Research Article

Kevin Han* and Johan Ugander

Model-based regression adjustment with model-free covariates for network interference

https://doi.org/10.1515/jci-2023-0005
received February 09, 2023; accepted July 26, 2023

Abstract: When estimating a global average treatment effect (GATE) under network interference, units can have widely different relationships to the treatment depending on a combination of the structure of their network neighborhood, the structure of the interference mechanism, and how the treatment was distributed in their neighborhood. In this work, we introduce a sequential procedure to generate and select graph- and treatment-based covariates for GATE estimation under regression adjustment. We show that it is possible to simultaneously achieve low bias and considerably reduce variance with such a procedure. To tackle inferential complications caused by our feature generation and selection process, we introduce a way to construct confidence intervals based on a block bootstrap. We illustrate that our selection procedure and subsequent estimator can achieve good performance in terms of root-mean-square error in several semi-synthetic experiments with Bernoulli designs, comparing favorably to an oracle estimator that takes advantage of regression adjustments for the known underlying interference structure. We apply our method to a real-world experimental dataset with strong evidence of interference and demonstrate that it can estimate the GATE reasonably well without knowing the interference process *a priori*.

Keywords: causal inference with interference, SUTVA, A/B testing, regression adjustment, social network analysis

MSC 2020: 62D10, 62P25

1 Introduction

In standard experiments, researchers typically assume that one unit's assignment does not affect another unit's response; this is usually referred to as the no interference assumption [1, Chapter 2] or the stable unit treatment value assumption (SUTVA) [2]. However, when experimental units interact with each other, SUTVA is often untenable. Violation of SUTVA has been found in many applications, including politics [3], education [4,5], economics [6,7], and public health [8]. Recently, technology companies developing products with social or market interactions have developed methods to manage the considerable interference in their product experiments [9–11]. In practice, researchers look for an underlying structure that limits the scope of interference and estimation of causal effects proceeds from assuming the structure. Aronow and Samii [12] propose to use a lower dimensional representation of the interference mechanism and estimate causal effects accordingly. In the no-interference literature, regression adjustment has shown to be effective in both theory [13] and practice [14]. Chin [15] considers regression adjustment under interference when assuming a linear model for the

*** Corresponding author: Kevin Han**, Department of Statistics, Stanford University, Stanford, California, United States, e-mail: kevinwh@stanford.edu
Johan Ugander: Department of Management Science and Engineering, Stanford University, Stanford, California, United States, e-mail: jugander@stanford.edu

outcomes, and estimates the parameters of the model from the experimental data. Such a linear model assumption is not uncommon and has also been studied in design of experiments [16] and interference detection [10]. There has also been literature on new designs that tackle the complication of interference. For example, Ugander et al. [17] and Ugander and Yin [18] consider (randomized) cluster randomized designs that effectively account for interference by doing randomization on cluster level instead of unit level.

In this article, we provide a procedure to estimate the global average treatment effect (GATE) by using regression adjustment without assuming the true set of features as given by Chin [15]. We generate the features for adjustment based on observed experimental data in a model-free manner. As an outline for this work, we first give preliminaries of the problem setup and motivate our method through a study of the classic linear-in-means model in econometrics. We then provide our general procedure to generate model-free covariates based on the observed experimental data. Finally, we show how to do estimation and inference for the GATE with model-free covariates. We conclude with simulations, an empirical application, and discussions.

1.1 Related work

The present work is most closely related to the literature on estimating causal effects in the presence of interference. Interference introduces complexities in identifying units that are "equivalently treated," unlike scenarios under SUTVA, where every unit can share the same treatment condition, regardless of exposure to global treatment or control. As a result, all existing methods rely on making certain assumptions about the interference structure. A primary approach in this area revolves around defining exposure mappings [12] and subsequently estimating the causal effect [17,19,20]. For instance, when the experimental units are linked through a social network, it might be assumed that only a unit's direct neighbors are of significance. In such a situation, the Horvitz–Thompson estimator is often utilized to estimate average outcomes under specific exposure values [12]. Despite obtaining an unbiased estimator, its variance scales inversely with the probability of a unit's complete neighborhood being entirely in either the treatment or control group. As a result, the variance is exponentially dependent on the degrees of nodes, making the estimator impractical for use in settings with complex real-world interference networks. Hence, another line of work focuses on introducing functional assumptions on potential outcomes. For instance, in the study by Cai et al. [21], insurance adoption is modeled by a linear model incorporating the assignment vector and network statistics/features. Similar ideas are also examined in previous studies [15,22,23]. Though regression adjustment offers estimates with low variance, it relies on selecting the appropriate features for the linear model. In contrast to those prior works, we do not assume *a priori* knowledge of the oracle set of features. We adapt the ReFeX framework [24] from the literature on graph mining to generate candidate features. We modify the standard feature selection procedure from ReFeX, replacing the standard heuristic with a lease absolute shrinkage and selection operator (LASSO)-based procedure, making it possible to align the favorable feature generation abilities of ReFeX with known results for inference in LASSO literature, allowing us to more reliably estimate causal effects.

Finally, our work is also related to the literature on block bootstrap. Due to the absence of a closed-form expression for the variance of our estimator, we employ bootstrap methods, as classically introduced by Efron [25], to construct confidence intervals. The block bootstrap is particularly useful in our case as it addresses the issue of sample correlation arising from the network structure, a scenario where the standard bootstrap approach falls short. Block bootstrap was designed for bootstrapping time series data [26–28], primarily addressing serial correlations within the data. More recently, Kojevnikov [29] studied the use of the bootstrap for network-dependent processes to construct confidence intervals for the means of network-dependent processes. However, none of these existing works can be applied directly to our setup, given that our approach incorporates both feature selection and modeling components. As such, we have developed a customized block bootstrap method, specifically tailored for use in estimating causal effects after feature selection.

2 Setup

Consider a randomized experiment on n units where there is a simple undirected graph $G = (V, \mathcal{E})$ that describes the social network of interactions among n units. The graph G is associated with a symmetric matrix $A \in \mathbb{R}^n$ so that $A_{ij} = 1$ if $(i, j) \in \mathcal{E}$ and zero otherwise. Let $\mathcal{N}_i^{(k)}$ denote the k-hop neighborhood around each node $i \in V$. We omit the superscript when $k = 1$ and let d_i denote the degree of each node (or equivalently, $d_i = |\mathcal{N}_i|$). We denote by W_i the random assignment and $x_i \in \mathcal{X}$ the pretreatment covariates for unit i. We assume that the experimental population is the population of interest and hence view pretreatment covariates as fixed. We only consider binary treatments but note that extensions to nonbinary treatments are straightforward. Throughout, we use lower case letters with the appropriate subscript for realizations of the random variables and for nonrandom quantities.

We work under the Rubin causal model [2,30,31]. For every unit i, we associate it with potential outcomes $Y_i(w) \in \mathbb{R}$ for $w \in \{0, 1\}^n$. In our analysis, we treat the potential outcomes as random variables. Diverging from the super-population perspective, we do not consider the potential outcomes as independent and identically distributed (i.i.d.) samples drawn from a super-population. As a result, each unit might exhibit distinct expected outcomes under global treatment, $\mathbb{E}[Y_i(\mathbf{1})]$, and global control, $\mathbb{E}[Y_i(\mathbf{0})]$. This finite population perspective is usually preferred in both social science and technology industry practitioners, as people find themselves willing to think of the outcomes as random rather than having a super-population in mind; the effect people care about is the effect for the specific population being studied. We are interested in the following causal estimand that we call the GATE:

$$\tau = \frac{1}{n}\sum_{i=1}^{n}\mathbb{E}[Y_i(\mathbf{1}) - Y_i(\mathbf{0})]. \tag{1}$$

Here, $\mathbf{1}$ denotes the n-dimensional ones vector and similarly for $\mathbf{0}$. The GATE estimand, also known as the total treatment effect (TTE) in the study by Yu et al. [32], measures the overall effect of the intervention on the experimental units. Under SUTVA, the assignments of other units would not affect one's response, and hence, there are only two potential outcomes per unit, $Y_i(0)$ and $Y_i(1)$. Under SUTVA, the GATE is then simply the average treatment effect (ATE). When there is interference along a network, there may be up to 2^n different potential outcomes per unit. In the absence of further assumptions, it is impossible to observe $Y_i(\mathbf{1})$ for some unit i and also observe $Y_j(\mathbf{0})$ for any other unit j.

In this work, we take a regression perspective and assume two functions f_0 and f_1 such that for each unit i and each assignment vector $w \in \{0, 1\}^n$,

$$Y_i(w) = w_i f_1(i, w, x_i, G) + (1 - w_i) f_0(i, w, x_i, G) + \varepsilon_i, \tag{2}$$

with ε_i's being exogenous, i.e. $\mathbb{E}[\varepsilon_i|w] = 0$. The functions f_0 and f_1 each take as input the node label i, the assignment vector w, the covariate vector x_i, and graph G. This approach uses exposure mappings [12] as functions that map an assignment vector w and x_i to a specific exposure value so that if two assignment vectors w and w' induce the same exposure value for a unit, then they have the same value of potential outcome. Since the potential outcomes only depend on the exposure values, we can view them as a function of exposure values and we can rewrite the potential outcomes as in (2). Given (2), since functions f_1 and f_0 are shared across all units, we can use the treated units to estimate f_1 and control units to estimate f_0. Suppose $\hat{f}_0$ and $\hat{f}_1$ are two estimates of f_0 and f_1, respectively, then a natural estimator of the GATE would be

$$\hat{\tau} = \frac{1}{n}\sum_{i=1}^{n}[\hat{f}_1(i, \mathbf{1}, x_i, G) - \hat{f}_0(i, \mathbf{0}, x_i, G)].$$

Unfortunately, estimation of the GATE will be impossible without any further assumptions on the structure of the functions f_0 and f_1[1]. To motivate our structural assumptions on f_0 and f_1, we look at the following example.

1 Basse and Airoldi [33] have a discussion from an inference perspective.

Example 1. (Linear-in-means model) Consider the structural model [34–36]

$$\mathbf{y} = \alpha\mathbf{1} + \beta\tilde{A}\mathbf{y} + \gamma\mathbf{w} + \delta\tilde{A}\mathbf{w} + \boldsymbol{\varepsilon}, \quad \mathbb{E}[\boldsymbol{\varepsilon}|\mathbf{w}] = 0, \tag{3}$$

where $\mathbf{y}$ is the $n \times 1$ outcome vector, $\tilde{A}$ is the degree-normalized adjacency matrix, i.e., $\tilde{A}_{ij} = A_{ij}/d_i$, $\mathbf{w}$ is the assignment vector, and $(\alpha, \beta, \gamma, \delta)$ are parameters. Bramoullé et al. [36] show that under some mild conditions on the coefficients and the graph G, we can rewrite the aforementioned model as follows:

$$\mathbf{y} = \alpha/(1-\beta)\mathbf{1} + \gamma\mathbf{w} + (\gamma\beta + \delta)\sum_{j=0}^{\infty}\beta^j\tilde{A}^{j+1}\mathbf{w} + \sum_{j=0}^{\infty}\beta^j\tilde{A}^{j+1}\boldsymbol{\varepsilon}. \tag{4}$$

Note that now the outcome is linear in the assignment vector $\mathbf{w}$ as well as $\{\tilde{A}^{j+1}\mathbf{w}\}_{j=0}^{\infty}$. Let $f_0(i, w, x_i, G) = f_1(i, w, x_i, G) = \alpha/(1-\beta) + \gamma w_i + (\gamma\beta + \delta)\sum_{j=0}^{\infty}\beta^j\tilde{A}^{j+1}w$ and notice that $\mathbb{E}[\sum_{j=0}^{\infty}\beta^j\tilde{A}^{j+1}\boldsymbol{\varepsilon}|w] = 0$. Thus, the linear-in-means model (3) can be written in the form of (2).

