURE A.4. Illustrations reflecting interactions between the selected counties and other counties: (a) bivariate correlation between the confirmed case number series in different counties, (b) bivariate correlation between keyword popularity in different counties, (c) geographic distance between counties, and (d) mobility flow volume between counties. The counties in each row of (a) are ranked by the bivariate correlation of the confirmed case number in an ascending order, and all the results are averaged over every 10 percentile of counties. Each row in (b), (c) and (d) follows the same order of the counties as in (a).

APPENDIX B

Details for Chapter 5

B1 More Experimental Results

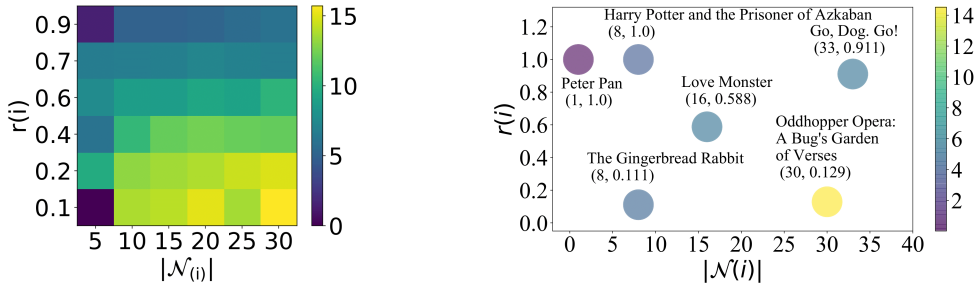
B1.1 ITE Estimation Performance under Different Settings on All the Datasets

In this section, we show the ITE estimation performance under different settings (including the linear and quadratic settings with β among $\{1.0, 3.0, 5.0\}$) on all the datasets in Fig. B.2, Fig. B.3, and Fig. B.4. We can observe that the proposed HyperSCI consistently outperforms the baselines under different settings on all the datasets. The superiority of our framework against baselines becomes more obvious when β is larger (i.e., the interference is stronger), because our framework can better handle the interference in the hypergraph.

B1.2 Case Studies

We further conduct case studies to investigate how the proposed method acts on individuals in responding to their neighboring nodes (i.e., the size of one's neighborhood and the homophily of treatment assignments within one's neighborhood). The neighborhood of i is defined as the set of nodes which are connected with i via any hyperedges, i.e., $\mathcal{N}_i = \bigcup_{e \in \mathcal{E}_i} \{j \in \mathcal{N}_e\}$. The homophily of treatment assignment is defined as the ratio of neighboring nodes which share the same treatment assignment as oneself, i.e., $r(i) = \frac{\sum_{j \in \mathcal{N}_i} \mathbf{1}(t_j = t_i)}{|\mathcal{N}_i|}$. In Fig. B.1a, we show the difference between the ITE estimation results made with the original hypergraph and with the projected graph, w.r.t. $\mathcal{N}_i$ and $r(i)$. Overall, we see larger divergences on individuals with a larger neighborhood size but less agreement with their neighbors in terms of treatment assignments. In Fig. B.1b, we further showcase the insights by presenting several representative children books on the GoodReads dataset. For example, the author of "Peter Pan" had not published many works but these books all received good rating scores, leading to a "consistent" reputation of the author. Therefore, the outcome of the book "Peter Pan" is less impacted by the high-order interference among its neighbors. On the other hand, the

high rating score of the book "Oddhopper Opera" differs from most of its neighbors, leading to a mixed reputation of the author. In this case, the potential outcome is more likely to be affected by the high-order interference on the hypergraph.



(A) Heatmap (B) Case studies

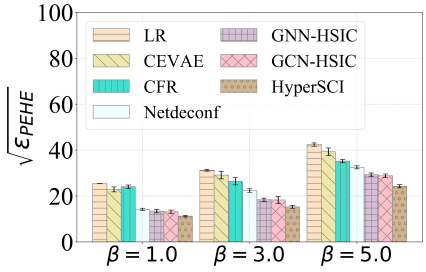
FIGURE B.1. (a) Heatmap: the difference between ITE estimations with hypergraph and with projected ordinary graph on GoodReads. Nodes are divided into 6×6 grids w.r.t. their number of neighbors $|\mathcal{N}_i|$ and the homophily of treatment assignment $r(i)$. (b) Case studies of representative books.

