Minimizing Interference and Selection Bias in Network Experiment Design

Zahra Fatemi, Elena Zheleva
Department of Computer Science, University of Illinois at Chicago
Chicago, IL, USA
{zfatem2, ezheleva}@uic.edu

Abstract

Current approaches to A/B testing in networks focus on limiting interference, the concern that treatment effects can "spill over" from treatment nodes to control nodes and lead to biased causal effect estimation. Prominent methods for network experiment design rely on two-stage randomization, in which sparsely-connected clusters are identified and cluster randomization dictates the node assignment to treatment and control. Here, we show that cluster randomization does not ensure sufficient node randomization and it can lead to selection bias in which treatment and control nodes represent different populations of users. To address this problem, we propose a principled framework for network experiment design which jointly minimizes interference and selection bias. We introduce the concepts of *edge spillover probability* and *cluster matching* and demonstrate their importance for designing network A/B testing. Our experiments on a number of real-world datasets show that our proposed framework leads to significantly lower error in causal effect estimation than existing solutions.

Introduction

The gold standard for inferring causality is the use of *controlled experiments*, also known as A/B tests and randomized controlled trials, in which experimenters can assign treatment (e.g. prompt users to vote) to a random subset of a population of interest and compare their outcome with the outcome of a control group, randomly selected from the same population (e.g., a group of users who were not prompted to vote). Through randomization, the experimenter can control for confounding variables that are not present in the data but can impact the treatment and outcome assignment (e.g. distance from voting polls location) and to assess whether the treatment can cause the target variable (e.g. vote) to change. Controlled experiments are widely used in the social and biological sciences (Imbens and Rubin 2015), and have numerous applications in industry (Varian 2016; Kohavi et al. 2013), from understanding the impact of personalization algorithms to measuring incremental revenue due to ads.

While it is straightforward to randomly assign treatment and control to units that are i.i.d., it is much harder to do for units that interact with each other. One of the challenges in network experiment design is dealing with interference (or spillover), the problem of treatment "spilling over" from a treated node to a control node through a shared edge, e.g., information flowing from person to person in an online social network, and diseases spreading between people interacting in the same physical space. The presence of interference breaks the Stable Unit Treatment Value Assumption (SUTVA), the assumption that one unit's outcomes are unaffected by another unit's treatment assignment, and challenges the validity of causal inference (Imbens and Rubin 2015). Since SUTVA is hard to guarantee in real-world scenarios, recent research on causal inference from graphs focuses on designing controlled experiments in a way that minimizes interference.

Prominent methods for interference minimization in controlled experiments rely on graph clustering (Eckles, Karrer, and Ugander 2017; Pouget-Abadie et al. 2018; Saveski et al. 2017; Ugander et al. 2013). Graph clustering aims to find densely connected clusters of nodes, such that few edges exist across clusters (Schaeffer 2007). The basic idea of applying it to causal inference is that little interference can occur between nodes in different clusters. Treatment and control is assigned at the cluster level, and the cluster assignment dictates the node assignment within each cluster, an experiment design known as two-stage randomization (Basse and Feller 2018).

In this work, we make the key observation that there is an inherent tradeoff between interference and selection bias in cluster-based randomization based on the chosen number of clusters (as demonstrated in Figure 1). Due to the heterogeneity of real-world graphs, discovered clusters can be very different from each other, and the nodes in these clusters may not represent the same underlying population. For example, a treatment cluster may represent "predominantly Democrats from Oklahoma" while a control cluster may represent "predominantly Republicans from New York." Therefore, cluster randomization can lead to selection bias in the data with causal effects that are confounded by the difference in node features of each cluster, rather than the presence or absence of a treatment. Ideally, treatment and control groups should represent the same populations, e.g., there should be clusters of "predominantly Democrats from Oklahoma" assigned to both treatment and control. A

common method for dealing with selection bias in observational treatment and control data is matching, where nodes are matched based on their similarity and then assigned randomly to treatment and control (Stuart 2010). However, there are no methods for incorporating node matching into matching graph clusters, and our work is the first to propose such a method.

Our second main contribution is in introducing the concept of "edge spillover probability" and account for it in the design. Clustering a connected graph component is guaranteed to leave edges between clusters, therefore removing interference completely is impossible. At the same time, some node pairs are more likely to interact than others and assigning such pairs to different treatment groups is more likely to lead to undesired spillover (and biased causal effect estimation) than separating pairs with low probability of interaction.

The goal of our work is to develop a framework for network experiment design in a way that minimizes both selection bias and interference. We propose *CMatch*, a two-stage framework that achieves this goal through a novel objective function for matching clusters and combining node matching with weighted graph clustering. While the idea of using graph clustering to address interference is not new (Ugander et al. 2013; Saveski et al. 2017; Eckles, Karrer, and Ugander 2017), incorporating node matching and edge spillover probabilities into it is novel.

Related Work

Causal inference models are studied by multiple disciplines, including statistics (Imbens and Rubin 2015), computer science (Pearl 2009), and the social sciences (Varian 2016). Here, we review relevant work on causal inference for graphs.

Graph mining. Our framework relies on three preprocessing steps which can leverage existing graph mining algorithms for edge strength estimation, graph clustering, and node representation learning. The goal of edge (or tie or relationship) strength estimation is to assign each existing edge a numeric value which corresponds to a metric of interest for a pair of nodes, such as how close of friends two people are or how likely they are to communicate. Existing approaches rely on topological proximity (Liben-Nowell and Kleinberg 2007), supervised models on node attributes (Gilbert and Karahalios 2009), or latent variable models (Li et al. 2010). Graph clustering aims to find subgraph clusters with high intra-cluster and low inter-cluster edge density (Yang and Leskovec 2015; Zhou, Cheng, and Yu 2009). A number of algorithms exist for weighted graph clustering (Schaeffer 2007). Node representation learning approaches range from graph motifs (Milo et al. 2002) to embedding representations (Hamilton, Ying, and Leskovec 2017) and statistical relational learning (SRL) (Rossi et al. 2012).

Dealing with interference bias. Recent work that addresses interference in graphs relies on separating data samples through graph clustering (Backstrom and Kleinberg 2011; Eckles, Karrer, and Ugander 2017; Gui et al. 2015; Pouget-Abadie et al. 2018; Saveski et al. 2017; Ugander et al. 2013), relational d-separation (Lee and Honavar 2016; Maier et al. 2010; 2013; Marazopoulou, Maier, and Jensen 2015; Rattigan, Maier, and Jensen 2011), or sequential randomization design (Toulis and Kao 2013). Since our work is closest to the approaches based on graph clustering, we use these approaches as baselines in our experiments. None of the existing approaches account for interference heterogeneity and the fact that different edges can have different spillover probabilities; this is one of our main contributions.

Matching and selection bias. In controlled experiments, the treatment assignment is randomized by the experimenter, whereas in estimating causal effects from observational data, the process by which the treatment is assigned is not decided by the experimenter and is often unknown. Matching is a prominent method for mimicking randomization in observational data by pairing treated units with similar untreated units. Then, the causal effect of interest is estimated based on the matched pairs, rather than the full set of units present in the data, thus reducing the selection bias in observational data (Stuart 2010). There are two main approaches to matching, fully blocked and propensity score matching (PSM) (Stuart 2010). Fully blocked matching selects pairs of units whose distance in covariate space is under a pre-determined distance threshold. PSM models the treatment variable based on the observed covariates and matches units which have the same likelihood of treatment. The few research articles that look at the problem of matching for relational domains (Oktay, Taylor, and Jensen 2010; Arbour et al. 2014) consider SRL data representations. None of them consider cluster matching for two-stage design which is one of our contributions.