While in this example the linear model is infinite-dimensional, the linear structure of (4) motivates us to look at linear models for both f_0 and f_1. To make it formal, we make the following definition:

Definition 2.1. (Linear interference) We say that the model $\mathcal{Y} = \{Y_i(w) : w \in \{0,1\}^n, i \in [n]\}$ exhibits *linear interference* if there exists a function $g : [n] \times \{0,1\}^n \times \mathcal{X} \times \mathcal{G} \to \mathbb{R}^K$ and $\theta_0 \in \mathbb{R}^K$, $\theta_1 \in \mathbb{R}^K$ such that $f_0(i, w, x_i, G) = \theta_0^T g(i, w, x_i, G)$ and $f_1(i, w, x_i, G) = \theta_1^T g(i, w, x_i, G)$. We call each coordinate function g_j of g a *feature* of the interference.

Despite the simplicity of linear interference, from a graph perspective, it can be shown that convolutions on graphs can be well approximated by linear expansion [37]. Such a linear interference assumption is not uncommon [10,14,15]. Chin [15] shows how to do inference once we have access to the oracle g, while, [10] give a testing procedure to detect network interference under linear interference. Moreover, because we are interested in the quality of our estimated functions $\hat{f}_0$ and $\hat{f}_1$ for (only) $w = \mathbf{0}, \mathbf{1}$, we are effectively attempting generalization. Simple models usually generalize well [38,39], and thus, linear interference provides credibility of inference without losing flexibility in a world, where g can be arbitrarily complex.

Before proceeding, we can simplify (2) somewhat. Note that

$$\begin{aligned} Y_i(w) &= w_i f_1(i, w, x_i, G) + (1 - w_i) f_0(i, w, x_i, G) + \varepsilon_i \\ &= w_i f_1(i, w^{(i\to 1)}, x_i, G) + (1 - w_i) f_0(i, w^{(i\to 0)}, x_i, G) + \varepsilon_i \\ &= w_i \tilde{f}_1(i, w^{(-i)}, x_i, G) + (1 - w_i)\tilde{f}_0(i, w^{(-i)}, x_i, G) + \varepsilon_i, \end{aligned} \tag{5}$$

where $w^{(i\to t)}$ denotes the $n$dimensional vector that replaces w_i by t and $\tilde{f}_t$ is a function of i, $w^{(-i)}$, x_i and G only. Therefore, without loss of generality, we assume that the domain of g and hence the domain of f_0 and f_1 is $[n] \times \{0,1\}^{n-1} \times \mathcal{X} \times \mathcal{G}$.

From here on, for presentational simplicity, we will omit the pretreatment covariates x_i in our discussion. Extensions to the case of including pretreatment covariates will be discussed when not obvious. As a result, g is a function of the node label i, the assignment vector w, and the graph G only.

We focus on design that satisfies the following uniformity assumption:

Assumption 2.2. (Uniformity) We assume that W_i's are independent and $\forall i, \mathbb{P}(W_i = 1) = p_i$ for some $0 < p_i < 1$.

We make this assumption to follow the common practice of using Bernoulli randomization in network experiments, e.g., Karrer et al. [11]. As an alternative, estimates from designs that accounts for network interference (for example, graph cluster randomization) may suffer from sizable variance [18]. Hereinafter, we assume that W_i's are i.i.d. Bernoulli(p) random variables with $0 < p < 1$, i.e., we work with data from experiments under a Bernoulli design.

If we know the function g *a priori*, Chin [15] provides a complete solution. However, if we do not know the function g, then there are three significant challenges, all of which we address in this work. First, how should

we *construct* g so that the one we construct approximates the true one? Second, suppose we have many candidate functions then how should we *select* among them? Third, even if we have satisfactory answers to the first two questions, how should we do inference? We will address the first two challenges in the next section and the third challenge later.

3 Model-free covariates

Now by (5), the function g from Definition 2.1 takes node label i, $w^{(-i)}$, and G as input, and outputs a K-dimensional vector, what g essentially does is to produce K covariates based on $w^{(-i)}$ and G for each unit i. In this section, we describe a sequential procedure to generate and select model-free covariates. A high-level description of our method would be that we generate rich candidate features based solely on the graph structure as well as the assignment vector and select among these features based on the observed outcomes. We first give the procedure in Algorithm 1 and then explain the steps in more detail. We call the procedure ReFeX-LASSO as it builds on the graph mining technique ReFeX [24] to generate candidate features while using LASSO [40] to select features.

Algorithm 1. ReFeX-LASSO

Input: Graph $G = (V, \mathcal{E})$, assignment vector $w \in \{0, 1\}^n$, maximum number of iterations T.
Output: A set of covariates S.
1: Initialize $S = \{\}$, active feature set $A = \{\}$.
2: For each node/unit i, construct m base features and add m base features to A.
3: **for** $t = 1$ to T **do**
4: Regress y on w and features from S and A using LASSO with no penalty on features from S.
5: If no feature in A is selected, return S. Otherwise, add selected features from A to S.
6: Recursively construct features by performing aggregations of features in A over neighbors in 1-hop neighborhood.
7: Delete old features in A and add those new features to A.
8: **end for**
9: Return S.

ReFeX (Recursive Feature eXtraction) was originally designed to generate features for graph mining tasks and can be viewed as a recursive algorithm that starts with base features of each node in the graph and iteratively (i) adds and (ii) prunes features based on aggregations over features from neighboring nodes. ReFeX can be viewed as a simple early precursor to recent methods for graph representation learning based on graph convolution networks [41,42]. We adopt the feature generation step in ReFeX algorithm, but replace the feature pruning part of the original algorithm by LASSO, a modification that allows us to more precisely characterize the features that are available at any given step of the algorithm.

ReFeX has two ingredients – base features and aggregation functions. Given w, $\{x_i\}_{i=1}^n$, and G, base features are those features that can be constructed by only looking at each node's 1-hop neighborhood. They can be arbitrary as long as they satisfy this local look-up constraint. Base features can be purely graph features like degree, centrality, clustering coefficient, etc. They can also be pretreatment covariates x_i. Often we would also like to have base features that depend on not just one input of the function g but features computed from two inputs of g. For example, features like the number of treated neighbors, which depends on both the assignment vector w as well as the graph G, or the average feature value over all neighbors, which depends on the pretreatment covariates and G. With ReFeX, the base features are chosen by the analyst. The selection of base features can often be informed by domain-specific knowledge. For instance, in a scenario where treatment is favored and peer effects positively influence the units, it could be reasonably proposed that an increased

number or proportion of treated neighbors associates with a higher outcome. Consequently, a logical choice for base features might be the number or fraction of treated neighbors. Moreover, in other scenarios where demographic attributes may play a significant role, it may also be appropriate for the analyst to incorporate these demographic features as base features. Aggregation functions are functions that take features from neighboring nodes as inputs and output a single value. Hence, one aggregation function essentially computes a statistic based on the sample of feature values from neighbors. The aggregation functions again can be arbitrary and chosen by the analyst. Some common examples include min, max, sum, mean, and variance [24].

We are now ready to introduce the ReFeX-LASSO algorithm. The ReFeX-LASSO algorithm starts with two empty feature sets, the target set S and the active feature set A. The first set S stores the selected features, and features in S will be used for adjusting the GATE estimate. The active feature set A contains features that were recursively added in the previous step and yet to be selected. At the beginning of the procedure, we construct base features for each unit i. Equipped with a set of base features, each time we regress the outcome vector y on features from both set S and set A using LASSO. The LASSO regularization parameter can be chosen by cross-validation, and hence, we do not need extra hyper-parameters of the algorithm. Note that we do not put a penalty on features in S since they have already been selected and should be kept. The intuition behind this step is that in general features generated later (pulling information from farther in the graph) should not be more predictive than features selected previously. Next, depending on the number of newly selected features, we either terminate the construction and return the current S or add those selected features to S and proceed with the recursive construction. We then need to generate new features and add them to A. To do so, we now perform aggregations on old features over all neighboring units. Finally, we add those features to A and delete all old features in A.

The maximum number of iterations in Algorithm 1 limits the distance in the graph that we can pull information from. Although each step only performs aggregations over neighbors in the 1-hop neighborhood, by repeatedly performing the aggregations, we are able to construct features that are informative for the k-hop neighborhood. To illustrate this point, we give an example.

Example 2. (ReFeX and multi-hop information) Suppose one of the base features we use in ReFeX-LASSO is the fraction of treated neighbors,

$$\rho_i = \frac{1}{d_i} \sum_{j \in \mathcal{N}_i} w_j,$$

and supposed we limit ourselves to mean aggregation, i.e., we look at each unit's neighbors and aggregate their fraction of treated neighbors using a mean function. We call this new feature $\tilde{\rho}_i$. We then have that

$$\begin{aligned}
\tilde{\rho}_i &= \frac{1}{d_i} \sum_{j \in \mathcal{N}_i} \rho_j \\
&= \frac{1}{d_i} \sum_{j \in \mathcal{N}_i} \frac{1}{d_j} \sum_{k \in \mathcal{N}_j} w_k \\
&= \sum_{j=1}^{n} \frac{A_{ij}}{d_i} \sum_{k=1}^{n} \frac{A_{jk}}{d_j} w_k \\
&= \sum_{j=1}^{n} \tilde{A}_{ij} \sum_{k=1}^{n} \tilde{A}_{jk} w_k \\
&= [\tilde{A}^2 w]_i,
\end{aligned}$$

where A and $\tilde{A}$ are the same as defined in the linear-in-means model example from (3). The identical set of features utilized by the linear-in-means model is obtained by the aforementioned recursive process. Note that the summand is 1 if and only if A_{ij}, A_{jk}, and w_k are all 1s. In other words, if we ignore the normalizing terms, the sum essentially represents the number of length-2 paths in G that start at unit i and arrive at a treated unit. With the normalizing terms, it is close to the fraction of such paths among all length-2 paths that start at unit i. Clearly, this feature is informative for unit i's 2-hop neighborhood.

The aforementioned example shows the power of recursion. It allows us to have access to information about much larger neighborhoods without actually looking up all units in larger neighborhoods. In fact, the ReFeX component of ReFeX-LASSO is very efficient in terms of computational complexity [24], making the procedure ideal for large-scale experiments on online platforms where network interference is ubiquitous. Another advantage of our algorithm is that all the covariates generated are model-agnostic or model-free – we do not generate them according to any particular response model (or graph model). Since the aggregation functions are arbitrary, ReFeX can quickly generate a very large number of features, even for modest iterations budgets T. Despite the fraction of treated neighbors we just saw, we are also able to obtain the number of treated neighbors for each unit by using sum as the aggregation function. In general, using more complicated aggregation functions yields more complicated features. Thus, the recursive step offers rich features for each unit.