B2 Details of Experimental Settings

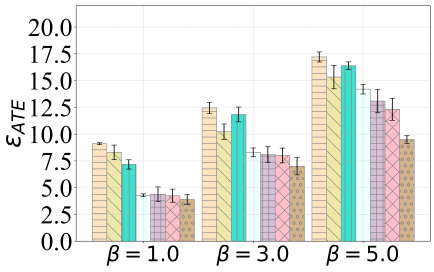
All the experiments are conducted under the following environment:

- Operating system: Ubuntu 18.04
- GPU memory: 16GB
- Pytorch 1.9.0, Cuda ToolKit 11.1, cuDNN 8.0.5

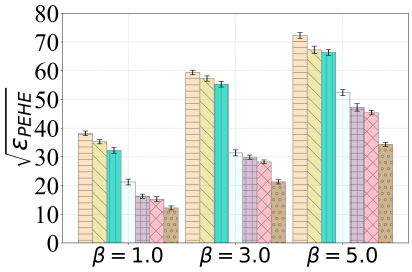
Baseline parameter settings. For the baselines CEVAE, CFR, Netdeconf, GNN-HSIC, and GCN-HSIC, we set the representation dimension as 32, 32, 100, 64, respectively. The numbers of training epochs for these baselines are set as 500. The number of samples in CEVAE in training is set as 5 by default.



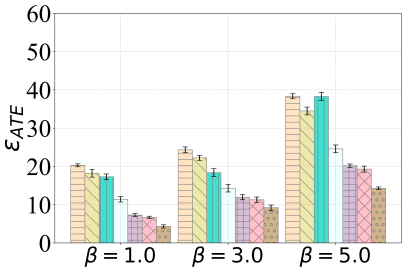
(A) Linear, $\sqrt{\epsilon_{PEHE}}$, CT



(B) Linear, ϵ_{ATE}, CT

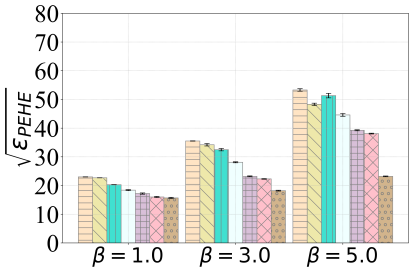


(C) Quadratic, $\sqrt{\epsilon_{PEHE}}$, CT

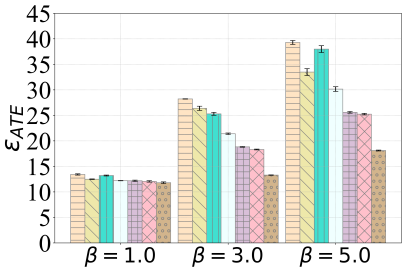


(D) Quadratic, ϵ_{ATE}, CT

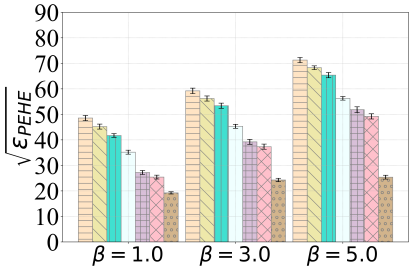
FIGURE B.2. Comparison of the performance of ITE estimation under different settings on Contact dataset.



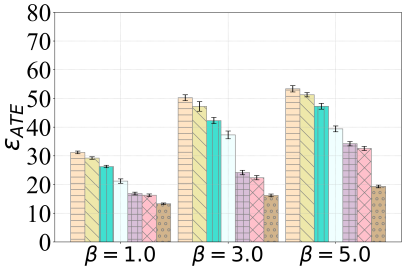
(A) Linear, $\sqrt{\epsilon_{PEHE}}$, GR