Network experiment design

The goal of designing *network experiments* is to ensure reliable causal effect estimation in controlled experiments for graphs. As a running example, imagine that we are interested to test whether showing a social media post about the benefits of voting would lead to a higher voter turnout. In this section, we present our data model, give a brief overview of the potential outcomes framework for causal inference, and present the challenges with causal effect estimation in graphs. Then, we describe the causal effects of interest and formally define the problem of network experiment design.

Data model

A graph $G = (V, E)$ consists of a set of n nodes V and a set of edges $E = \{e_{ij}\}$ where e_{ij} denotes that there is an edge between node $v_i \in V$ and node $v_j \in V$. Let N_i denote the set of neighbors for node v_i, i.e. set of nodes that share an edge with v_i. Let $v_i.X$ denote the pre-treatment node feature variables (e.g., Twitter user features) for unit v_i. Let $v_i.Y$ denote the outcome variable of interest for each node v_i (e.g., voting), and $v_i.T \in \{0, 1\}$ denote whether node v_i (e.g., social media user) has been treated (e.g., shown a post about the benefits of voting), $v_i.T = 1$, or not, $v_i.T = 0$. V_1 and V_0 indicate the sets of units in treatment and control groups, respectively. For simplicity, we assume that both $v_i.T$ and $v_i.Y$ are binary variables. The edge spillover probability $e_{ij}.p$ refers to the probability of interference occurring between two nodes.

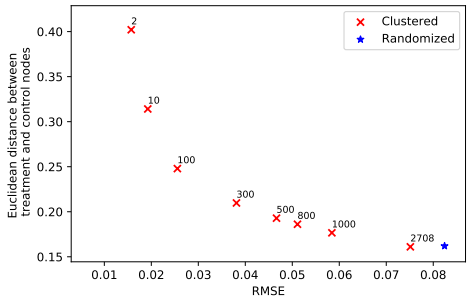

Figure 1: The tradeoff between selection bias (distance) and undesired spillover (RMSE) in cluster-based randomization; each data point is annotated with the number of clusters.

Potential outcomes framework

The fundamental problem of causal inference is that we can observe the outcome of a target variable for an individual v_i in either the treatment or control group but not in both. Let $v_i.y(1)$ and $v_i.y(0)$ denote the *potential outcomes* of $v_i.y$ if unit v_i were assigned to the treatment or control group, respectively. The treatment effect (or causal effect) is the difference $g(i) = v_i.y(1) - v_i.y(0)$. Since we can never observe the outcome of a unit under both treatment and control simultaneously, the effect $\hat{\mu}$ of a treatment on an outcome is typically calculated through averaging outcomes over treatment and control groups via difference-in-means: $\hat{\mu} = \overline{V_1.Y} - \overline{V_0.Y}$ (Stuart 2010). For the treatment effect to be estimable, the following *identifiability* assumptions have to hold:

- *Stable unit treatment value assumption* (SUTVA) refers to the assumption that the outcomes $v_i.y(1)$ and $v_i.y(0)$ are independent of the treatment assignment of other units: $\{v_i.y(1), v_i.y(0)\} \perp\!\!\!\perp v_j.T, \forall v_j \neq v_i \in V$.
- *Ignorability* (Imbens and Rubin 2015) – also known as *conditional independence* (Pearl 2009) and *absence of unmeasured confoundness* – is the assumption that all variables $v_i.X$ that can influence both the treatment and outcome $v_i.Y$ are observed in the data and there are no unmeasured confounding variables that can cause changes in both the treatment and the outcome: $\{v_i.y(1), v_i.y(0)\} \perp\!\!\!\perp v_i.T \mid v_i.X$.
- *Overlap* is the assumption that each unit assigned to the treatment or control group could have been assigned to the other group. This is also known as *positivity* assumption: $P(v_i.T|v_i.X) > 0$ for all units and all possible T and X.

Challenges with causal effect estimation in graphs

Challenge no. 1: It is hard to separate a graph into treatment and control nodes without leaving edges across. The presence of interference breaks the SUTVA assumption and leads to biased causal effect estimation in relational data. Two-stage experimental design addresses this problem by finding groups of units that are unlikely to interact with each other (stage 1) and then randomly assigning each group to treatment and control (stage 2). Clustering has been proposed as a way to discover such groups that are strongly connected within but loosely connected across, thus finding treatment and control subgraphs that have low probability of spillover from one to the other. However, due to the density of real-world graphs, graph clustering techniques can leave as many as 65% to 79% of edges as inter-cluster edges (Table 2 in (Saveski et al. 2017)). Leaving these edges across treatment and control nodes would lead to a large amount of spillover. Incorporating information about the edge probability of spillover into the clustering helps alleviate this problem and is one of the main contributions of our work.

Challenge no. 2: There is a tradeoff between interference and selection bias in cluster-based network experiments. While randomization of i.i.d. units in controlled experiments can guarantee ignorability and overlap, two-stage design does not. One of the key observations in our work is that dependent on the number of clusters, there is a tradeoff between interference and selection bias in terms of the treatment and control group not representing the same underlying distribution. Figure 1 illustrates this tradeoff for Cora, one of the datasets in our experiments, using $reLDG$ as the clustering method. When a network is separated into very few clusters, the Euclidean distance between nodes in treatment and control clusters is larger than the Euclidean distance when a lot of clusters are produced over the same network (e.g., 0.4 vs. 0.18 for 2 and 1000 clusters). This is intuitive because as the clusters get smaller and smaller, their randomization gets closer to mimicking full node randomization (shown as a star). At the same time, a larger number of clusters translates to a higher likelihood of edges between treatment and control nodes, which leads to higher undesired spillover and causal effect estimation error (e.g., 0.015 vs. 0.059 for 2 and 1000 clusters).

Types of causal effects in networks

In real-world scenarios, we are interested in estimating the *Total Treatment Effect (TTE)*. Let $\mathbf{Z} \in \{0,1\}^N$ be the treatment assignment vector of all nodes. *TTE* is defined as the outcome difference between two alternative universes, one in which all nodes are assigned to treatment ($\mathbf{Z_1} = \{1\}^N$) and one in which all nodes are assigned to control ($\mathbf{Z_0} = \{0\}^N$) (Ugander et al. 2013; Saveski et al. 2017):

$$TTE = \frac{1}{N} \sum_{v_i \in V} (v_i.Y(\mathbf{Z_1}) - v_i.Y(\mathbf{Z_0})).$$

TTE is estimated as averages over the treatment and control group, and it accounts for two types of effects, *individual effects (IE)* and *peer effects (PE)*:

$$\hat{TTE} = \overline{V_1.Y} - \overline{V_0.Y} = IE(V) + PE(V_1) - PE(V_0).$$

Average individual effects (*IE*) reflect the difference in outcome between treated and untreated subjects which can be attributed to the treatment alone. They are estimated as:

$$IE(V) = \mathop{\mathbb{E}}_{v_i \in V}[v_i.Y|v_i.T = 1] - \mathop{\mathbb{E}}_{v_i \in V}[v_i.Y|v_i.T = 0].$$

Average peer effects (*PE*) reflect the difference in outcome that can be attributed to influence by other subjects in the

experiment. Let $N_i.\boldsymbol{\pi}$ denote the vector of treatment assignments to node v_i's neighbors N_i. Average *PE* is estimated as having neighbors with a treatment vector:

$$PE(V) = \mathop{\mathbb{E}}_{v_i \in V}[v_i.Y | v_i.T = t, N_i.\boldsymbol{\pi}] - \mathop{\mathbb{E}}_{v_i \in V}[v_i.Y | v_i.T = t, N_i = \emptyset].$$

Here, we distinguish between two types of peer effects, *allowable peer effects* (*APE*) and *unallowable peer effects* (*UPE*). Allowable peer effects are peer effects that occur within the same treatment group, and they are a natural consequence of network interactions. For example, if a social media company wants to introduce a new feature (e.g., nudging users to vote), it would introduce that feature to all users and the total effect of the feature would include both individual and peer effects. Unallowable peer effects are peer effects that occur across treatment groups and contribute to undesired spillover and incorrect causal effect estimation.