With minor modifications, we can see that all pruning steps in our procedure can be grouped together and done *ex ante*, i.e., before running the experiment and observing the outcomes. Then, after the experiment, we use the observed outcomes to select covariates among all the covariates we have generated. This method has certain advantages, so for completeness, we give such a modified version of ReFeX-LASSO in Algorithm 2, calling it post-ReFeX-LASSO.

Algorithm 2. Post-ReFeX-LASSO

Input: Graph $G = (V, \mathcal{E})$, assignment vector $w \in \{0, 1\}^n$, maximum number of iterations T.
Output: A set of covariates S.
1: Initialize $S = \{\}$.
2: For each node/unit i, construct m base features and add m base features to S.
3: **for** $t = 1$ to T **do**
4: Recursively construct features by performing aggregations of features in S that were added in the previous iteration over neighbors in 1-hop neighborhood.
5: Add those newly constructed features to S.
6: **end for**
7: Regress y on w as well as features from S using LASSO.
8: Keep selected features in S and remove other features from S.
9: Return S.

An operational advantage of post-ReFeX-LASSO is that two parts of the algorithm, feature generation and selection, can be done separately. However, in practice, we find that post-ReFeX-LASSO leads to estimates with larger variance. Our explanation for this increased variance is twofold. First, since the number of features generated from ReFeX may be large, separating the generation step and the selection step seems to make the selection step unstable. Second, many of the features generated along the way of post-ReFeX-LASSO are correlated, and including all of them simultaneously leads to greater uncertainty in terms of features being selected. Hence, it leads to estimates with larger variance, and we recommend ReFeX-LASSO over post-ReFeX-LASSO in all use cases when operationally feasible.

4 Inference with model-free covariates

In the previous section, we gave a sequential procedure that outputs a set of covariates S that can be used for regression adjustments when estimating GATEs. This section devotes to inference with model-free covariates. We first discuss how to use model-free covariates returned from ReFeX-LASSO or post-ReFeX-LASSO to do regression adjustment. Following that, we show one selection property of ReFeX-LASSO. We then give

theoretical properties of regression adjustment estimator of the GATE using model-free covariates as well as a simple way to construct confidence interval for τ.

4.1 Estimation

Let $u_i^1,\ldots, u_i^K$ denote the K covariates returned by ReFeX-LASSO or post-ReFeX-LASSO for unit i, and let $u_i = [u_i^1, \ldots, u_i^K]^T \in \mathbb{R}^K$ be the whole feature vector for unit i. We further let $\hat{g}$ be the function that maps (i, w, x_i, G) to u_i for each unit i. Finally, we denote by n_c the number of control units and n_t the number of treated units with $n_c + n_t = n$.

To estimate the GATE, we fit two linear models on control and treated units using u_i's. Ideally, we hope that there exist vectors β_0, β_1 such that $\beta_0^T u_i$ and $\beta_1^T u_i$ are good approximations of f_0 and f_1. To be specific, we first run an ordinary least squares with observations that are from the control group only and obtain $\hat{\beta}_0$. We then run ordinary least squares again, but now with observations that are from treatment group only and obtain $\hat{\beta}_1$. Meanwhile, the features u_i are all features under the treatment assignment w for which the responses were collected. To estimate the GATE, we are interested not in the response under u_i as it was, but u_i as it would be if $w = \mathbf{0}$ or $w = \mathbf{1}$. We thus pass $\mathbf{0}$ and $\mathbf{1}$ to $\hat{g}$ to obtain the feature vectors u_i^{gc} and u_i^{gt} under global control and global treatment, respectively.

Combining the coefficient estimates $\hat{\beta}_1$ and $\hat{\beta}_0$ with the vectors u_i^{gc} and u_i^{gt}, our estimate of the GATE is then simply

$$\hat{\tau} = \frac{1}{n}\sum_{i=1}^{n}(\hat{\beta}_1^T u_i^{gt} - \hat{\beta}_0^T u_i^{gc}). \tag{6}$$

Though assuming a linear model is restrictive, as we discussed previously, if we are able to generate predictive features then the linear model can be a good approximation to the true model. ReFeX-LASSO or post-ReFeX-LASSO help us choose good features to adjust for and thus both reduce the variance of the estimate[2] and reduce the bias we typically incur when ignoring interference.

4.2 Selection properties

Before we delve into inference details, we first discuss selection properties of ReFeX-LASSO, drawing inspiration from the prior work on sequential LASSO [43]. To this end, we introduce some additional notation. For each iteration t, let $\{u_1^t, u_2^t, \ldots, u_{i_t}^t\}$ be the set of features generated in the ReFeX step of ReFeX-LASSO and s_{*t} be the selected features at the tth iteration (note that s_{*t} may contain features that were selected in previous iterations and thus are not in the set $\{u_1^t, u_2^t, \ldots, u_{i_t}^t\}$). Moreover, we let $\mathcal{R}(s)$ to denote the space spanned by features in s.

Proposition 4.1. *For $t \geq 1$ and any $j \in \{1, \ldots, i_{t+1}\}$, if $u_j^{t+1} \in \mathcal{R}(s_{*t})$, then $j \notin s_{*(t+1)}$.*

This first proposition implies two things. First, we have a full rank design matrix at each iteration. Second, the subsequent selection will disregard the features that are highly correlated with the existing ones and hence provides intuition for why the post-ReFeX-LASSO leads to estimate with high variance. Without the sequential procedure of (non-post-) ReFeX-LASSO, two highly correlated features may enter the selection stage together.

Proposition 4.2. *Our selection is nested in the sense that $s_{*1} \subseteq s_{*2} \subseteq \cdots \subseteq s_{*T}$.*

2 In fact, in the case of no interference, Lin [13] shows that doing linear adjustment can only improve the precision.

This second proposition is relatively self-explanatory and ensures that the sequential procedure actually provides nested feature sets, i.e., by excluding penalties on selected features, we are able to keep them in our feature set S. Though our selection procedure in ReFeX-LASSO is quite different from sequential LASSO [43], and the proofs of the aforementioned two propositions are analogous to those in ref. [43]. There are two key differences between our selection procedure and sequential LASSO. First, instead of keeping all the features for every iteration, we throw away nonselected features in previous iterations. Second, the features under consideration at each iteration are newly generated features rather than existing features. Put another way, we find that the analysis in the study by Luo and Chen [43] is robust to such a change in procedure. Note that sequential LASSO can be used for post-ReFeX-LASSO (but not ReFeX-LASSO) since for post-ReFeX-LASSO we generate all the candidate features in advance. These two propositions together establish two intuitive properties of our selection step in ReFeX-LASSO that we should expect to hold for our purpose. Their proofs can be found in Appendix A.

4.3 Consistency

We now prove that post-ReFeX-LASSO leads to a consistent estimator of the GATE under standard assumptions one would require for consistency of LASSO. For each unit i, we denote the set of features generated by the ReFeX step in post-ReFeX-LASSO as $\{u_i^1, \ldots, u_i^M\}$. We drop the subscript i when we refer to the jth feature vector, i.e., $u^j = [u_1^j, \ldots, u_n^j]^T$. Furthermore, we assume that there exists a subset $S_* \subset \{u^1, \ldots, u^M\}$ with $|S_*| = s$ such that both f_0 and f_1 are linear in features in S_* with coefficient vectors β_0 and β_1, respectively. Finally, we denote the design matrix when estimating β_0 by U^0 and the design matrix when estimating β_1 by U^1.

Theorem 4.3. *Suppose that there exists a constant $C > 0$ such that*

$$\max_{j=1,\ldots,M} \frac{||u^j||_2}{\sqrt{n}} \leq C,$$

and the two design matrices U^0 and U^1 satisfy the $(\kappa;3)$-RE condition over S, then $\hat{\tau}$ is consistent for τ.

A proof of Theorem 4.3 appears in Appendix A and uses mostly standard tools for the study of LASSO ℓ_2-error bounds [44]. The restricted eigenvalue (RE) condition in Theorem 4.3 is a standard assumption when proving ℓ_2-error bound on the coefficient vector. It constrains the curvature of the objective function in such a way that a minor deviation between the empirical loss at the true minimizer and the empirical loss at the empirical risk minimizer implies a small error vector. Consequently, under the RE condition, the error is well controlled. While demonstrating that the RE condition is satisfied for generic random matrices can prove challenging, it has been shown to hold for correlated Gaussian designs [45]. It is defined as follows [45–47].

Definition 4.4. The matrix $\mathbf{X}$ satisfies the restricted eigenvalue (RE) condition over S with parameters $(\kappa;\alpha)$ if

$$\frac{1}{n}||\mathbf{X}\Delta||_2^2 \geq \kappa||\Delta||_2^2 \quad \text{for all } \Delta \in \mathbb{C}_\alpha(S),$$

where $\mathbb{C}_\alpha(S) \coloneqq \{\Delta \in \mathbb{R}^d \mid ||\Delta_{S^c}||_1 \leq \alpha||\Delta_S||_1\}$.

Under the assumptions of Theorem 4.3, we are now able to prove GATE consistency under LASSO-based feature selection in at least simple settings such as the following, an example setting where our feature generation procedure outputs two simple features.

Proposition 4.5. *Suppose we run a Bernoulli randomized experiment with treatment probability $0 < p < 1$ and we only generate two features, the fraction of treated neighbors ρ_i and number of treated neighbors v_i. Furthermore, suppose the graph G consists of disjoint cliques of size $3 \leq m_c \leq M$ (m_c is the size of the c th cluster) for some positive constant $M \geq 3$. If the true f_0 and f_1 are only linear in ρ_i, then $\hat{\tau}$ is consistent for τ.*

The lower bound on m_c is for identifiability and since when all clusters have size 2, then ρ_i and v_i are essentially the same, and we end up with completely duplicated features. Notice also that when all m_c's are equal, we end up with perfect colinearity so in that case we would not consider distinguishing between these two features. While the aforementioned result applies only in a simple setting, it is of its own importance. In practice, it is not uncommon to adjust for fraction of treated neighbors and report the resulting estimate as the estimate of the GATE [11,48]. The aforementioned proposition shows that when we only want to distinguish covariates between fraction of treated neighbors and number of treated neighbors, LASSO is a handy tool.

4.4 Confidence interval via a block bootstrap

Researchers are usually not just interested in a point estimate of the GATE, they also want to know the uncertainty contained in the estimate, e.g., through confidence intervals. ReFeX-LASSO brings flexibility in doing regression adjustment for GATE estimation, but there is no free lunch and it also brings us difficulty in doing inference, i.e., in constructing confidence interval for τ. First, unlike [15] where one assumes an oracle model, here the true model is unknown. Second, features constructed in ref. [15] do not use the observed outcomes. With ReFeX-LASSO, though all the features constructed from ReFeX do not use the outcomes, our selections of covariates *depend* on the realized outcomes. Therefore, ReFeX-LASSO leads to an estimator with no clear variance expression. Moreover, since our final estimate depends on the actual selected covariates, we require some technique analogous to post-selection inference as in the study by Lee et al. [49]. Lee et al. [49] considered confidence intervals of coefficients conditional on being selected by LASSO. Yet we are interested in the confidence interval of τ, not the coefficients, where our estimate $\hat{\tau}$ is calculated based on the estimated coefficients as well as selected covariates. Because of the combination of these complexities, we are not able to simply import any known results for inference in this setting.