(B) Linear, ϵ_{ATE}, GR

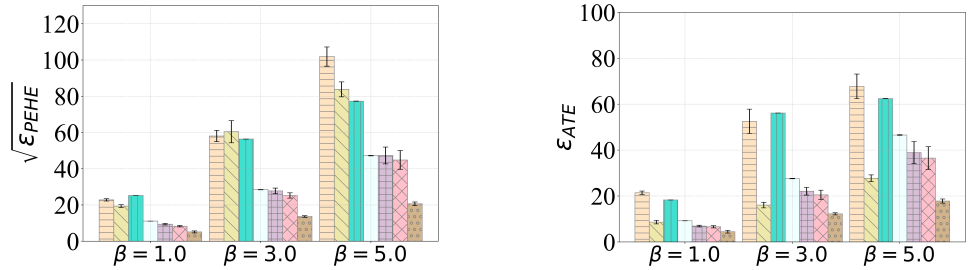


(C) Quadratic, $\sqrt{\epsilon_{PEHE}}$, GR



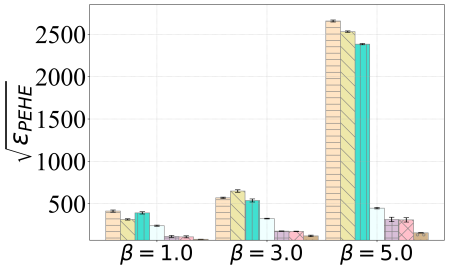
(D) Quadratic, ϵ_{ATE}, GR

FIGURE B.3. Comparison of the performance of ITE estimation under different settings on GoodRead dataset.

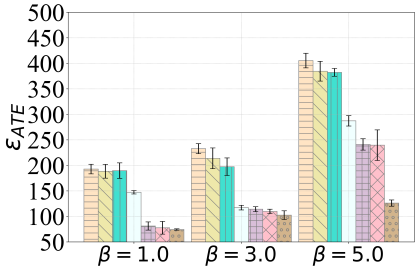


(A) Linear, $\sqrt{\epsilon_{PEHE}}$, MS

(B) Linear, ϵ_{ATE}, MS



(C) Quadratic, $\sqrt{\epsilon_{PEHE}}$, MS



(D) Quadratic, ϵ_{ATE}, MS

FIGURE B.4. Comparison of the performance of ITE estimation under different settings on Microsoft dataset.

APPENDIX C

Details of Chapter 6

C1 Analysis

In this section, we provide detailed analysis of the proposed method NEAT. Again, the outcome generation function defined in Eq. (6.5) is:

$$Y^p = \mathcal{Y}(T^p, X^p, M^p) + g(U^p). \tag{C.1}$$

Inspired by [35], a counterfactual prediction function is defined as:

$$\mathcal{H}(T^p, X^p, M^p) = \mathcal{Y}(T^p, X^p, M^p) + \mathbb{E}[g(U^p)|X^p, M^p]. \tag{C.2}$$

Here, $\mathcal{H}(T^p, X^p, M^p)$ is what we aim to estimate. As the hidden confounders U^p cannot be observed, it is difficult for classical methods to directly fit this function from observational data. Fortunately, based on the assumptions mentioned in Section 6.2, we have:

$$\begin{aligned}
\mathbb{E}[Y^p|X^p, M^p, A^p] &= \mathbb{E}[\mathcal{Y}(T^p, X^p, M^p) + g(U^p)|X^p, M^p, A^p] \\
&= \mathbb{E}[\mathcal{Y}(T^p, X^p, M^p)|X^p, M^p, A^p] \\
&\quad + \mathbb{E}[g(U^p)|X^p, M^p, A^p] \\
&= \mathbb{E}[\mathcal{Y}(T^p, X^p, M^p)|X^p, M^p, A^p] \\
&\quad + \mathbb{E}[g(U^p)|X^p, M^p] \\
&= \int \mathcal{Y}(T^p, X^p, M^p) d\mathcal{F}(T^p|X^p, M^p, A^p) \\
&\quad + \int \mathbb{E}[g(U^p)|X^p, M^p] d\mathcal{F}(T^p|X^p, M^p, A^p) \\
&= \int \mathcal{H}(T^p, X^p, M^p) d\mathcal{F}(T^p|X^p, M^p, A^p),
\end{aligned} \tag{C.3}$$

where $\mathcal{F}(T^p|X^p, M^p, A^p)$ is the conditional distribution of treatment. Here, $\mathcal{H}$ can be estimated with an inverse problem based on observable functions $\mathbb{E}[Y^p|X^p, M^p, A^p]$ and $\mathcal{F}(T^p|X^p, M^p, A^p)$. In our two-stage IV analysis, the first stage estimates $\mathcal{F}(T^p|X^p, M^p, A^p)$, and the second stage estimates $\mathcal{H}(T^p, X^p, M^p)$.