For each node v_i in treatment group t, we have two types of neighbors: 1) neighbors N_i^t in the same treatment class as node v_i with treatment assignment set $N_i^t.\boldsymbol{\pi}$; 2) set of neighbors in a different treatment class $N_i^{\bar{t}}$ ($\bar{t} \neq t$) with treatment assignment denoted by $N_i^{\bar{t}}.\boldsymbol{\pi}$. The *APE* is defined as:

$$APE(V) = \mathop{\mathbb{E}}_{v_i \in V}[v_i.Y | v_i.T = t, N_i^t.\boldsymbol{\pi}] - \mathop{\mathbb{E}}_{v_i \in V}[v_i.Y | v_i.T = t, N_i^t = \emptyset],$$

and the *UPE* is defined as:

$$UPE(V) = \mathop{\mathbb{E}}_{v_i \in V}[v_i.Y | v_i.T = t, N_i^{\bar{t}}.\boldsymbol{\pi}] - \mathop{\mathbb{E}}_{v_i \in V}[v_i.Y | v_i.T = t, N_i^{\bar{t}} = \emptyset].$$

Ideally, we would like to measure $TTE = IE(V) + APE(V_1) - APE(V_0)$. Due to undesired spillover in a controlled experiment, what we are able to measure instead is the overall effect that comprises of both allowable and unallowable peer effects $TTE = IE(V) + APE(V_1) - APE(V_0) + UPE(V_1) - UPE(V_0)$. Therefore, when we designing an experiment for minimum interference, we are interested in setting it up in a way that makes $UPE(V_1) = 0$ and $UPE(V_0) = 0$.

There are two types of pairwise interference that can occur, direct interference and contagion (Ogburn, VanderWeele, and others 2014). What we have described so far is causal effect estimation for direct interference which refers to a treated node v_i ($v_i.X = 1$) influencing the outcome of another node v_j. For example, a treated social media user who sees the post decides to share it with another user who ends up voting as a result. Contagion refers to the outcome of node v_i influencing another node v_j to have the same outcome. For example, a social media user who votes can convince another user to vote. The above definitions of peer effect can also be defined for contagion by conditioning them on neighbor outcomes, rather than neighbor treatments. We leave this exercise to the reader.

Problem definition

The goal of network experiment design is to minimize both unallowable peer effects and selection bias in node assignment to treatment and control. More formally:

Problem 1 (Network experiment design) *Given a graph $G = (\boldsymbol{V}, \boldsymbol{E})$, a set of attributes $\boldsymbol{V.X}$ associated with each node and a set of spillover probabilities $\boldsymbol{E.P}$ associated with the graph edges, we want to construct two sets of nodes, the control nodes $\boldsymbol{V}_0 \in \boldsymbol{V}$ and the treatment nodes $\boldsymbol{V}_1 \in \boldsymbol{V}$ such that:*

a. $\boldsymbol{V}_0 \cap \boldsymbol{V}_1 = \emptyset$
b. $|\boldsymbol{V}_0| + |\boldsymbol{V}_1|$ *is maximized*
c. $\theta = UPE(\boldsymbol{V}_1) - UPE(\boldsymbol{V}_0)$ *is minimized*
d. $\boldsymbol{V}_0.\boldsymbol{X}$ *and* $\boldsymbol{V}_1.\boldsymbol{X}$ *are identically distributed*

This problem definition describes the desired qualities of the experiment design at a high level. The first component ensures that the treatment and control nodes do not overlap. The second component aims to keep as many nodes as possible from $\boldsymbol{V}$ in the final design. The third component minimizes unallowable spillover. The fourth component requires that there is no selection bias between the treatment and control groups. The second and third component are at odds with one another and require a trade off because the lower θ, the lower the number of selected nodes for the experiment $|\boldsymbol{V}_0| + |\boldsymbol{V}_1|$. As we showed in Figure 1, there is also a tradeoff between the third and fourth component. In the next section, we propose a solution to this problem.

CMatch: a network experiment design framework

Our network experiment design framework *CMatch*, illustrated in Figure 2, has two main goals: 1) *spillover minimization* which it achieves through weighted graph clustering, and 2) *selection bias minimization* which it achieves through cluster matching. Clusters in each matched pair are assigned to different treatments, thus achieving covariate balance between treatment and control. The first goal addresses part c of *Problem 1* and the second goal addresses part d. While the first goal can be achieved with existing graph mining algorithms, solving for the second one requires developing novel approaches. To achieve the second goal, we propose an objective function, which can be solved with maximum weighted matching, and present the nuances of operationalizing each step.

Step 1: Interference minimization through weighted graph clustering

Existing cluster-based techniques for network experiment design assume unweighted graphs (Backstrom and Kleinberg 2011; Eckles, Karrer, and Ugander 2017; Gui et al. 2015; Saveski et al. 2017; Ugander et al. 2013) and do not consider that different edges can have different likelihood of spillover. Incorporating information about the edge probability of spillover into the clustering helps alleviate this problem and is one of the main contributions of our work. In order to minimize undesired spillover, we operationalize

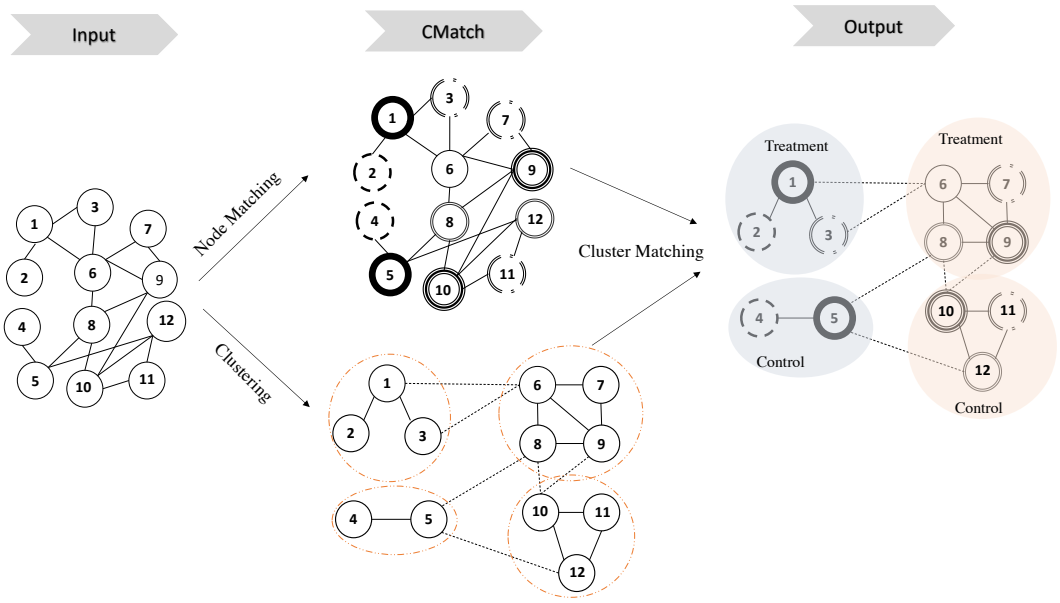

Figure 2: Illustration of $CMatch$ framework for minimizing interference and selection bias in controlled experiments. **Input**: a graph of nodes and the connection between them. **CMatch**: node and cluster matching; the dashed circles indicates the clusters. Matched nodes are represented with a similar circle border. **Output**: assigning the matched cluster pairs to treatment and control randomly; circles with the same color represent matched clusters.