Let us consider the nature of the inference problem we are facing. In general, the randomness of our estimate is incurred not only by the randomness of the potential outcomes but also by the randomness of the assignment vector. To construct the confidence interval, we need to quantify how these two resources of randomness affect our estimate of the GATE. Note that since we know the distribution of the assignment vector, the distribution of a given feature is in fact known. What we do not have a good characterization of is the randomness of the selection procedure incurred by the randomness of the assignment vector. In other words, we require understanding how the random assignments affect the feature selection procedure.

To tackle this complication, we introduce a way to construct confidence intervals based on a block bootstrap. Ideally if we can do the experiment infinitely many times, we could run 2^n experiments and calculate 2^n estimates of the GATE. A confidence interval for τ could then be derived easily. Our obvious difficulty is then how should we use one single sample to approximate the sample randomness. We turn to the block bootstrap [25,50,51]. The intuition of this usage is that features of units are correlated according to the particular graph structure of G, and hence, by sampling clusters (which we expect to be relatively disconnected), we are able to keep the bootstrap sample looking like the original sample. On the other hand, resampling units will fail as it cannot replicate the underlying correlation structure in the data. Though we do not provide theoretical guarantees, we will show that in practice the coverage is good and the resulting confidence intervals are of reasonable width. We also note in passing that recent results in the study by Kojevnikov [29] demonstrate that there is a version of block bootstrap that does provide theoretical guarantee for certain highly stylized network processes.

Example 3. Consider the case where our social network G consists of C disjoint cliques $C_1, \ldots, C_C$ of size m. Units are fully connected within each clique. This setup can be viewed as a special case of the household experiment studied in ref. [52]. In such a case, it is natural to consider sampling all C cliques with replacement to obtain a bootstrap sample. For network-dependent processes satisfying certain technical assumptions, this sampling process is the correct thing to do using arguments in the study by Kojevnikov [29]. Suppose we have a network-dependent process $\{Y_n, G_n\}$ that satisfies assumptions in ref. [29]. To make block bootstrap consistent, i.e., producing a confidence interval that is consistent in level, Assumption 4.1 in ref. [29] needs to hold. Tersely

employing the notation of that assumption, it is easy to verify that in our case, $\delta_n(s_n) = m$, $\Delta_n(s_n, 2) = 0$, and $D_n(s_n) = m$ for $\forall s_n \geq \max_c \mathrm{diam}(C_c)$, since our graph consists of nonoverlapping blocks with equal size m. Moreover,

$$\omega_n(i,j) = \begin{cases} 1 & \text{if } i \text{ and } j \text{ are in the same cluster}, \\ 0 & \text{otherwise}. \end{cases}$$

and $\omega_n(j) = 1$ for all $j \in [n]$. With these values, we immediately see that the Assumption 4.1 in the study by Kojevnikov [29] holds as long as $m = o(n)$. Since the only remaining assumptions needed to make block bootstrap consistent are about the network-dependent process itself, we can conclude that block bootstrap would be valid in this toy model for network-dependent processes given in ref. [29].

We present two versions of block bootstrap here, one for regression adjustment with post-ReFeX-LASSO and one for regression adjustment with ReFeX-LASSO. Before actually giving the two block bootstrap procedures, we first introduce the key ingredient in our block bootstrap procedure, a randomized graph clustering algorithm. Our block bootstrap procedure involves partition the graph into several clusters. The generic algorithm we use is k-hop-max clustering [18], a simple adaptation of the CKR partitioning algorithm [53]. The details are shown in Algorithm 3. The algorithm provides a random clustering of the graph that depends on random initial conditions. The algorithm is light in computation when $k = 1$ as we only need to look at one's direct neighbors. Also, it returns neighborhood-like clusters. As a remark connecting back to the aforementioned example, if our graph consists of disjoint fully connected clusters then 1-hop max clustering is able to return exactly these clusters as final output. In general, when $k > 1$, we obtain larger clusters that are centered around fewer nodes.

Algorithm 3. k-hop-max graph clustering

Input: Graph $G = (V, E)$.
Output: Graph clustering $C_1, \ldots, C_c$.
1: **for** $i \in V$ **do**
2: $X_i \leftarrow \mathcal{U}(0, 1)$;
3: **end for**
4: **for** $i \in V$ **do**
5: $i \leftarrow \mathrm{argmax}([X_j \text{ for } j \in B_k(i)])$;
6: **end for**
7: Return $C_1, \ldots, C_c$.

We first present the block bootstrap procedure for post-ReFeX-LASSO, given in Algorithm 4. With post-ReFeX-LASSO, the bootstrap procedure is simpler since the feature generation and selection part are separated. Unlike the usual bootstrap where we sample random individual units with replacement, here we sample random clusters from the graph clustering algorithm with replacement. The intuition is that features u_i of units are correlated according to the particular graph structure of G and hence by sampling clusters, which we expect to be relatively disconnected, we are able to keep the bootstrap sample "looking like" the original sample. As a specific caveat, though in expectation the bootstrap sample has sample size n, if we do not have uniformly sized clusters, then the bootstrap sample may end up with much larger or smaller sample size. Hence, we run the graph clustering algorithm l times, and for each clustering, we run block bootstrap with the number of bootstrap replicates B. We use $k = T^* + 1$ for k-hop-max clustering in Algorithm 4 where T^* is the number of iteration where there were features still got selected. The rationale for this choice is twofold. First, if no feature got selected in the $(T^* + 1)$th iteration, then interference is likely to occur within the T^*-hop neighborhood. In addition, as highlighted in the study by Sävje et al. [54], dependencies among outcomes can stretch beyond the direct dependencies that inform the exposure mappings. This makes it essential to extend our analysis beyond the diameter where interference is occurring. Choosing $k = T^* + 1$ generates clusters that consider information from beyond the T^*-hop neighborhood.

Algorithm 4. Block bootstrap for post-ReFeX-LASSO

Input: Graph $G = (V, \mathcal{E})$, assignment vector $w \in \{0,1\}^n$, number of bootstrap samples B.
Output: Confidence interval for τ.

1: Collect the assignment w_i, features $u_i^1, \ldots, u_i^M$ in S generated before running LASSO, outcome y_i for each unit i. Record the maximum iteration number T^* where one of the features generated at that iteration was selected.
2: Use k-hop max clustering algorithm with $k = T^* + 1$ to divide n units into C clusters $C_1, \ldots, C_C$.
3: **for** $b = 1$ to B **do**
4: Sample C clusters with replacement from $C_1, \ldots, C_C$.
5: Construct the bth bootstrap sample $(w^b, u^{1,b}, \ldots, u^{M,b}, y^b)$ with units from sampled clusters.
6: Regress y on w as well as M features using LASSO.
7: Compute the estimate $\hat{\tau}^b$ using selected features and the bootstrap sample.
8: **end for**
9: Repeat line 2–8 for ℓ times and obtain $\ell \cdot B$ bootstrap estimates in total.
10: Compute the $\alpha/2$th quantile $q^*_{\alpha/2}$ and the $(1-\alpha/2)$th quantile $q^*_{1-\alpha/2}$ of the sample of all bootstrap estimates $\hat{\tau}^1, \ldots, \hat{\tau}^{\ell B}$.
11: Return $[q^*_{\alpha/2}, q^*_{1-\alpha/2}]$ as the $(1-\alpha) \times 100\%$ confidence interval for τ.

Next we present the version of block bootstrap with ReFeX-LASSO, given in Algorithm 5. Note that we cannot simply use the same algorithm since it performs feature generation and feature selection concurrently. Compared to Algorithm 4, T^* now represents the stopping time of ReFeX-LASSO. Meanwhile, similar to Algorithm 4, the bootstrap sample is only used in the feature selection step of ReFeX-LASSO. That being said, for each iteration, we still use the same graph G to generate features, but then we use the bootstrap sample of these features to do selection. The intuition behind using the original graph is that we view the graph as fixed and the correlation structure of all features are then induced by this graph. Therefore, we do not paste all sampled clusters together to form a new graph to generate features for next iteration. On the other hand, if we do believe that the graph is generated from some random process, then we may also reconstruct the graph from sampled units by pasting all sampled clusters together.

Algorithm 5. Block bootstrap for ReFeX-LASSO

Input: Graph $G = (V, \mathcal{E})$, assignment vector $w \in \{0,1\}^n$, number of bootstrap samples B.
Output: Confidence interval for τ.

1: Collect the assignment w_i and outcome y_i for each unit i. Record the stopping time for ReFeX-LASSO T^*.
2: Use k-hop max clustering with $k = T^* + 1$ to divide n units into C clusters $C_1, \ldots, C_C$.
3: **for** $b = 1$ to B **do**
4: Sample C clusters with replacement from $C_1, \ldots, C_C$.
5: Construct the bth bootstrap sample with units from sampled clusters.
6: Rerun ReFeX-LASSO with the original sample for feature generation and the bootstrap sample for feature selection.
7: Use the covariates returned from last step as well as the bootstrap sample to get estimate of τ, $\hat{\tau}^b$.
8: **end for**
9: Repeat line 2–8 for ℓ times and obtain $\ell \cdot B$ bootstrap estimates in total.
10: Compute the $\alpha/2$th quantile $q^*_{\alpha/2}$ and the $(1-\alpha/2)$th quantile $q^*_{1-\alpha/2}$ of the sample of all bootstrap estimates $\hat{\tau}^1, \ldots, \hat{\tau}^{\ell B}$.
11: Return $[q^*_{\alpha/2}, q^*_{1-\alpha/2}]$ as the $(1-\alpha) \times 100\%$ confidence interval for τ.

In the aforementioned two algorithms, we utilize a randomized graph clustering algorithm that can be easily implemented. Of course, this is not the only possible choice for the graph clustering algorithm one can use. We note by passing that there are many graph clustering algorithms available for practitioners [55–58] that exhibit various properties.

We conclude this section with a discussion of how to suitably choose the sizes of clusters. We consider three scenarios and show why they may fail with heuristics from Kojevnikov [29]. Though we are not considering the same problem as in Kojevnikov [29], given that we have a more complicated setup, we do not expect that weaker assumptions than those in the study by Kojevnikov [29] would be sufficient for good coverage in our case. Therefore, we view assumptions in the study by Kojevnikov [29] as what we should expect to have to make our block bootstrap consistent.

The first scenario that we consider is when we have $O(n)$ clusters with nonconstant sizes. Then the second absolute central moment of block sizes may be nonvanishing as $n \to \infty$, but the average block size is $O(1)$. This implies that unless the clusters are relatively uniform, there would be a violation to Assumption 4.1 in ref. [29]. As a second scenario, consider the case when we have $O(1)$ clusters. Now the maximum block size must be of order $O(n)$ and the average block size is at most $O(n)$, hence Assumption 4.1 in ref. [29] is certainly violated. In general, we do not want to have too many clusters or too few clusters. Finally, then, consider a scenario where we have $\sqrt{n} - 1$ clusters of size $\sqrt{n}$ and $\sqrt{n}$ clusters of size 1. Now the average block size is of order $O(n^{1/2})$ and the second absolute central moment of block sizes is not of lower order, which implies that the ratio does not vanish as $n \to 0$, and again Assumption 4.1 in ref. [29] is violated. This last example shows that the cluster sizes are not simply a matter of avoiding too big/small or few/many clusters, but instead here, we see we cannot have two groups of clusters with different size magnitudes. In summary, the advice is to use a reasonable number of clusters that have sizes of roughly the same magnitude. What we present in Algorithms 4 and 5 are good default choices if the network is not very dense.