C2 Details of Experiments

In this section, we introduce more details of experiment setup for reproducibility of the experimental results.

C2.1 Baseline Settings

We use the implementation released in the EconML package[1] for S-Learner, causal forest, and DeepIV. Here are more details for settings of each baseline:

- **S-Learner**: We use linear regression as the estimator in S-Learner.
- **Causal forest**: We set the number of trees as 100, the minimum number of samples required to be at a leaf node as 10. The maximum depth of the tree as 10.
- **Counterfactual regression**: The number of epochs is set as 500, the learning rate is 0.001, batch size is 4000, representation dimension is 25. We choose Wasserstein-1 distance [37] for representation balancing.
- **NetDeconf**: We set the number of epochs as 500, learning rate as 0.005, representation dimension as 32, representation balancing weight as 0.5.
- **DNDC**: We set the number of epochs as 800, learning rate as 0.001, representation dimension as 32.
- **DeepIV**: We use the default parameter setting in the EconML package.

C2.2 Experiment Settings

All the experiments in this work are conducted in the following environment:

- Ubuntu 18.04
- Python 3.6
- Scikit-learn 1.0.1
- Scipy 1.6.2
- Pytorch 1.10.1
- Pytorch-geometric 1.7.0
- Networkx 2.5.1
- Numpy 1.19.2
- Cuda 10.1

[1] https://github.com/microsoft/EconML

C2.3 Dataset Details

Transaction. This dataset is collected from the anti-money laundering (AML) financial system [218, 219] which provides transaction records between users over time. At each timestamp, we construct a transaction network to represent the transactions occurring inside this timestamp. In the transaction network, each user is represented by a node, and a transaction is an edge between users. We use the user profiles such as location as their covariates.

Social. This dataset contains a real-world social network of people at different timestamps based on tracking from smart devices [220]. Each node represents a user, and each edge represents a friendship between two users.

APPENDIX D

Details for Chapter 9

D1 Theory

Theorem 2. The evidence lower bound (ELBO) to optimize the framework CLEAR is:

$$\ln P(G^{CF}|S,Y^*,G) \geq \mathbb{E}_Q[\ln P(G^{CF}|Z,S,Y^*,G)] - \mathrm{KL}(Q(Z|G,S,Y^*)\|P(Z|G,S,Y^*)). \tag{D.1}$$

PROOF.

$$\begin{aligned}
&\ln P(G^{CF}|S,Y^*,G) \\
=&\ln \int_Z P(G^{CF},Z|S,Y^*,G)dZ \\
=&\ln \int_Z Q(Z|G,S,Y^*)\frac{P(G^{CF},Z|S,Y^*,G)}{Q(Z|G,S,Y^*)}dZ \\
\geq&\int_Z Q(Z|G,S,Y^*)\ln \frac{P(G^{CF},Z|S,Y^*,G)}{Q(Z|G,S,Y^*)}dZ \\
=&\mathbb{E}_Q[\ln \frac{P(G^{CF},Z|S,Y^*,G)}{Q(Z|G,S,Y^*)}] \\
=&\mathbb{E}_Q[\ln \frac{P(G^{CF}|Z,S,Y^*,G)\cdot P(Z|G,S,Y^*)}{Q(Z|G,S,Y^*)}] \\
=&\mathbb{E}_Q[\ln P(G^{CF}|Z,S,Y^*,G)] - \mathbb{E}_Q[\ln \frac{Q(Z|G,S,Y^*)}{P(Z|G,S,Y^*)}] \\
=&\mathbb{E}_Q[\ln P(G^{CF}|Z,S,Y^*,G)] - \mathrm{KL}(Q(Z|G,S,Y^*)\|P(Z|G,S,Y^*)).
\end{aligned} \tag{D.2}$$

□

D2 Reproducibility

In this section, we provide more details of model implementation and experiment setup for reproducibility of the experimental results.

D2.1 Details of Model Implementation

D2.1.1 Details of the Prediction Model

The prediction model f is implemented with a graph neural network based model. Specifically, this prediction model includes the following components:

- Three layers of graph convolutional network (GCN) [162] with learnable node masks.
- Two graph pooling layers with mean pooling and max pooling, respectively.
- A two-layer multilayer perceptron (MLP) with batch normalization and ReLU activation function.