minimizing θ as minimizing the edges, and more specifically the edge spillover probabilities, between treatment and control nodes: $\hat{\theta} = \sum_{\forall v_i \in \boldsymbol{V}_0, \forall v_j \in \boldsymbol{V}_1} e_{ij}.p$. To achieve this, $CMatch$ creates graph clusters for two-stage design by employing two functions, edge spillover probability estimation and weighted graph clustering.

Edge spillover probability estimation. We consider edge strength, how strong the relationship between two nodes is, as a proxy for edge spillover probability. This reflects the notion that the probability of a person influencing a close friend to do something is higher than the probability of influencing an acquaintance. We can use common graph mining techniques to calculate edge strength, including ones based on topological proximity (Liben-Nowell and Kleinberg 2007), supervised classification (Gilbert and Karahalios 2009), or latent variable models (Li et al. 2010).

Weighted graph clustering. In order to incorporate edge strength into clustering, we can use any existing weighted graph clustering algorithm (Enright, van Dongen, and Ouzounis 2002; Schaeffer 2007; Yang and Leskovec 2015). In our experiments, we use a prominent non-parametric algorithm, the *Markov Clustering Algorithm (MCL)* (Enright, van Dongen, and Ouzounis 2002) which applies the idea of random walk for clustering graphs and produces non-overlapping clusters. We also compare this algorithm with *reLDG* which was the basis of previous work (Saveski et al. 2017). One of the advantages of MCL is that it automatically finds the optimal number of clusters, rather than requiring it as input. The main idea behind MCL is that nodes in the same cluster are connected with higher-weighted shortest paths than nodes in different clusters.

Step 2: Selection bias minimization through cluster matching

Randomizing treatment assignment over clusters in a two-stage design does not guarantee that nodes within those clusters would represent random samples of the population. We propose to address this selection bias problem by *cluster matching* and balancing covariates across treatment and control clusters. While methods for matching nodes exist (Stuart 2010; Oktay, Taylor, and Jensen 2010; Arbour et al. 2014), this work is the first to propose methods for matching clusters.

Objective function. The goal of cluster matching is to find pairs of clusters with similar node covariate distributions and assign them to different treatment groups. We propose to capture this through a maximum weighted matching objective over a cluster graph in which each discovered cluster from step 1 is a node and edges between clusters represent their similarity. Suppose that graph G is partitioned into $C = \{c_1, c_2, ..., c_g\}$ clusters. We define $A \in \{0, 1\}$, such that $a_{ij} = 1$ if two clusters c_i and c_j are matched, else $a_{ij} = 0$. $w_{i,j} \in \mathbb{R}$ represents the similarity between two clusters c_i and c_j. Then the objective function of $CMatch$ is as follows:

$$\begin{aligned} &\underset{A}{arg\,max} \quad \sum_{i=1}^{g} \sum_{j=i+1}^{g} (a_{ij} \cdot w_{ij}) \\ &\text{subject to} \quad \forall c_i \in C, \sum_{j=1}^{|c_i|} a_{ij} \leq 1, a_{ij} \in \{0, 1\}. \end{aligned} \tag{1}$$

This objective function maps to a maximum weighted

matching problem for which there is a linear-time approximation algorithm (Duan and Pettie 2014) and a polynomial-time exact algorithm with $O(N^{2.376})$ (Mucha and Sankowski 2004; Harvey 2009).

Solution. In order to operationalize the solution to this objective, the main question that needs to be addressed is: what does it mean for two clusters to be similar? We propose to capture this cluster similarity through matched nodes. The more nodes can be matched based on their covariates across two clusters, the more similar two clusters are. Thus, the operationalization comes down to the following three questions which we address next:

1. What constitutes a ***node match***?
2. How are node matches taken into consideration in computing the pairwise ***cluster weights*** (cluster similarity)?
3. Given a cluster weight, what constitutes a potential cluster match, and thus an edge in the ***cluster graph***?

Once these three questions are addressed, the cluster graph can be built and an existing maximum weighted matching algorithm can be applied on it to find the final cluster matches.

Node Matching. The goal of node matching is to reduce the imbalance between treatment and control groups due to their different feature distributions. Given a node representation, fully blocked matching would look for the most similar nodes based on that representation (Stuart 2010). It is important to note that propensity score matching does not apply here because it models the probability of treatment in observational data and treatment is unknown at the time of designing a controlled experiment. In its simplest form, a node can be represented as a vector of attributes, including node-specific attributes, such as demographic characteristics, and structural attributes, such as node degree. For any two nodes, it is possible to apply an appropriate similarity measure $sim(v_i, v_j)$, in order to match two nodes, including cosine similarity, Jaccard similarity or Euclidean distance.

We consider two different options to match a pair of nodes in different clusters (and ignore matches within the same cluster):

- **Threshold-based node matching (TNM)**: Node v_k in cluster c_i is matched with node v_l from a different cluster c_j if the pairwise similarity of nodes $sim(v_k, v_l) > \alpha$. The threshold α can vary from 0, which liberally matches all pairs of nodes, to the maximum possible similarity which matches nodes only if they are exactly the same. In our experiments, we set α based on the covariate distribution of each dataset and consider different quartiles of pairwise similarity as thresholds. This allows for each node to have multiple possible matches across clusters.
- **Best node matching (BNM)**: Node v_k in cluster c_i is matched with only one node v_l which is most similar to v_k in the whole graph; v_l should be in a different cluster. This is a very conservative matching approach in which each node is uniquely matched but allows the matching to be asymmetric.

Cluster Weights. After the selection of a node matching mechanism, we are ready to define the pairwise similarity of clusters which is the basis of cluster matching. We consider three simple approaches and three more expensive approaches which require maximum weighted matching between nodes:

- **Euclidean distance (E)**: This approach is the simplest of all because it does not consider node matches and it simply calculates the Euclidean distance between the node attribute vector means of two clusters.
- **Matched node count (C)**: The first approach counts the number of matched nodes in each pair of clusters c_i and c_j and consider the count as the clusters' pairwise similarity: $w_{ij} = \sum_{k=1}^{|c_i|} \sum_{l=1}^{|c_j|} r_{kl}^{ij}$. A node in cluster c_i can have multiple matched nodes in c_j.
- **Matched node average similarity (S)**: Instead of the count, this approach considers the average similarity between matched nodes across two clusters c_i and c_j: $w_{ij} = \frac{\sum_{k=1}^{|c_i|} \sum_{l=1}^{|c_j|} r_{kl}^{ij} \cdot sim(v_k, v_l)}{\sum_{k=1}^{|c_i|} \sum_{l=1}^{|c_j|} r_{kl}^{ij}}$.