5 Simulation experiments

In this section, we use simulations to provide both empirical guidance on our method when theory is lacking and empirical evidence of the usefulness of our method. We make use of the Facebook 100 dataset [59] of real-world social networks. The networks in this dataset are complete online friendship networks for 100 colleges and universities collected from a single-day snapshot of Facebook in September 2005. For our simulations, we use the network of Swarthmore college students, being of modest size. We extract the largest connected components of the Swarthmore network, obtaining a social network with 1,657 nodes and 61,049 edges. The diameter of the network is 6, and the average pairwise distance is 2.32. Since this network is quite dense, estimation of the GATE would be very difficult when interference is strong. Indeed, we choose to test our simulations on this college social network specifically because of the challenges posed to GATE estimation by the high density. Comparable simulations on the more widely studied farmer's network from Cai et al. [21], which we also study in Section 6 as a real-world test of our methods, are given in Appendix B. We use this network to demonstrate that even for such a network, we are still able to obtain relatively good estimates from (post-) ReFeX-LASSO.

We generate an assignment vector using a Bernoulli design with success probability 0.5 and generate outcome variables according to certain models with varying magnitude of network interference; these models are summarized in Tables 1 and 2. We will discuss in detail about these outcome models in Section 5.2. Our

Table 1: Parameters of simple linear interference outcome model ((8) and (9)) used in simulation experiments

Model type	(α_0, α_1)	(ξ_0, ξ_1)	(γ_0, γ_1)
Model 0	(0, 2)	(0, 0)	(0, 0)
Model 1	(0, 2)	(1, 1.5)	(0.005, 0.0025)
Model 2	(0, 2)	(1, 2)	(0.005, 0.01)

Table 2: Parameters of truncated linear-in-means outcome model used in simulation experiments

Model type	α	β	γ	J
Model 3	1	5	2	2
Model 4	1	5	3	2
Model 5	1	5	1	3
Model 6	1	5	2	3

simulations can be viewed as semi-synthetic experiments – we use a true social network, but we generate outcomes according to specified models. Section 5.1

introduces the baseline estimators that we compare with in our simulations. Section 5.2 discusses the outcome models that we use for generating the outcomes with various degree of interference. Section 5.3 compares the regression adjustment estimator using model-free covariates with those commonly used estimators in practice as given in Section 5.1 and demonstrates that it has good performance in terms of root-mean-square error (RMSE). Section 5.4 explores the empirical performance of the confidence interval constructed via block bootstrap and discusses some practical aspects in the procedure.

5.1 Estimation of the GATE

Our ultimate goal of constructing model-free covariates is to use them in GATE estimation. We first explore the empirical performance of the regression adjustment estimator using model-free covariates. Specifically, we compare it with two kinds of estimators that are commonly used in practice: (i) the difference-in-mean estimator and (ii) a Hájek estimator under a network exposure model [7]. Difference-in-mean estimator calculates the difference between average outcome among treated units and average outcome among control units:

$$\hat{\tau}^{DM} = \frac{1}{\sum_{i=1}^{n} W_i} \sum_{i=1}^{n} Y_i W_i - \frac{1}{\sum_{i=1}^{n} (1 - W_i)} \sum_{i=1}^{n} Y_i (1 - W_i).$$

Obviously this estimator ignores interference and will thus incur large bias when interference is significant.

The basic Hájek estimator for the ATE is defined as follows:

$$\hat{\tau}^{\text{Hájek}} = \frac{\sum_{i=1}^{n} Y_i W_i / \mathbb{P}(W_i = 1)}{\sum_{i=1}^{n} \mathbb{I}(W_i = 1) / \mathbb{P}(W_i = 1)} - \frac{\sum_{i=1}^{n} Y_i (1 - W_i) / \mathbb{P}(W_i = 0)}{\sum_{i=1}^{n} \mathbb{I}(W_i = 0) / \mathbb{P}(W_i = 0)}.$$

Here, we will consider a version of Hájek estimator that accounts for interference. Manski [7] studies identification of potential outcome distributions under interference. One concrete example is when one's outcome only depends on one's own assignment as well as the distribution of assignments for his/her neighbors. Ugander et al. [17] further considers a fractional exposure model where it is assumed that if one is treated and a $q > 0.5$ fraction of one's neighbors is treated, then one's outcome is equal to the potential outcome associated with the assignment vector $\mathbf{1}$. Similarly, in this exposure model if one is not treated and one's fraction of treated neighbors is at most $1 - q$, then one's outcome is equal to the potential outcome associated with the assignment vector $\mathbf{0}$. Formally, $\forall w, w' \in \{0, 1\}^n$, this fractional exposure model assumes:

$$w_i = 1, \frac{1}{|\mathcal{N}_i|} \sum_{j \in \mathcal{N}_i} w_j \geq q \;\Rightarrow\; Y_i(w) = Y_i(\mathbf{1}),$$

and

$$w_i = 0, \frac{1}{|\mathcal{N}_i|} \sum_{j \in \mathcal{N}_i} w_j \leq 1 - q \;\Rightarrow\; Y_i(w) = Y_i(\mathbf{0}).$$

We can then use a Hájek estimator that corrects for the probability that these conditions are met under a Bernoulli design. Specifically, we define the events $E_i^{1,q} = \{W_i = 1, \frac{1}{|\mathcal{N}_i|}\sum_{j\in\mathcal{N}_i} w_j \geq q\}$ and $E_i^{0,1-q} = \{W_i = 0, \frac{1}{|\mathcal{N}_i|}\sum_{j\in\mathcal{N}_i} w_j \leq 1-q\}$. The corresponding Hájek estimator under a fractional exposure model is then

$$\hat{\tau}^{\text{Hájek}}_{q,1-q} = \frac{\sum_{i=1}^n Y_i \mathbb{1}(E_i^{1,q})/\mathbb{P}(E_i^{1,q})}{\sum_{i=1}^n \mathbb{1}(E_i^{1,q})/\mathbb{P}(E_i^{1,q})} - \frac{\sum_{i=1}^n Y_i \mathbb{1}(E_i^{0,1-q})/\mathbb{P}(E_i^{0,1-q})}{\sum_{i=1}^n \mathbb{1}(E_i^{0,1-q})/\mathbb{P}(E_i^{0,1-q})}. \tag{7}$$

This estimator accounts for interference by taking the assignments of direct neighbors into consideration. If we still assume local interference in the sense that only one's direct neighbors can impact one's response but want a fully agnostic setting, then we could choose $q = 1$ (notice that in this case the Hájek estimator is consistent). In our case, the number of neighbors that one has is usually quite large, and under independent Bernoulli assignment, we would not expect to observe many units with all neighbors being treated or not treated. As a bias-variance compromise, we choose $q = 0.8$.

Finally, we also compare our (post-) ReFeX-LASSO regression adjustment estimator with two linear regression adjustment estimators that adjust for specific features. We will describe these two estimators in detail later when we present the simulation results in Section 5.3. For post-ReFeX-LASSO and ReFeX-LASSO, we choose $T = 2$ and the base features to be fraction of treated neighbors, number of treated neighbors, fraction of edges in neighborhood that connects a treated unit, and a control unit and also fraction of edges in neighborhood that connects a treated unit, and a treated unit. For aggregation functions in (post-) ReFeX-LASSO, we use both the mean and variance.

5.2 Outcome models

Here, we describe the outcome models we use in our simulation study. We carry forward the notation as in Proposition 4.5, using ρ_i to denote the fraction of treated direct neighbors for unit i and ν_i to denote number of treated direct neighbors.

We first consider estimation under linear interference. The first model is a linear model in both number of treated neighbors and fraction of treated neighbors. Such model is also considered in the studies by Pouget-Abadie et al. [10] and Chin [15]. Specifically,

$$f_0(w, G) = \alpha_0 + \xi_0\rho_i + \gamma_0\nu_i \tag{8}$$

and

$$f_1(w, G) = \alpha_1 + \xi_1\rho_i + \gamma_1\nu_i. \tag{9}$$

The difference $\alpha_1 - \alpha_0$ can be viewed as the primary effect of the treatment and coefficients (ξ_w, γ_w) for $w = 0, 1$ govern how the unit respond to treatment and control, respectively. In particular, if $\xi_w = \gamma_w = 0$, then there is no interference, and we are back to usual setup of ATE estimation under SUTVA. Note that for this model, there is no interference beyond the 1-hop neighborhood, and hence, the estimation problem is considerably easier. We will refer to this response model as simple linear interference.

Building on the discussion of the linear-in-means model in the introduction, we also consider a response model where the interference propagates out to k-hop neighborhoods for $k \geq 2$. This model can be viewed as a truncated linear-in-means model; instead of summing up to infinity, we truncate the model at $j = J$ for some number $J > 1$.

Overall we consider the following model configurations of linear interference. Table 1 and 2 summarize the configurations of the models we consider for simulations. Note that model 0 exhibits no interference. For all models, the error terms are independently normally distributed with variance 1. The true GATE in these outcome models (either by an exact calculation or by a Monte Carlo estimate on the Swarthmore network) are 2, 3.69, 4.74, 15, 20, 15, and 35.

Beyond linear interference, we also examine a slightly more complicated scenario where linear interference is violated. In particular, we consider f_0 and f_1 that are nonlinear in ρ_i and ν_i. The nonlinear functions we use are sigmoid-type so that it is hard to approximate by any linear model[3]. We use the Monte Carlo estimate, 9.55, as the true GATE when reporting the simulation results. Our purpose here is to show that even if we have nonlinear f_0 and f_1 which violates our linear interference assumption, our method still leads to an estimator with reasonable performance. This also echoes our previous discussion. In GATE estimation, we are always predicting for a data point that is outside the range of our observed/training data and hence a simple model can be quite reliable.