The prediction model uses negative log likelihood loss. The representation dimension is set as 32. We use Adam optimizer, set the learning rate as 0.001, weight decay as $1e{-}5$, the training epochs as 600, dropout rate as 0.1, and batch size as 500. As shown in Table D.1, we observe that the prediction model f achieves high performance of graph classification on all datasets.

TABLE D.1. Performance of the prediction model on the test data of the three datasets.

Dataset	Community	Ogbg-molhiv	IMDB-M
Accuracy	0.949 ± 0.006	0.897 ± 0.004	0.995 ± 0.002
AUC-ROC	0.993 ± 0.002	0.997 ± 0.002	1.000 ± 0.001
F1-score	0.947 ± 0.005	0.906 ± 0.004	0.994 ± 0.003

D2.1.2 Details of CLEAR

CLEAR is designed in a general way, which can be adaptable to different graph representation learning modules and different techniques in graph generative models. Specifically, in our implementation, we apply a graph convolution [162] based module as the encoder, and use a multilayer perceptron (MLP) as the decoder. We also use MLPs to learn the mean and covariance of the prior distribution of the latent variables. We choose the pairwise distance using L_2 norm to implement $d_X(\cdot)$, and use cross entropy loss to implement $d_A(\cdot)$. We implement the counterfactual prediction loss with the negative log likelihood loss. Following [207], we assume that the maximum number of nodes in the graph is k, and use a graph matching technique to align the input graph and counterfactuals. The detailed implementation contains the following components:

- **Prior distribution:** Two different two-layer MLPs are used to learn the mean and covariance of the prior distribution $P(Z|G, S, Y^*)$, respectively.
- **Encoder:** The encoder contains a single-layer graph convolutional network, a graph pooling layer with mean pooling, and two linear layers with batch normalization and ReLU activation function to learn the mean and covariance of the approximate posterior distribution $Q(Z|G, S, Y^*)$.
- **Decoder:** The decoder uses two three-layer MLPs to output the node features and graph structure of the counterfactual G^{CF}, respectively. These MLPs use batch normalization, and take ReLU as activation function in the middle layers. At the last layer of decoder, the MLP which generates the graph structure uses Sigmoid as activation function to output a probabilistic adjancency matrix $\hat{A}^{CF}$ with elements in range $[0, 1]$.

Inspired by [207], we use a graph matching technique to align the input graph and counterfactuals. Specifically, we learn a graph matching matrix $M = \{0, 1\}^{k\times n}$ to match the generated counterfactual with the original explainee graph. Here, n is the number of nodes in the original graph, and $M_{(i,j)} = 1$ if and only if node i is in G^{CF} and node j is in G, and $M_{(i,j)} = 0$ otherwise.

D2.2 Details of Experiment Setup

D2.2.1 Baseline Settings

Here we introduce more details of baseline setting:

- **Random**: For each explainee graph, it randomly perturbs the graph structure for at most $T = 150$ steps. In each step, at most one edge can be inserted or removed. We stop the process if the perturbed graph can achieve a desired predicted label.
- **GNNExplainer:** For each graph, GNNExplainer [24] outputs an edge mask which estimates the importance of different edges in model prediction. In CFE generation, we set a threshold 0.5 and remove edges with edge mask weight smaller than the threshold. Although GNNExplainer can also identify important node features in a similar way, when we apply GNNExplainer for CFE generation, the perturbation on node features cannot be designed as straightforwardly as the perturbation on graph structure, thus we did not involve perturbation on node features in GNNExplainer.
- **CF-GNNExplainer:** CF-GNNExplainer [25] is originally proposed for node classification tasks, and it only focuses on the perturbations on the graph structure. Originally, for each explainee node, it takes its neighborhood subgraph as input. To apply it on graph classification tasks, we use the graph instance as the neighborhood

TABLE D.2. Detailed statistics of the datasets.

Dataset	Community	Ogbg-molhiv	IMDB-M
# of graphs	$10,000$	$31,957$	$1,160$
Avg # of nodes	20	20.8	9.4
Avg # of edges	45.0	22.4	32.8
Max # of nodes	20	30	15
# of classes	2	2	2
Feature dimension	16	11	2
Avg node degree	2.24	1.07	3.4

subgraph, and assign the graph label as the label for all nodes in the graph. We set the number of iterations to generate counterfactuals for each graph as 150.