These first two approaches allow a single node to be matched with multiple nodes in another cluster and each of those matches to count towards the cluster pair weight. In order to distinguish this from a more desirable case in which multiple nodes in one cluster are matched to multiple nodes in another cluster, we propose approaches that allow each node to be considered only once in the matches that count towards the weight. For each pair of clusters, we build a node graph in which an edge is formed between nodes v_i and v_j in the two clusters and the weight of this edge is $sim(v_i, v_j)$. Maximum weighted matching will find the best possible node matches in the two clusters. We consider three different variants for calculating the cluster pair weight based on the maximum weighted matching of nodes:

- **Maximum matched node count (MC)**: This method calculates the cluster weight the same way as **C** except that the matches (whether r_{kl}^{ij} is 0 or 1) are based on the maximum weighted matching result.
- **Maximum matched node average similarity (MS)**: This method calculates the cluster weight the same way as **S** except that the node matches are based on the maximum weighted matching result.
- **Maximum matched node similarity sum (MSS)**: This method calculates the cluster weight similarly to **MS** except that it does not average the node similarity: $w_{ij} = \sum_{k=1}^{|c_i|} \sum_{l=1}^{|c_j|} r_{kl}^{ij} \cdot sim(v_k, v_l)$.

Cluster Graph. Once the cluster similarities of have been determined, we need to decide what similarity constitutes a potential cluster match. Such potential matches are added as edges in the cluster graph which is considered for maximum weighted matching. We consider three different options:

- **Threshold-based cluster matching (TCM)**: Cluster c_i is considered as a potential match of cluster c_j if their weight $w_{i,j} > \beta$. The threshold β can vary from 0, which allows all pairs of clusters to be potential matches, to the maximum possible similarity which allows matching between clusters only if they are exactly the same. In our

experiments, we set β based on the distribution of pairwise similarities and their quartiles as thresholds.

- **Greedy cluster matching (GCM)**: For each cluster c_i, a sorted list of the similarities between c_i and all other clusters is defined. Cluster c_i is considered a potential match only to the cluster with the highest similarity value in the list.

The last step in $CMatch$ runs maximum weighted matching on the cluster graph. For every matched cluster pair, it assigns one cluster to treatment and the other one to control at random. This completes the network experiment design.

Experiments

In this section, we evaluate the advantages of $CMatch$ in minimizing interference and selection bias compared to state-of-the-art methods.

Semi-synthetic data

We consider four real-world attributed network datasets. Table 1 shows the basic dataset characteristics. The *50 Women* dataset (Michell and Amos 1997) incorporates the smoking, alcohol, sport and drug habits of 50 students. $Hamsterster$ (Zheleva et al. 2008) describes the online social network of hamsters and their attributes. $Cora$ and $Citeseer$ (Sen et al. 2008) are two citation networks with binary bag-of-words attributes for each article.

We assume that the underlying probability of activating a node (changing the outcome) due to treatment and allowable peer effects in the treatment group is 0.4 and the underlying probability of activating a control node due to treatment and allowable peer effects is 0.2 which makes the true causal effect $TTE = 0.2$. Based on these probabilities, we randomly assign each node as activated or not. For each inactivated nodes, we simulate two types of interference considering both fixed values (0.1 and 0.5) and values based on the edge weights for $e.p$:

1. Direct interference: each treated neighbor of a control node activates the node with unallowable spillover probability of $e.p$.
2. Contagion: inactive treated and untreated nodes get activated with the unallowable spillover probability of $e.p$ if they are connected to at least one activated node in a different treatment class.

Table 1: Number of nodes, edges and attributes in the datasets

Dataset	Nodes	Edges	Attributes
Citeseer	3,312	4,675	3709
Cora	2,708	5,278	1440
Hamsterster	2,059	10,943	6
50 Women	50	74	4

Main algorithms and baselines

All our baseline and main algorithm variants take an attributed graph as an input and produce a set of clusters, each assigned to treatment, control, or none. For graph clustering, we considered two main algorithms, *Restreaming Linear Deterministic Greedy (reLDG)* (Nishimura and Ugander 2013) and *Markov Clustering Algorithm (MCL)* (Enright, van Dongen, and Ouzounis 2002). $reLDG$ takes as input an unweighted graph and desired number of clusters and produces a graph clustering. $reLDG$ was reported to perform very well in state-of-the-art methods for network experiment design (Saveski et al. 2017). MCL is a non-parametric algorithm which takes as input a weighted graph and produces a graph clustering. The edge weights which correspond to the probabilities of spillover are estimated based on node pair similarity using one minus the normalized L2 norm: $1 - L_2(v_i.x, v_j.x)$.

The main algorithms and baselines are:

- $\boldsymbol{Randomized}$: This algorithm assigns nodes to treatment and control randomly, ignoring the network.
- $\boldsymbol{CR}$ (Saveski et al. 2017): The *Completely Randomized (CR)* algorithm was used as a baseline in (Saveski et al. 2017). The algorithm clusters the unweighted graph using *reLDG* algorithm, assigns similar clusters to the same strata and assigns nodes in strata to treatment and control in a randomized fashion
- $\boldsymbol{CBR_{reLDG}}$ (Saveski et al. 2017): *Cluster-based Randomized assignment (CBR)* is the main algorithm proposed by (Saveski et al. 2017). The algorithm clusters the unweighted graph using *reLDG*, assigns similar clusters to the same strata and randomly picks clusters within the same strata as treatment or control.
- $\boldsymbol{CBR_{MCL}}$: A variant of $\boldsymbol{CBR}$ that we introduce for the sake of fairness which uses MCL for weighted-graph clustering.
- $\boldsymbol{Match}$: This algorithm matches nodes using maximum weighted matching algorithm and then randomly assigns nodes in each matched pair to treatment and control at random without considering clustering.
- $\boldsymbol{CMatch_{reLDG}}$: This method uses our $CMatch$ framework but works on an unweighted graph. It uses $reLDG$ for graph clustering.
- $\boldsymbol{CMatch_{MCL}}$: This is our proposed technique which uses MCL for weighted graph clustering.

$CMatch$ uses the $maximum\_weight\_matching$ function from the $NetworkX$ Python library.

Experimental setup

We run a number of experiments varying the underlying spillover assumptions, clustering algorithms, number of clusters, and node matching algorithms. Our experimental setup measures the desired properties for network experiment design, as described in Problem 1 and follows the experimental setups in existing work (Arbour et al. 2014; Eckles, Karrer, and Ugander 2017; Maier et al. 2013; Stuart 2010; Saveski et al. 2017).

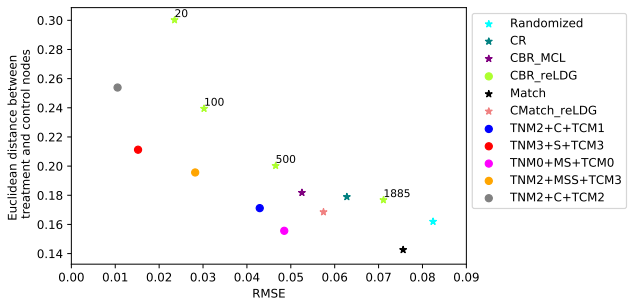

Figure 3: The tradeoff between selection bias (distance) and undesirable spillover (RMSE) in $CMatch_{MCL}$ variants (labeled with methods applied in) and baselines in Cora dataset for *e.p = edge-weight*; CBR_{reLDG} is annotated with the number of clusters

To measure the strength of interference bias in different estimators, we report on two metrics:

1. *Root Mean Squared Error* (RMSE) of the total treatment effect calculated as:

$$RMSE = \sqrt{\frac{1}{S}\sum_{s=1}^{S}((\hat{\tau}_s - \tau_s)^2)}$$

where S is the number of runs and τ_s and $\hat{\tau}_s$ are the true and estimated causal effect in run s, respectively. We set $S = 10$ in all experiments. Error can be attributed to undesired spillover only.