5.3 Simulation results

We study both the bias and the RMSE of each estimator under these varied models. Additional simulation results, employing two commonly studied network models – the stochastic block model [60–63] and the small-world model [64–68] – can be found in Appendix B. The objective of these additional simulations is to demonstrate that the results presented in this section remain largely consistent, even when the structure of the underlying networks is changed. Tables 3 and 4 show the RMSE and bias of several different estimators under linear interference. In these two tables, we show results of two kinds of regression adjustment estimators. $\hat{\tau}_{\text{frac}}$ is the regression adjustment estimator that adjusts for the fraction of treated neighbors and $\hat{\tau}_{\text{num}}$ adjusts for the number of treated neighbors. They are also considered in ref. [15]. We also show the oracle adjustment estimator $\hat{\tau}_{\text{oracle}}$ as a reference, which marks the best we can do with full knowledge of the

Table 3: RMSE of estimators of the GATE assuming linear interference (simple linear interference and truncated linear-in-means) outcome models

Estimator	$\hat{\tau}^{\text{DM}}$	$\hat{\tau}^{\text{Hájek}}_{0.8,0.2}$	$\hat{\tau}_{\text{frac}}$	$\hat{\tau}_{\text{num}}$	Post-ReFeX-LASSO	ReFeX-LASSO	$\hat{\tau}_{\text{oracle}}$
Model 0	0.05	0.76	0.24	0.07	0.50	0.32	0.05
Model 1	1.53	1.02	0.36	1.22	1.54	0.70	0.25
Model 2	2.06	1.41	0.47	1.49	1.49	0.59	0.24
Model 3	10.02	3.84	0.37	9.86	1.08	0.93	0.37
Model 4	15.02	5.60	0.56	14.72	1.68	1.59	0.56
Model 5	9.92	6.61	4.53	9.82	1.38	1.73	1.98
Model 6	29.67	22.31	18.22	29.46	2.42	2.47	4.45

Table 4: Empirical bias of estimators of the GATE assuming linear interference (simple linear interference and truncated linear-in-means) outcome models

Estimator	$\hat{\tau}^{\text{DM}}$	$\hat{\tau}^{\text{Hájek}}_{0.8,0.2}$	$\hat{\tau}_{\text{frac}}$	$\hat{\tau}_{\text{num}}$	Post-ReFeX-LASSO	ReFeX-LASSO	$\hat{\tau}_{\text{oracle}}$
Model 0	0.004	0.120	0.021	0.009	0.106	0.042	0.004
Model 1	–1.53	–0.68	–0.25	–1.22	–0.72	–0.004	–0.05
Model 2	–2.06	–1.16	–0.39	–1.48	–0.56	–0.29	0.008
Model 3	–10.02	–3.67	–0.01	–9.85	0.28	0.19	–0.01
Model 4	–15.02	–5.46	0.003	–14.72	0.36	0.31	0.03
Model 5	–9.92	–6.55	–4.52	–9.82	–0.01	–0.24	0.68
Model 6	–29.67	–22.28	–18.21	–29.45	–0.21	–0.31	2.77

3 We document this model in the Appendix B.

Table 5: RMSE and empirical bias of estimators of the GATE assuming a nonlinear interference (Appendix B) outcome model

Estimator	$\hat{\tau}^{\mathrm{DM}}$	$\hat{\tau}^{\mathrm{Hájek}}_{0.8,0.2}$	$\hat{\tau}_{\mathrm{frac}}$	$\hat{\tau}_{\mathrm{num}}$	Post-ReFeX-LASSO	ReFeX-LASSO
Bias	–5.54	–2.72	–1.56	–2.72	1.29	1.33
RMSE	5.55	3.73	1.92	2.73	5.68	2.75

response model. Note that in some cases other estimators can perform better than the oracle since the oracle adjustment estimator only means we use oracle control covariates. The covariates are inevitably random and we are not averaging over all possible assignment vectors. Moreover, for the truncated linear-in-means model, the true covariates are highly correlated, causing the oracle adjustment estimator to have a large variance. Finally, $\hat{\tau}^{\mathrm{DM}}$ and $\hat{\tau}^{\mathrm{Hájek}}_{0.8,0.2}$ refer to the simple difference-in-mean estimator and the Hájek estimator in Equation (7) with $q = 0.8$ as we mentioned earlier.

First, if we look at the results for Model 0, i.e., when there is no interference, post-ReFeX-LASSO and ReFeX-LASSO all give better performance compared to the Hájek estimator. Second, for Model 1 and Model 2, the true interference mechanism is simple linear interference. As we can see from the first two rows of Table 3 and Table 4, if we fail to account for one feature, the bias and/or the RMSE can be large. Also, ReFeX-LASSO is dominating post-ReFeX-LASSO with significantly lower bias and RMSE since for this case ReFeX-LASSO is able to stop considering further features after the first iteration. For Models 3–6, the underlying model is a truncated linear-in-means model, and the only difference between them is the stopping number J. For the models with $J = 2$ (Models 3 and 4), the interference is still local, i.e., within one's direct neighbors, but for $J = 3$ (Models 5 and 6), it is crucial to consider information from 2-hop neighbors. Our simulation results verify this intuition. We see that $\hat{\tau}_{\mathrm{frac}}$ is doing well for model 3 and 4 but very poorly for model 5 and 6. We also note in passing that $\hat{\tau}_{\mathrm{frac}}$ and $\hat{\tau}_{\mathrm{oracle}}$ give the same results for Models 3 and 4, since for these two models $\hat{\tau}_{\mathrm{frac}}$ happens to use the oracle features. Both post-ReFeX-LASSO and ReFeX-LASSO lead to estimators with relatively small bias and small RMSE for these more challenging response models.

Turning to the nonlinear model, Table 5 shows our results there. In this case, $\hat{\tau}_{\mathrm{frac}}$ and $\hat{\tau}_{\mathrm{num}}$ represent the same regression adjustment estimators as in the linear case. Compared to difference-in-means and Hájek, ReFeX-LASSO leads to estimator with much better performance. Also, based on the comparison of $\hat{\tau}_{\mathrm{frac}}$, $\hat{\tau}_{\mathrm{num}}$, and ReFeX-LASSO, we see that, as in the linear interference case, even if we happen to adjust for some feature that is of importance, failing to take all relevant features into account will lead to estimators with either large bias, large variance, or both. In other words, ReFeX-LASSO helps one choose which set of features to adjust for and hence incur much smaller bias or variance.

From these simulations, we take away that ReFeX-LASSO is able to identify influential features for regression adjustment and hence produce an estimator with relatively good performance across many model specifications. We also see that ReFeX-LASSO generally, though not always, performs significantly better than post-ReFeX-LASSO. This is due to the fact that we select features sequentially and hence reduce the variance. In contrast, a standard regression adjustment estimator considered in the study by [15] for some network features ($\hat{\tau}_{\mathrm{frac}}$ and $\hat{\tau}_{\mathrm{num}}$ in our simulations) can be far-off if we fail to choose the right feature. Finally, exposure mapping-based estimator like the fractional-exposure-Hájek estimator can also be pretty bad if we have interference that is quite different from the assumptions of the exposure model that such estimators assume.

5.4 Confidence interval for the GATE

In Section 4.4, we introduced a way to construct a confidence interval for τ via a block bootstrap and gave an explicit algorithm for graph-based block construction. We now evaluate the empirical coverage of the resulting confidence interval from our block bootstrap. Throughout this section, we focus on 90% confidence interval for τ. Instead of using the Swarthmore College network as in the previous section, we use the farmer network in ref. [21] where we have a larger and sparser network compared to the Swarthmore College

Table 6: Additional parameters of simple linear interference model ((8) and (9)) used in simulation experiments

Model type	(α_0, α_1)	(ξ_0, ξ_1)	(γ_0, γ_1)
Model 2a	(0, 2)	(1, 3)	(0.01, 0.025)
Model 2b	(0, 2)	(1, 3)	(0.05, 0.15)

Table 7: Coverage of different bootstrap 90% confidence intervals for the GATE with linear interference (simple linear interference and truncated linear-in-means) outcome models

Model	Post-ReFeX-LASSO (%)	ReFeX-LASSO (%)	Naive Bootstrap (%)	Bootstrap with oracle clustering (%)
Model 2a	93	92	94	92
Model 2b	92	96	93	95
Model 3	90	90	83	91
Model 4	88	87	80	91
Model 5	91	93	84	92
Model 6	90	91	67	93

Table 8: Average length of 90% confidence intervals for the GATE with linear interference (simple linear interference and truncated linear-in-means) outcome models

Model	Post-ReFeX-LASSO	ReFeX-LASSO	Naive Bootstrap	Bootstrap with oracle clustering
Model 2a	0.245	0.220	0.235	0.228
Model 2b	0.435	0.384	0.390	0.382
Model 3	0.403	0.380	0.330	0.414
Model 4	0.569	0.534	0.437	0.593
Model 5	0.552	0.549	0.431	0.567
Model 6	1.316	1.316	0.751	1.412

network. In fact, the average size of 2-hop neighborhoods in the Swarthmore network is 1092.65 and the average size of 3-hop neighborhoods in the Swarthmore network is 1622.27. Hence, if we believe that interference is beyond 1-hop neighborhood, bootstrap will not perform well on such a dense graph since it is hard to create bootstrap samples that respect the structure in the original sample.[4] On the other hand, the farmer network in ref. [21] is less dense with 2-hop neighborhoods having an average size 23.95 and 3-hop neighborhoods having an average size 41.49. We will introduce in more details about the background and the details of this network in Section 6. In general, if the network is too dense to produce well-isolated and balanced clusters, then the bootstrap would fail. One thing to notice is that the farmer network itself is associated with a natural clustering based on which village the each farmer lives in, namely, each village can be viewed as a cluster in the network. In our simulations here, we thus also show the results of constructing the confidence interval with block bootstrap of ReFeX-LASSO that uses this "oracle clustering" of villages. Finally, since we have a sparser network (making interference easier to manage), we consider two different sets of parameters for linear models that make the effect from number of treated neighbors larger (and thus GATE estimation harder). Table 6 shows the values of the parameters, loosely based on Model 2 (thus named 2a and 2b).

We first evaluate the effectiveness of such a bootstrap method. We assume linear interference and consider Models 3–6 as well as Models 2a and 2b. We fix $\ell = 3$, $B = 100$, and the coverage is calculated by repeating the whole process 100 times. Tables 7 and 8 show the coverage and the average length of the

4 We found that the block bootstrap still gives near to nominal coverage on Swarthmore network when interference is local, i.e., within direct neighbors.

Table 9: Coverage of block bootstrap 90% confidence intervals for the GATE using different graph clustering algorithms with Model 6 as the true outcome model

Model	2-hop max	3-hop max	Five clusters
Model 6	84%	89%	45%

confidence intervals constructed from our block bootstrap of post-ReFeX-LASSO and ReFeX-LASSO. To show the necessity of using block bootstrap and of considering the randomness of the assignment vector, we also include the result of constructing confidence interval using a naive bootstrap where we just sample each unit with replacement.

As we can see from the results, our block bootstrap gives us near nominal coverage for ReFeX-LASSO and slightly worse but still close to nominal coverage for post-ReFeX-LASSO. However, the naive bootstrap fails to deliver confidence interval with nominal coverage. In fact, naive bootstrap-based confidence intervals can give us very bad coverage in some cases. We are also able to obtain good confidence intervals if we use the oracle clustering that is associated with the network. In scenarios where there are clear natural clusters in the network, these clusters can be a good default choice to use for block bootstrap. Moreover, as is shown in Table 8, both the block bootstrap confidence interval for ReFeX-LASSO and the block bootstrap confidence interval for post-ReFeX-LASSO are of reasonable length.

We conclude this section with a simulation to show why choosing the k for k-hop max clustering adaptively in our block bootstrap procedure is important and how partitioning the graph into just two clusters fails to give correct coverage. To this end, we consider using 2-hop max and 3-hop max clustering to divide units into clusters as well as randomly divide units into five clusters, i.i.d., without considering the underlying graph structure. We choose to consider 2-hop max and 3-hop max as we found in the simulations that in most of the cases ReFeX-LASSO will stop after selecting features about 2-hop neighborhoods. For the Cai network, on average 2-hop max clustering and 3-hop clustering produce 267 and 269 clusters, respectively. We choose to compare them with a five-cluster clustering as five is a lot less than the number of clusters we may have using k-hop max clustering. We rerun the block bootstrap procedure with these new clusters for Model 6 using ReFeX-LASSO. Table 9 shows the coverage of the confidence intervals. As we can see, contrast to the 91% coverage in Table 7 provided by the adaptive k-hop max based block bootstrap, all these three clustering methods fail to give us nominal coverage. In particular, completely ignoring the graph structure ("five clusters") leads to confidence intervals with really poor coverage.