- **MEG:** MEG [200] is specifically proposed for molecular prediction tasks. This model explicitly incorporates domain knowledge in chemistry. The CFE generator is developed based on reinforcement learning, and it designs the reward based on the prediction on the counterfactual, as well as the similarity between the original graph and the counterfactual. In each step, MEG enumerates all possible perturbations (e.g., adding an atom) which are valid w.r.t. chemistry rules to form an action set. We apply it to general graphs by removing the constraints of domain knowledge, and enumerating the perturbations as: 1) adding or removing a node; 2) adding or removing an edge; 3) staying the same. We set the number of action steps as 150.

D2.2.2 Datasets

For each dataset, we filter out the graphs with the number of nodes larger than a threshold k. The setting of k (i.e., max # of nodes) can be found in Table D.2. As some of the baselines need to be optimized separately for each graph, we compare the performance of all methods on a small set of test data with 20 graphs for evaluation in RQ1. For other RQs, we evaluate our framework on the whole test data.

1. Community. We first generate a synthetic dataset in which we can fully control the data generation process. In this dataset, each graph consists of two 10-node communities generated using the Erdös-Rényi (E-R) model [215] with edge rate p_1 and p_2, respectively. Specifically, we simulate the data with the following causal model:

$$
\begin{aligned}
S \sim \text{Uniform}(\{0, ..., 9\}),\ p_1 = U_1 \sim \text{Uniform}([0, 1]),\\
U_2 \sim \text{Uniform}([\delta S + b, \delta(S + 1) + b]),\ p_2 = \max\{0, \min\{1, -0.15 p_1 + U_2\}\},\\
X \sim \mathcal{N}(0, I),\ Y \sim \text{Bernoulli}(\text{Sigmoid}(\deg_1(A) - \text{ADG}_1 + \epsilon_y)).
\end{aligned} \tag{D.3}
$$

U_1 and U_2 are two exogenous variables associated with p_1 and p_2, respectively. Notice that p_2 is determined by p_1 and U_2. Here, the auxiliary variable S provides help to infer the value of exogenous variables (specifically, U_2 in this case). We set $\delta = 0.085, b = 0.15$. p_1 and p_2 thereby generate the graph structure inside the two communities, respectively. We also randomly add few edges between these two communities. The edges connecting two communties are randomly generated with an edge rate of 0.05. In this way, the adjacency matrix A of each graph is simulated. $\deg_1(A)$ (determined by p_1) denotes the average node degree in the first community of each graph A. **Label generation:** The label Y is determined by $\deg_1(A)$ together with a Gaussian noise $\epsilon_y \sim \mathcal{N}(0, 0.01^2)$. ADG_1 is a constant, which is the average value of $\deg_1(A)$ over all graphs. **Causality:** To elicit a different predicted label, $\deg_1(A)$ in the counterfactual is supposed to be perturbed, while other variables can remain the same. But considering that with the above causal model, when $\deg_1(A)$ increases (decreases), the average node degree in the second community $\deg_2(A)$ (determined by p_2) should decrease (increase) correspondingly. We take this causal relation $\deg_1(A) \rightarrow \deg_2(A)$ as our causal relation of interest, and denote it as R. Correspondingly, we define a causal constraint for evaluation of causality: "$(\deg_1(A^{CF}) > \deg_1(A)) \Rightarrow (\deg_2(A^{CF}) < \deg_2(A))$" OR "$(\deg_1(A^{CF}) < \deg_1(A)) \Rightarrow (\deg_2(A^{CF}) > \deg_2(A))$".