2. The number of edges and sum of edge weights between treatment and control nodes assigned by each algorithm.

To show the selection bias, we want to assess how different treatment vs. control nodes are. We compute the Euclidean distance between the attribute vector mean of treated and that of untreated nodes. We show the average and standard deviation over 10 runs.

We run all 115 possible combinations of CMatch options for node matching, cluster weights and cluster graph for each dataset. We consider four different values for the threshold α in **TNM**: 0 (**TNM0**), first (**TNM1**), second (**TNM2**) and third (**TNM3**) quantile of pairwise nodes' similarity distribution where $sim(v_i, v_j)$= (1- the normalized L_2 norm). For **TCM**, we consider four different β values: 0 (**TCM0**), first (**TCM1**), second (**TCM2**) and third (**TCM3**) quantile of the pairwise clusters' similarity distribution for each dataset. We use **TNM2 + C + TCM2** in all the experiments of $CMatch_{reLDG}$.

Unless otherwise specified, the number of clusters is the same for all CBR and CMatch versions based on the optimal determination by MCL as optimal for each respective dataset. The number of clusters determined by MCL is $2,497$ for $Citeseer$, $1,885$ for $Cora$, $1,056$ for $Hamsterster$ and 20 in *50 Women* dataset.

Results

Tradeoff between interference and selection bias in *CMatch* variants and baselines: Given the large number of $CMatch$ option combinations (115), we first find which ones of these combinations have a good tradeoff between RMSE and Euclidean distance (between treatment and control) with *e.p = edge-weight*. Based on these experiments, we notice that 1) methods with stricter cluster thresholds (**TCM2** and **TCM3**) tend to have lower error, 2) stricter node match thresholds (**TNM2** and **TNM3**) have lower error than others for **S** and **MSS** and 3) **MS** has high error across thresholds. Due to space constraints, we are showing detailed results for Cora only.

Fig. 3 shows the results for the $CMatch$ variants with the best tradeoffs and their better performance when compared to the baselines for Cora. Full $CMatch$ results can be found in Table 2. The figure clearly shows that the selection bias decreases at the expense of interference bias. For example, while the Euclidean distance for **TNM0+MS+TCM0** is low (0.155) when compared to **TNM2+C+TCM2** (0.253), its RMSE is higher, 0.048 vs. 0.01. The comparison between CBR_{reLDG} with different possible number of clusters is consistent with the tradeoff shown in Fig.1. CBR_{reLDG} with the highest error (annotated with 1885) and $CMatch_{MCL}$ have the same number of clusters. It is intuitive that the $Match$ method has the least selection bias, because all nodes have their best matches. However, similar to the $Randomized$ method, it suffers from high interference bias (RMSE) because of the high density of edges between treatment and control nodes.

Interference evaluation for contagion: We choose two $CMatch$ variants with low estimation errors: **TNM2 + MSS + TCM3** and **TNM2 + C + TCM2**, denoted by $CMatch_{MCL_{MSS}}$ and $CMatch_{MCL_C}$ respectively, and compare their causal effect estimation error with the baselines. The first method uses a simpler cluster weight assignment while the second one uses the expensive maximum weighted matching of nodes. Fig. 4 shows that both variants of $CMatch_{MCL}$ get significantly lower error than other methods, especially in Citeseer and Cora with 75.5% and 81.8% estimated error reduction in comparison to CBR_{reLDG} for *e.p = edge-weight*. $CMatch_{MCL_{MSS}}$ has higher error than $CMatch_{MCL_C}$ in most of the experiments which is expected as shown in Figure 3. $Randomized$ and $Match$ approaches have similar performance in all datasets because of their similarity in node randomization approach. We also notice that CBR_{reLDG} has the highest estimation error in Hamsterster data which confirms that clustering has a significant effect on the unallowable spillover. Meanwhile, $CMatch_{reLDG}$ outperforms other baselines in some datasets (Citeseer) and but not in others (Hamsterster and 50 Women). In Citeseer, the CR method gets the largest estimation error.

Fig. 4 also shows that the higher the unallowable spillover probability, the larger the estimation error but also the better our method becomes relative to the baselines. For example, by increasing the unallowable spillover probability from 0.1 to 0.5 in Citeseer, the estimation error increases from 0.005 to 0.02 for $CMatch_{MCL_C}$ and from 0.023 to 0.086 for CBR_{reLDG}.

Interference evaluation for direct interference: Fig. 5 shows the difference between RMSE of different estima-

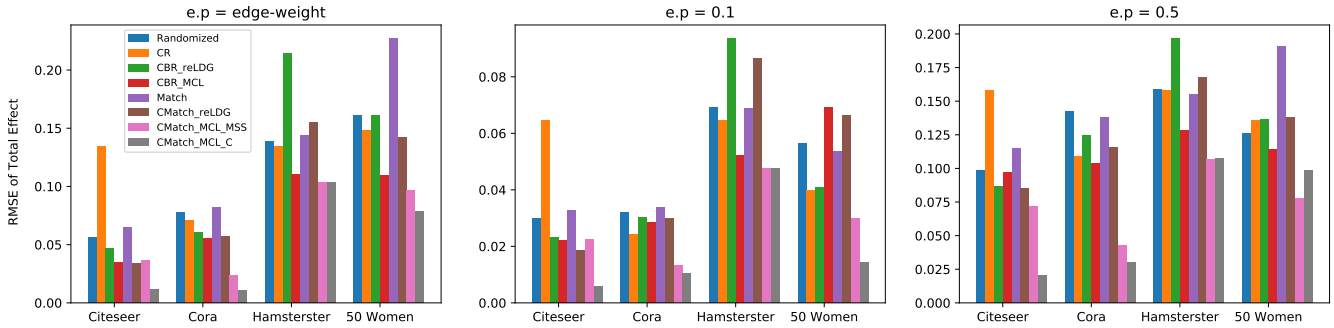

Figure 4: RMSE of total effect in the presence of contagion considering different unallowable spillover probabilities in all datasets; $CMatch_{MCL_C}$ achieves the lowest error in all datasets

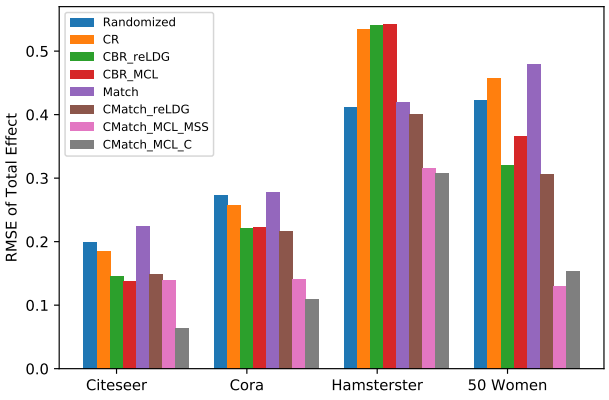

Figure 5: RMSE of total effect in the presence of direct interference (*e.p = edge-weight*). $CMatch_{MCL_C}$ and $CMatch_{MCL_{MSS}}$ obtain the lowest RMSE for all datasets.