6 Real data example

In this section, we would like to apply our method to a real experiment where interference is known to exist and simple estimators such as difference-in-means would give poor GATE estimates. We consider data from the intervention in ref. [21]. They designed a randomized experiment to study the role of social networks on insurance adoption in rural China. Specifically, a random subset of farmers were provided with intensive information sessions about the an insurance product. Cai et al. [21] found that the diffusion of insurance knowledge drove network effects in product adoption. Hence, these data are ideal for our purpose in the sense that we know for sure that SUTVA is violated and we should not trust the simple difference-in-means estimate for estimating the GATE. Moreover, though we know that network effects do exist, defining an exact exposure model as in ref. [12] is difficult. Hence, analysis done in ref. [15] is limited since there only four prespecified features were considered and hence the regression adjustment estimator implicitly assumed a certain exposure model. We revisit this experiment and estimate the GATE using our method.

In the original field experiment in the study by Cai et al. [21], the intensive information sessions were offered in two separate rounds, leading to four separate treatment arms. For our purpose, following [15], we simplify the experiment by viewing the two intensive information sessions as the same treatment arm. Hence,

Table 10: Estimates and standard errors of different estimators for the global average treatment effect on insurance adoption [21]

Estimator	Estimate	Standard error
DM	0.078	—
Hájek_1hop (q = 0.75)	0.163	—
Hájek_2hop (q = 0.75)	0.167	—
$\hat{\tau}_{\text{chin}}$	0.122	0.056
$\hat{\tau}_{\text{num}}$	0.178	0.027
$\hat{\tau}_{\text{refex-lasso}}$	0.178	0.043

we reduce the original field experiment to a binary randomized experiment. As in the study by Cai et al. [21], the outcome variable is set to be the binary indicator variable for the weather insurance adoption, and we do not include villagers whose treatment or response information was missing as well as villagers whose network information was missing. We also combine all the villages into one social network, denoting this single social network by G. In summary, we have 4,382 nodes and 17,069 edges. This network is also the one that we used in Section 5.4.

The first step for our method is generating model-free covariates. We use exactly the same set of base features as in the previous simulation section – fraction of treated neighbors, number of treated neighbors, fraction of edges in neighborhood that connects a treated unit, and a control unit and also fraction of edges in neighborhood that connects a treated unit and a treated unit. We then use ReFeX-LASSO to generate a group of covariates, using mean and variance aggregation functions (again, as in the previous simulation section), and estimate the GATE by adjusting for these covariates with a linear model. We compare the standard error estimate from block bootstrap with the one computed in ref. [15].

Table 10 shows the resulting GATE estimates, where $\hat{\tau}_{\text{chin}}$ is the estimator in ref. [15] that adjusts for four covariates: the fraction of treated neighbors, the number of treated neighbors, the fraction of treated neighbors in 2-hop neighborhoods, and the number of treated neighbors in 2-hop neighborhoods. Meanwhile, $\hat{\tau}_{\text{num}}$ only adjusts for the number of treated neighbors and $\hat{\tau}_{\text{refex-lasso}}$ is the ReFeX-LASSO-based adjustment estimator. Here, *DM* refers to the difference-in-means estimator, *Hájek_1hop* assumes a fractional exposure model for 1-hop neighborhood, while *Hájek_2hop* assumes a fractional exposure model for 2-hop neighborhood, i.e., we use (7) but consider 2-hop neighbors instead. The intuition is that sometimes units that are not direct neighbors, but neighbors of direct neighbors matter as well, and by considering fractional exposure model for 2-hop neighborhood, we are able to take these units into account for the exposure model. As shown in Cai et al. [21], the network effects in this scenario are primarily driven by the diffusion of insurance knowledge across the network. Consequently, one can reasonably anticipate that a higher number of treated units would correspond to an elevated insurance adoption rate. That being said, the GATE should not be lower than the estimate derived from the Hájek estimator, since the working assumption of the Hájek estimator is that there is no difference in the outcomes when a unit has more than 75% or less than 25% of treated neighbors (within either the 1-hop or 2-hop neighborhood). We further notice that $\hat{\tau}_{\text{num}}$ and $\hat{\tau}_{\text{refex-lasso}}$ give us the same estimate, and indeed, the only covariate selected from ReFeX-LASSO is the number of treated neighbors. They are also the only two estimators that give estimates greater than or equal to those derived from the two Hájek estimators. Compared to $\hat{\tau}_{\text{chin}}$, $\hat{\tau}_{\text{refex-lasso}}$ also has a smaller standard error despite having a larger estimate of the effect. Finally, though $\hat{\tau}_{\text{num}}$ and $\hat{\tau}_{\text{refex-lasso}}$ give nearly the same estimates (same up to three decimal digits), we see that the former has a smaller standard error. The reasons are twofold. First, the bootstrap in general is conservative. Second, the ReFeX-based estimate should have larger variance as we have a random selection procedure involved.

7 Discussion

In this article, we have developed a method to do estimation and inference for the GATE when network interference is present. We develop a procedure that can be used to estimate the GATE without prespecifying

either exposure mappings or outcome models. We also give a way to construct confidence intervals for the GATE using a block bootstrap. We evaluate our method both through simulations and a real data example.

Many interesting avenues of further investigation have been left unexplored in this article. First, our results only consider designs that satisfy the uniformity assumption (e.g., Bernoulli designs): this is, of course, limiting, but it does present a useful benchmark. We are particularly interested in exploring how to extend our work to designs that violate the uniformity assumption such as cluster randomized design. This is challenging since the covariates we adjust for may be correlated with the treatment assignment. Second, while our simulations show that the block bootstrap behaves well in practice, formal results are absent for anything other than a simple toy setting. Third, beyond linear adjustment, we may also want to have a completely nonlinear model to estimate the outcomes using the covariates returned from the ReFeX-LASSO feature generation and selection process.

Funding information: This work was supported in part by ARO MURI award #W911NF-20-1-0252 and NSF CAREER Award #2143176.

Author contributions: All authors have accepted responsibility for the entire content of this article and approved its submission.

Conflict of interest: The authors state no conflict of interest.

Data availability statement: The Facebook college network dataset is available in the GitHub repository [https://github.com/ajchin/regression-adjustments]. The farmers network dataset is available in the openICPSR repository [https://www.openicpsr.org/openicpsr/project/113593/version/V1/view].

Ethical approval: The conducted research is not related to either human or animals use.

References

[1] Cox DR. Planning of experiments. New York: Wiley; 1958.
[2] Rubin DB. Estimating causal effects of treatments in randomized and nonrandomized studies. J Educ Psychol. 1974;66(5):688.
[3] Sinclair B, McConnell M, Green DP. Detecting spillover effects: design and analysis of multilevel experiments. Amer J Politic Sci. 2012;56(4):1055–69. https://onlinelibrary.wiley.com/doi/abs/10.1111/j.1540-5907.2012.00592.x.
[4] Hong G, Raudenbush SW. Evaluating Kindergarten retention policy. J Amer Stat Assoc. 2006;101(475):901–10. doi: 10.1198/016214506000000447.
[5] Rosenbaum PR. Interference between units in randomized experiments. J Amer Stat Assoc. 2007;102(477):191–200. doi: 10.1198/016214506000001112.
[6] Sobel ME. What do randomized studies of housing mobility demonstrate? J Amer Stat Assoc. 2006;101(476):1398–407. doi: 10.1198/016214506000000636.
[7] Manski CF. Identification of treatment response with social interactions. Econom J. 2013;16(1):S1–S23. https://onlinelibrary.wiley.com/doi/abs/10.1111/j.1368-423X.2012.00368.x.
[8] Halloran ME, Struchiner CJ. Causal inference in infectious diseases. Epidemiology. 1995;6(2):142–51. http://www.jstor.org/stable/3702315.
[9] Eckles D, Karrer B, Ugander J. Design and analysis of experiments in networks: reducing bias from interference. J Causal Infer. 2017;5(1):20150021. doi: 10.1515/jci-2015-0021 [cited 2022-10-27].
[10] Pouget-Abadie J, Saint-Jacques G, Saveski M, Duan W, Ghosh S, Xu Y, et al. Testing for arbitrary interference on experimentation platforms. Biometrika. 2019 Sep;106(4):929–40. doi: 10.1093/biomet/asz047.
[11] Karrer B, Shi L, Bhole M, Goldman M, Palmer T, Gelman C, et al. Network experimentation at scale. In: Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining. KDD '21. New York, NY, USA: Association for Computing Machinery; 2021. p. 3106–16. doi: 10.1145/3447548.3467091.
[12] Aronow PM, Samii C. Estimating average causal effects under general interference, with application to a social network experiment. Annal Appl Stat. 2017;11(4):1912–47. doi: 10.1214/16-AOAS1005.
[13] Lin W. Agnostic notes on regression adjustments to experimental data: reexamining Freedman's critique. Ann Appl Stat. 2013;7(1):295–318. doi: 10.1214/12-AOAS583.