2. Ogbg-molhiv. Ogbg-molhiv is adopted from the MoleculeNet [221] datasets. All molecules are preprocessed with RDKit [222]. The original node features are 9-dimensional, containing atom features such as atomic number, formal charge and chirality. In this dataset, each graph stands for a molecule, where each node represents an atom, and each edge is a chemical bond. As the ground-truth causal model is unavailable, we simulate the label and causal relation of interest as follows: **Label generation:** $Y \sim \text{Bernoulli}(\text{Sigmoid}(X_1 - \text{AVG}_{x1}))$, where X_1 is the average value of a synthetic node feature over all nodes in each graph. This node feature is generated for each node from distribution $\text{Uniform}(0, 1)$. AVG_{x1} means the average value of X_1 over all graphs. **Causality:** We also add a causal relation of interest R between X_1 and another synthetic node feature X_2: $X_2 = U_2 + 0.5X_1$. Here U_2 is simulated in a similar way as the Community dataset. Correspondingly, we have the following causal constraint: "$(X_1^{CF} > X_1) \Rightarrow (X_2^{CF} > X_2)$" OR "$(X_1^{CF} < X_1) \Rightarrow (X_2^{CF} < X_2)$".

3. IMDB-M. This dataset contains movie collaboration networks from IMDB. In each graph, each node represents an actor or an actress, and each edge represents the collaboration between two actors or actresses in the same movie. Similarly as the above datasets, we simulate the label and causal relation of interest as follows: **Label generation:** $Y \sim \text{Bernoulli}(\text{Sigmoid}(\deg(A) - \text{ADG} + \epsilon_y)$. $\deg(A)$ is the average node degree in graph with adjacency matrix A. ADG is the average value of $\deg(A)$ over all graphs. **Causality:** We also add a causal relation of interest R from the average node degree to a synthetic node feature:

$X_1 = U_1 + 0.5\text{deg}(A)/\text{ADG}$, where $U_1 \sim \text{Uniform}[0.1S, 0.1S+0.1]$, $S \sim \text{Uniform}\{0, ..., 9\}$. We denote the causal relation $\text{deg}(A) \rightarrow X_1$ as R, and define an associated causal constraint: "$(\text{deg}(A^{CF}) > \text{deg}(A)) \Rightarrow (X_1^{CF} > X_1)$" OR "$(\text{deg}(A^{CF}) < \text{deg}_1(A)) \Rightarrow (X_1^{CF} < X_1)$".

D2.2.3 Experiment Settings

All the experiments are conducted in the following environment:

- Python 3.6
- Pytorch 1.10.1
- Pytorch-geometric 1.7.0
- Scikit-learn 1.0.1
- Scipy 1.6.2
- Networkx 2.5.1
- Numpy 1.19.2
- Cuda 10.1

In all the experiments of counterfactual explanation, each dataset is randomly split into 60%/20%/20% training/validation/test set. Unless otherwise specified, we set the hyperparameters as $\alpha = 5.0$ and $\beta = 10.0$. The batch size is 500, and the representation dimension is 32. The graph prediction models trained on all the above datasets perform well in label prediction (AUC-ROC score over 95% and F1 score over 90% on test data). We use NetworkX [170] to generate synthetic graphs. In our CFE generator CLEAR, the learning rate is 0.001, the number of epochs is 1,000. All the experimental results are averaged over ten repeated executions. The implementation is based on Pytorch. We use the Adam optimizer for model optimization.

D3 More Experimental Results

D3.1 Ablation Study

Fig. D.1 shows the results of ablation studies on the IMDB-M dataset. The observations are generally consistent with the observations on other two datasets as described in Section 4.6.

D3.2 Case Study

To better illustrate the explainability provided by CFE, we further conduct case studies to compare the original graphs and their counterfactuals. In the Community dataset, Fig. D.2

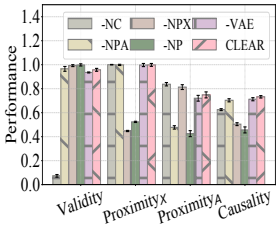


FIGURE D.1. Ablation studies on the IMDB-M dataset.

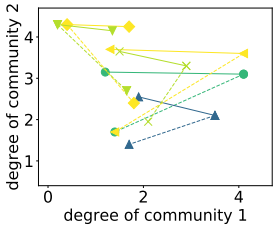
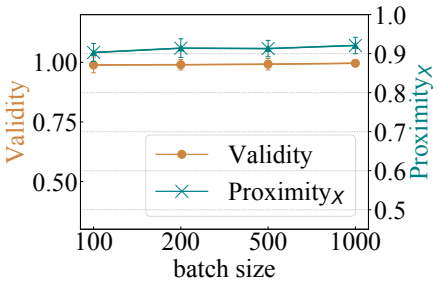
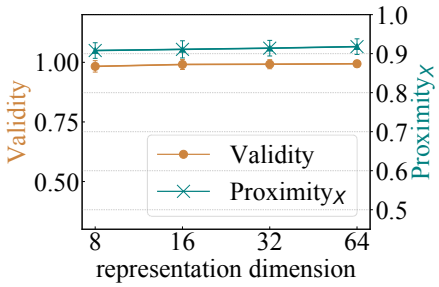


FIGURE D.2. Case study.