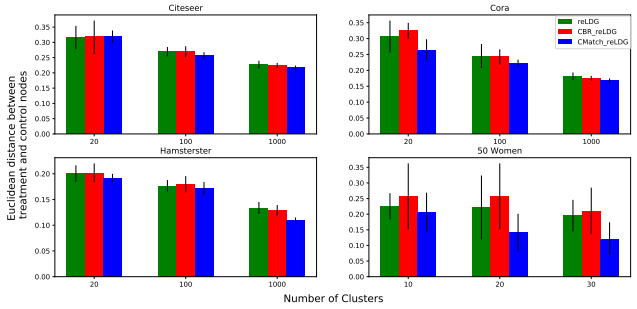

Figure 6: Euclidean distance between the attribute vector means of treatment and control nodes for different number of clusters. The higher the number of clusters, the lower the selection bias.

tors over the presence of direct interference for *e.p = edge-weight*. In four datasets, both variants of $CMatch_{MCL}$ get the lowest estimation error in comparison to baseline methods. For example, $CMatch_{MCL_C}$'s error is approximately half of the error of CBR_{reLDG} (0.06 vs. 0.13 for Citeseer, 0.1 vs. 0.22 for Cora, 0.31 vs. 0.54 for Hamsterster, 0.15 vs. 0.36 for 50 Women). Similar to contagion, $Match$ and $Randomized$ methods have similar estimation errors.

Potential spillover evaluation: Table 3 shows the potential spillover between treatment and control nodes assigned by different methods. This applies to both contagion and direct interference. $CMatch$ has the lowest sum of edges and edge weights between treatment and control nodes across all datasets. The difference between $CMatch_{MCL_C}$ and the baselines in Cora and Citeseer is substantial: $CMatch_{MCL_C}$ has between 13.5% and 34.8% lower number of edges between treatment and control across datasets.

Selection bias evaluation for contagion. In this experiment, we look at the relationship between number of clusters and the difference between treatment and control nodes with and without cluster matching. Fig. 6 shows the Euclidean distance between the average of treatment and control nodes' attributes in $CMatch_{reLDG}$, CBR_{reLDG} and $reLDG$ for three different number of clusters and unallowable spillover probability *e.p = edge-weight*. Since $CMatch_{reLDG}$ optimizes for selection bias directly, it is not surprising that it results in treatment and control nodes that have more similar feature distributions than the other two methods. In Citeseer the differences are more subtle than in the other datasets. Error bars show the variance of averages over 10 runs which confirms the low variance of estimations in all datasets except in 50 Women, which is a small dataset.

Sensitivity to spillover probability metrics. Our last experiment compares metrics for calculating the spillover probability, Cosine similarity, Jaccard similarity and the L2-based similarity used in all other experiments. We report on RMSE of total effect using $CMatch_{MCL_C}$ and $CMatch_{MCL_{MSS}}$ methods under contagion. Figure 7 shows that $CMatch_{MCL_C}$ with L2-based similarity obtains the least error in all datasets except for Citeseer where Cosine similarity has slightly lower error. For $CMatch_{MCL_{MSS}}$, Cosine similarity has the lowest RMSE in Citeseer and 50 Women dataset, while Euclidean similarity has the lowest error in the other datasets. Jaccard similarity has the highest estimation error in all almost all cases.

Table 2: The tradeoff between selection bias (distance) and undesirable spillover (RMSE) in $CMatch$ variants in Cora dataset. $CMatch_{MCL}$ variants used in Fig. 3 are in bold.

		TCM0		TCM1		TCM2		TCM3		GCM	
		RMSE	ED	RMSE	ED	RMSE	ED	RMSE	ED	RMSE	ED
C	TNM0	0.052	0.184	0.007	0.267	0.017	0.263	0.014	0.26	0.048	0.789
	TNM1	0.055	0.176	0.051	0.177	0.008	0.258	0.012	0.26	0.031	0.6
	TNM2	0.054	0.171	**0.042**	**0.171**	**0.01**	**0.253**	0.017	0.251	0.036	0.591
	TNM3	0.043	0.175	0.043	0.175	0.173	0.046	0.018	0.231	0.034	0.592
	BNM	0.012	0.262	0.037	0.481	0.049	0.485	0.059	0.479	0.025	0.274
S	TNM0	0.056	0.16	0.058	0.159	0.048	0.16	0.056	0.162	0.035	0.34
	TNM1	0.055	0.16	0.053	0.162	0.057	0.165	0.054	0.166	0.026	0.31
	TNM2	0.056	0.162	0.054	0.168	0.048	0.165	0.033	0.183	0.039	0.292
	TNM3	0.057	0.169	0.041	0.174	0.024	0.198	**0.015**	**0.211**	0.021	0.275
	BNM	0.014	0.253	0.017	0.264	0.02	0.27	0.027	0.303	0.014	0.277
MC	TNM0	0.049	0.177	0.015	0.261	0.01	0.262	0.008	0.263	0.042	0.189
	TNM1	0.055	0.173	0.052	0.174	0.01	0.257	0.012	0.253	0.040	0.191
	TNM2	0.047	0.171	0.051	0.177	0.013	0.261	0.007	0.263	0.024	0.211
	TNM3	0.047	0.173	0.049	0.178	0.051	0.176	0.011	0.249	0.012	0.244
	BNM	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
MS	TNM0	**0.048**	**0.155**	0.051	0.156	0.052	0.156	0.058	0.157	0.018	0.271
	TNM1	0.051	0.156	0.057	0.157	0.048	0.156	0.052	0.16	0.022	0.264
	TNM2	0.059	0.156	0.057	0.157	0.054	0.158	0.056	0.157	0.021	0.258
	TNM3	0.053	0.157	0.05	0.159	0.056	0.155	0.051	0.156	0.028	0.27
	BNM	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
MSS	TNM0	0.059	0.162	0.048	0.162	0.061	0.159	0.036	0.184	0.026	0.271
	TNM1	0.056	0.16	0.054	0.161	0.047	0.161	0.03	0.194	0.029	0.275
	TNM2	0.052	0.161	0.057	0.161	0.045	0.172	**0.028**	**0.195**	0.021	0.281
	TNM3	0.049	0.168	0.035	0.186	0.023	0.199	0.022	0.212	0.033	0.278
	BNM	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
E	N/A	0.051	0.178	0.05	0.18	0.031	0.203	0.012	0.242	0.042	0.718

Table 3: Percentage of edges (and edge weights) between treatment and control nodes. The lower the number, the lower probability of undesired spillover.