(A) Batch size



(B) Representation dimension

FIGURE D.3. Parameter studies on Ogbg-molhiv regarding batch size and representation dimension.

shows the change from original graphs to their counterfactuals w.r.t. the average node degree in the first community and in the second community, i.e., $\deg_1(A)$ and $\deg_2(A)$. Here, Fig. D.2 has the same x-axis and y-axis as Fig. 9.5. In Fig. D.2, we randomly select 6 graphs and show them in different shapes of markers. The colors denote their values of S with the same colorbar in Fig. 9.5(a-c). In Fig. D.2, we connect the pairs (original, counterfactual generated by CLEAR) with solid lines, and connect the pairs (original, counterfactual generated by CLEAR-VAE) with dashed lines. We have the following observations: 1) Compared with the input graph, the counterfactuals generated by CLEAR-VAE and CLEAR both make the correct perturbations to achieve the desired label (moving the variable $\deg_1(A)$ across the decision boundary at around $\deg_1(A) = 2$); 2) The counterfactuals generated by CLEAR better match the causality than CLEAR-VAE in two aspects: a) Qualitatively, the counterfactuals generated by CLEAR better satisfy the causal constraints introduced in the dataset description, i.e., $\deg_2(A)$ increases (decreases) when $\deg_1(A)$ decreases (increases); 2) Quantitatively, the changes from original graphs to their counterfactuals fit in well with the associated structural equations $(\deg_1(A), U_2) \to \deg_2(A)$. Notice that in counterfactuals, $\deg_1(A)$ changes but U_2 is supposed to maintain its original value.

D3.3 Parameter Study

Here, we conduct further parameter study with respect to the batch size and representation dimension. Specifically, we vary the batch size from range $\{100, 500, 1000, 2000\}$, and the representation dimension from range $\{8, 16, 32, 64\}$. From the results shown in Fig. D.3, we observe that the performance of CLEAR under different settings of these parameters is generally stable. This observation further validates the robustness of our framework.

D4 Further Discussion

CFEs in Other Tasks on Graphs. In this work, we mainly focus on the task of graph classification, but it is worth noting that the proposed framework CLEAR can also be used for counterfactual explanations in other tasks such as node classification. More specifically, in a node classification task, CLEAR can generate CFEs for nodes with the same loss function in Eq. (9.5). But differently, the encoder here learns node representations instead of graph representations at the bottleneck layer. Besides, in this case, Y^* is a vector which contains the desired labels for all the training nodes on graph G, and S is the vector of auxiliary variables for all the training nodes. Notice that in a graph, nodes are often not independent. To obtain a valid counterfactual for an explainee node, not only can we change the explainee node's own features and adjacent edges, but we can also change other nodes' features or any other part of the graph structure. Therefore, the decoder still needs to generate a graph G^{CF} as a counterfactual (but this process can be more efficient, as in many scenarios, we only need to generate counterfactuals for each node's neighboring subgraph instead of the whole graph). Similarly, our framework can also be extended to generate CFEs for graphs in other tasks, such as link prediction.

Limitation, Future Work, and Negative Societal Impacts. In this work, we mainly focus on promoting optimization, generalization, and causality in counterfactual explanations on graphs, while other important targets (e.g., actionability [223], sparsity [224], diversity [225], and data manifold closeness [226]) in traditional counterfactual explanations could be considered in graph data in the future. Noticeably, the definition and evaluation metrics with respect to these targets should be specifically tailored for graphs, rather than directly employed in the same way as other types of data. Besides, in terms of causality, another interesting direction is incorporating different levels of prior knowledge and assumptions regarding the underlying causal model into CFE generation on graphs, and quantifying the influence of different levels of prior knowledge and assumptions on the CFE performance. Currently, we have not found any negative societal impact regarding this work.