Dataset	$Randomized$	CR	CBR_{reLDG}	CBR_{MCL}	$Match$	$CMatch_{reLDG}$	$CMatch_{MCL_C}$
Citeseer	49.9%(50%)	35.9% (36.3%)	39.8% (38.4%)	38.9%(38.4%)	53.9% (56.6%)	35.8% (34.4%)	**7.5**%(7.2%)
Cora	49.7%(49.7%)	37.6%(37.6%)	43.4%(42.8%)	38.9% (33.6%)	51.8%(53.3%)	38.7% (38.2%)	**8.6**%(9.1%)
Hamsterster	50.2%(50.1%)	31.7%(30.4%)	48.3%(48.3%)	35.1% (34.7%)	50% (50.1%)	43.3% (44.4%)	**34.8**%(34.4%)
50 Women	48.5%(48.1%)	31.8%(30.5%)	36.6%(34.3%)	18.3%(11.4%)	52.5%(52.7%)	16%(18.6%)	**12.8**%(9.7%)

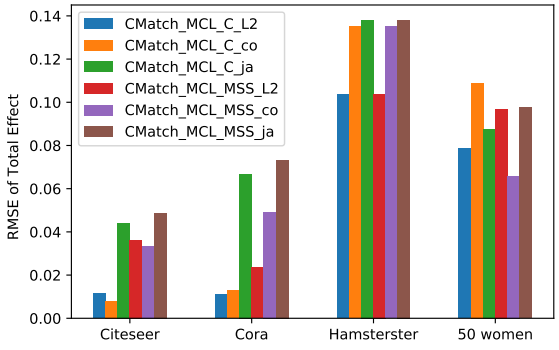

Figure 7: RMSE of total effect in the presence of contagion using three different similarity methods to calculate spillover probability: Cosine (co), Jaccard (ja) and L2 similarity.

Conclusion

We presented $CMatch$, the first optimization framework that minimizes both interference and selection bias in cluster-based network experiment design. We demonstrated the tradeoff between causal effect estimation error and distance between treatment and control groups, as well as the value of combining weighted graph clustering with cluster matching. Our experiments on four real-world network datasets showed that $CMatch$ reduces the causal effect estimation error by 8.6% to 91.4% when compared to state-of-the-art techniques. Some possible extensions of our framework include understanding the impact of network structural properties on estimation, jointly optimizing for interference and selection bias, and developing frameworks that are able to mitigate for multiple-hop diffusions.

References

Arbour, D. T.; Marazopoulou, K.; Garant, D.; and Jensen, D. D. 2014. Propensity score matching for causal inference with relational data. In *UAI Workshop on causal inference*, 25–34.

Backstrom, L., and Kleinberg, J. 2011. Network bucket testing. In *WWW*.

Basse, G., and Feller, A. 2018. Analyzing two-stage experiments in the presence of interference. *Journal of the American Statistical Association* 113(521):41–55.

Duan, R., and Pettie, S. 2014. Linear-time approximation for maximum weight matching. *J. ACM* 61(1):1:1–1:23.

Eckles, D.; Karrer, B.; and Ugander, J. 2017. Design and analysis of experiments in networks: Reducing bias from interference. *Journal of Causal Inference*.

Enright, A. J.; van Dongen, S.; and Ouzounis, C. A. 2002. An efficient algorithm for large-scale detection of protein families. *Nucleic acids research* 30 7:1575–84.

Gilbert, E., and Karahalios, K. 2009. Predicting tie strength with social media. In *CHI*, 211–220. ACM.

Gui, H.; Xu, Y.; Bhasin, A.; and Han, J. 2015. Network a/b testing: From sampling to estimation. In *WWW*.

Hamilton, W. L.; Ying, R.; and Leskovec, J. 2017. Representation learning on graphs: Methods and applications. *TKDE*.

Harvey, N. J. A. 2009. Algebraic algorithms for matching and matroid problems. *SIAM J. Comput.* 39(2):679–702.

Imbens, G., and Rubin, D. 2015. *Causal Inference in Statistics, Social and Biomedical Sciences: An Introduction*. Cambridge Univ Press.

Kohavi, R.; Deng, A.; Frasca, B.; Walker, T.; Xu, Y.; and Pohlmann, N. 2013. Online controlled experiments at large scale. In *KDD*.

Lee, S., and Honavar, V. 2016. On Learning Causal Models from Relational Data. In *AAAI*, 3263–3270.

Li, L.; Chu, W.; Langford, J.; and Schapire, R. E. 2010. A contextual-bandit approach to personalized news article recommendation. In *WWW*.

Liben-Nowell, D., and Kleinberg, J. 2007. The link-prediction problem for social networks. *JASIST* 58(7):1019–1031.

Maier, M.; Taylor, B.; Oktay, H.; and Jensen, D. 2010. Learning causal models of relational domains. In *AAAI*.

Maier, M.; Marazopoulou, K.; Arbour, D.; and Jensen, D. 2013. A sound and complete algorithm for learning causal models from relational data. In *UAI*.

Marazopoulou, K.; Maier, M.; and Jensen, D. 2015. Learning the structure of causal models with rel. and temporal dependence. In *UAI*.

Michell, L., and Amos, A. 1997. Girls, pecking order and smoking. *Social Science & Medicine* 44(12):1861 – 1869.

Milo, R.; Shen-Orr, S.; Itzkovitz, S.; Kashtan, N.; Chklovskii, D.; and Alon, U. 2002. Network motifs: simple building blocks of complex networks. *Science* 298(5594):824–827.

Mucha, M., and Sankowski, P. 2004. Maximum matchings via gaussian elimination. In *45th Annual IEEE Symposium on Foundations of Computer Science*, 248–255.

Nishimura, J., and Ugander, J. 2013. Restreaming graph partitioning: Simple versatile algorithms for advanced balancing. In *KDD*.

Ogburn, E. L.; VanderWeele, T. J.; et al. 2014. Causal diagrams for interference. *Statistical science* 29(4):559–578.

Oktay, H.; Taylor, B.; and Jensen, D. 2010. Causal discovery in social media using quasi-experimental designs. In *KDD Workshop on Social Media Analytics*.

Pearl, J. 2009. *Causality*. Cambridge Univ Press.

Pouget-Abadie, J.; Mirrokni, V.; Parkes, D. C.; and Airoldi, E. M. 2018. Optimizing cluster-based randomized experiments under monotonicity. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery Data Mining*, KDD '18, 2090–2099. New York, NY, USA: Association for Computing Machinery.

Rattigan, M. J.; Maier, M. E.; and Jensen, D. D. 2011. Relational blocking for causal discovery. In *AAAI*.

Rossi, R.; McDowell, L.; Aha, D.; and Neville, J. 2012. Transforming graph data for statistical relational learning. *JAIR* 45:363–441.

Saveski, M.; Pouget-Abadie, J.; Saint-Jacques, G.; Duan, W.; Ghosh, S.; Xu, Y.; and Airoldi, E. 2017. Detecting network effects: Randomizing over randomized experiments. In *KDD*.

Schaeffer, S. E. 2007. Graph clustering. *Computer science review* 1(1):27–64.

Sen, P.; Namata, G. M.; Bilgic, M.; Getoor, L.; Gallagher, B.; and Eliassi-Rad, T. 2008. Collective classification in network data. *AI Magazine* 29(3):93–106.

Stuart, E. A. 2010. Matching methods for causal inference: A review and a look forward. *Statistical science* 25(1):1.

Toulis, P., and Kao, E. 2013. Estimation of causal peer influence effects. In *ICML*, 1489–1497.

Ugander, J.; Karrer, B.; Backstrom, L.; and Kleinberg, J. 2013. Graph cluster randomization: Network exposure to multiple universes. In *KDD*.

Varian, H. 2016. Causal inference in economics and marketing. *PNAS*.

Yang, J., and Leskovec, J. 2015. Defining and evaluating network communities based on ground-truth. *KAIS* 42(1):181–213.

Zheleva, E.; J.; Kuter, U.; and Getoor, L. 2008. Using friendship ties and family circles for link prediction. *SNAKDD*.

Zhou, Y.; Cheng, H.; and Yu, J. X. 2009. Graph clustering based on structural/attribute similarities. *Proc. VLDB Endow.* 2(1):718–729.