[14] Deng A, Xu Y, Kohavi R, Walker T. Improving the sensitivity of online controlled experiments by utilizing pre-experiment data. In: Proceedings of the Sixth ACM International Conference on Web Search and Data Mining. WSDM '13. New York, NY, USA: Association for Computing Machinery; 2013. p. 123–32. doi: 10.1145/2433396.2433413.
[15] Chin A. Regression adjustments for estimating the global treatment effect in experiments with interference. J Causal Infer. 2019;7(2):20180026. doi: 10.1515/jci-2018-0026.
[16] Harshaw C, Sävje F, Eisenstat D, Mirrokni V, Pouget-Abadie J. Design and analysis of bipartite experiments under a linear exposure-response model. In: Proceedings of the 23rd ACM Conference on Economics and Computation. EC '22. New York, NY, USA: Association for Computing Machinery; 2022. p. 606. doi: 10.1145/3490486.3538269.
[17] Ugander J, Karrer B, Backstrom L, Kleinberg JM. Graph cluster randomization: network exposure to multiple universes. In: Dhillon IS, Koren Y, Ghani R, Senator TE, Bradley P, Parekh R, et al., editors. The 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD 2013, August 11–14, 2013. Chicago, IL, USA: ACM; 2013. p. 329–37. doi: 10.1145/2487575.2487695.
[18] Ugander J, Yin H. Randomized graph cluster randomization. 2020. https://arxiv.org/abs/2009.02297.
[19] Hudgens MG, Halloran ME. Toward causal inference with interference. J Amer Stat Assoc. 2008;103(482):832–42.
[20] Sussman DL, Airoldi EM. Elements of estimation theory for causal effects in the presence of network interference. 2017. arXiv: http://arXiv.org/abs/arXiv:170203578.
[21] Cai J, De Janvry A, Sadoulet E.. Social networks and the decision to insure. Amer Econ J Appl Econ. 2015 April;7(2):81–108. https://www.aeaweb.org/articles?id=10.1257/app.20130442.
[22] Toulis P, Kao E. Estimation of causal peer influence effects. In: Dasgupta S, McAllester D, editors. Proceedings of the 30th International Conference on Machine Learning. vol. 28 of Proceedings of Machine Learning Research. Atlanta, Georgia, USA: PMLR; 2013. p. 1489–97. https://proceedings.mlr.press/v28/toulis13.html.
[23] Gui H, Xu Y, Bhasin A, Han J. Network a/b testing: From sampling to estimation. In: Proceedings of the 24th International Conference on World Wide Web; 2015. p. 399–409.
[24] Henderson K, Gallagher B, Li L, Akoglu L, Eliassi-Rad T, Tong H, et al. It's who you know: graph mining using recursive structural features. In: Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. KDD '11. New York, NY, USA: Association for Computing Machinery; 2011. p. 663–71. doi: 10.1145/2020408.2020512.
[25] Efron B. Bootstrap methods: another look at the Jackknife. Ann Stat. 1979;7(1):1–26. doi: 10.1214/aos/1176344552.
[26] Kunsch HR. The Jackknife and the bootstrap for general stationary observations. Ann Stat. 1989;17(3):1217–41.
[27] Carlstein E. The use of subseries values for estimating the variance of a general statistic from a stationary sequence. Ann Stat. 1986;14(3):1171–9.
[28] Carlstein E, Do KA, Hall P, Hesterberg T, Kü nsch HR. Matched-block bootstrap for dependent data. Bernoulli. 1998;4(3):305–28.
[29] Kojevnikov D. The bootstrap for network-dependent processes. 2021. https://arxiv.org/abs/2101.12312.
[30] Holland PW. Statistics and causal inference. J Amer Stat Assoc. 1986;81(396):945–60. http://www.jstor.org/stable/2289064.
[31] Imbens GW, Rubin DB. Causal inference for statistics, social, and biomedical sciences: an introduction. Cambridge, UK: Cambridge University Press; 2015.
[32] Yu CL, Airoldi EM, Borgs C, Chayes JT. Estimating the total treatment effect in randomized experiments with unknown network structure. Proc Nat Acad Sci. 2022;119(44):e2208975119. https://www.pnas.org/doi/abs/10.1073/pnas.2208975119.
[33] Basse GW, Airoldi EM. Limitations of design-based causal inference and A/b testing under arbitrary and network interference. Sociol Methodol. 2018;48(1):136–51. doi: 10.1177/0081175018782569.
[34] Manski CF. Identification of endogenous social effects: the reflection problem. Rev Econ Stud. 1993;60(3):531–42. http://www.jstor.org/stable/2298123.
[35] Moffit RA. Policy interventions, low-level equilibria, and social interactions. In: Social dynamics. Cambridge, Massachusetts: The MIT Press; 2001. doi: 10.7551/mitpress/6294.003.0005.
[36] Bramoullé Y, Djebbari H, Fortin B. Identification of peer effects through social networks. J Econometr. 2009;150(1):41–55. https://www.sciencedirect.com/science/article/pii/S0304407609000335.
[37] Hammond DK, Vandergheynst P, Gribonval R. Wavelets on graphs via spectral graph theory. Appl Comput Harmonic Anal. 2011;30(2):129–50. https://www.sciencedirect.com/science/article/pii/S1063520310000552.
[38] Bousquet O, Boucheron S, Lugosi G. In: Bousquet O, vonLuxburg U, Rätsch G, editors. Introduction to statistical learning theory. Berlin, Heidelberg: Springer Berlin Heidelberg; 2004. p. 169–207. doi: 10.1007/978-3-540-28650-9_8.
[39] vonLuxburg U, Schölkopf B. In: Statistical learning theory: models, concepts, and results. vol. 10. Amsterdam, Netherlands: Elsevier North Holland; 2011. p. 651–706.
[40] Tibshirani R. Regression shrinkage and selection via the Lasso. J R Stat Soc Ser B (Methodological). 1996;58(1):267–88. http://www.jstor.org/stable/2346178.
[41] Hamilton W, Ying Z, Leskovec J. Inductive representation learning on large graphs. In: Guyon I, Luxburg UV, Bengio S, Wallach H, Fergus R, Vishwanathan S, et al., editors. Advances in neural information processing systems. Vol. 30. Curran Associates, Inc.; 2017. https://proceedings.neurips.cc/paper/2017/file/5dd9db5e033da9c6fb5ba83c7a7ebea9-Paper.pdf.
[42] Kipf TN, Welling M. Semi-supervised classification with graph convolutional networks. In: International Conference on Learning Representations; 2017. https://openreview.net/forum?id=SJU4ayYgl.
[43] Luo S, Chen Z. Sequential Lasso cum EBIC for feature selection with ultra-high dimensional feature space. J Amer Stat Assoc. 2014;109(507):1229–40. doi: 10.1080/01621459.2013.877275.

[44] Wainwright MJ. High-dimensional statistics: a non-asymptotic viewpoint. Cambridge series in statistical and probabilistic mathematics. Cambridge: Cambridge University Press; 2019.
[45] Raskutti G, Wainwright MJ, Yu B. Restricted eigenvalue properties for correlated Gaussian designs. J Machine Learn Res. 2010;11(78):2241–59. http://jmlr.org/papers/v11/raskutti10a.html.
[46] Bickel PJ, Ritov Y, Tsybakov AB. Simultaneous analysis of Lasso and Dantzig selector. Ann Stat. 2009;37(4):1705–32. doi: 10.1214/08-AOS620.
[47] van de Geer SA, Bü hlmann P. On the conditions used to prove oracle results for the Lasso. Electr J Stat. 2009;3:1360–92. doi: 10.1214/09-EJS506.
[48] Saint-Jacques G, Varshney M, Simpson J, Xu Y. Using ego-clusters to measure network effects at LinkedIn. 2019. https://arxiv.org/abs/1903.08755.
[49] Lee JD, Sun DL, Sun Y, Taylor JE. Exact post-selection inference, with application to the lasso. Ann Stat. 2016;44(3):907–27. doi: 10.1214/15-AOS1371.
[50] Efron B, Tibshirani RJ. An introduction to the Bootstrap. Chapman & Hall/CRC Monographs on Statistics & Applied Probability. New York: Taylor & Francis; 1994. https://books.google.com/books?id=gLlpIUxRntoC.
[51] Cameron AC, Gelbach JB, Miller DL. Bootstrap-based improvements for inference with clustered errors. Rev Econ Stat. 2008 Aug;90(3):414–27. doi: 10.1162/rest.90.3.414.
[52] Basse G, Feller A. Analyzing two-stage experiments in the presence of interference. J Amer Stat Assoc. 2018;113(521):41–55. doi: 10.1080/01621459.2017.1323641.
[53] Calinescu G, Karloff H, Rabani Y. Approximation algorithms for the 0-extension problem. SIAM J Comput. 2005;34(2):358–72.
[54] Sävje F, Aronow P, Hudgens M. Average treatment effects in the presence of unknown interference. Ann Stat. 2021;49(2):673.
[55] Nishimura J, Ugander J. Restreaming graph partitioning: simple versatile algorithms for advanced balancing. In: Proceedings of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. KDD '13. New York, NY, USA: Association for Computing Machinery; 2013. p. 1106–14. doi: 10.1145/2487575.2487696.
[56] Spielman DA, Teng SH. A local clustering algorithm for massive graphs and its application to nearly linear time graph partitioning. SIAM J Comput. 2013;42(1):1–26. doi: 10.1137/080744888.
[57] Awadelkarim A, Ugander J. Prioritized restreaming algorithms for balanced graph partitioning. In: Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining. KDD '20. New York, NY, USA: Association for Computing Machinery; 2020. p. 1877–87. doi: 10.1145/3394486.3403239.
[58] Shi L, Chen B. Comparison and benchmark of graph clustering algorithms. CoRR. 2020;abs/2005.04806. https://arxiv.org/abs/2005.04806.
[59] Traud AL, Mucha PJ, Porter MA. Social structure of Facebook networks. Phys A Stat Mech Appl. 2012;391(16):4165–80. https://www.sciencedirect.com/science/article/pii/S0378437111009186.
[60] Holland PW, Laskey KB, Leinhardt S. Stochastic blockmodels: first steps. Soc Netw. 1983;5(2):109–37.
[61] Airoldi EM, Blei D, Fienberg S, Xing E. Mixed membership stochastic blockmodels. J Mach Learn Res. 2008;9:1981–2014.
[62] Karrer B, Newman ME. Stochastic blockmodels and community structure in networks. Phys Rev E. 2011;83(1):016107.
[63] Abbe E. Community detection and stochastic block models: recent developments. J Machine Learn Res. 2017;18(1):6446–531.
[64] Milgram S. The small world problem. Psychol Today. 1967;2(1):60–7.
[65] Travers J, Milgram S. An experimental study of the small world problem. In: Social networks. Elsevier; 1977. p. 179–97.
[66] Watts DJ, Strogatz SH. Collective dynamics of small-world networks. Nature. 1998;393(6684):440–2.
[67] Newman ME. Models of the small world. J Stat Phys. 2000;101:819–41.
[68] Latora V, Marchiori M. Efficient behavior of small-world networks. Phys Rev Lett. 2001;87(19):198701.

Appendix
A Proofs

The proofs of Propositions 4.1 and 4.2 will be exactly the same as the proofs of Propositions 1 and 2 in ref. [43] once we realize that as long as the features that are included in the penalty do not overlap with the features that have already been selected then we can just use the proofs in the study by Luo and Chen [43], i.e., though our sequential selection procedure is different from that in the study by Luo and Chen [43], we share the same properties that make these two propositions hold.

Proof of Proposition 4.1. We denote by $X(s)$ the design matrix with features in s, i.e., if $|s| = m$, then $X(s)$ is a $n \times m$ matrix. At the $(t+1)$th iteration, β will be a $(|s_{*t}| + i_{t+1})$-dimensional vector, and we denote by $\beta(s)$ the $|s|$-dimensional vector with only coordinates of β that are in s. Finally, we denote by A_{t+1} the set $\{u_1^{t+1}, u_2^{t+1}, \ldots, u_{i_{t+1}}^{t+1}\}$.

First, we note that since $u_j^{t+1} \in \mathcal{R}(s_{*t})$, $\exists v \in \mathbb{R}^{|s_{*t}|}$ such that $u_j^{t+1} = X(s_{*t})v$. We now consider the objective function l_{t+1} at the $(t+1)$th iteration.

$$\begin{aligned} l_{t+1} &= ||y - X(s_{*t})(\beta(s_{*t}) + \beta(\{j\})v) - X(A_{t+1}/\{j\})\beta(A_{t+1}/\{j\})||_2^2 + \lambda(|\beta(\{j\})| + ||\beta(A_{t+1}/\{j\})||_1) \\ &= ||y - X(s_{*t})\tilde{\beta}(s_{*t}) - X(A_{t+1}/\{j\})\beta(A_{t+1}/\{j\})||_2^2 + \lambda(|\beta(\{j\})| + ||\beta(A_{t+1}/\{j\})||_1) \\ &\geq ||y - X(s_{*t})\tilde{\beta}(s_{*t}) - X(A_{t+1}/\{j\})\beta(A_{t+1}/\{j\})||_2^2 + \lambda||\beta(A_{t+1}/\{j\})||_1 \end{aligned}$$

Hence, when l_{t+1} is minimized, $\beta(\{j\})$ must be 0 and $j \notin s_{*(t+1)}$. □

Proof of Proposition 4.2. Again we consider the objective function at the $(t+1)$th iteration.

$$l_{t+1} = ||y - X(s_{*t})\beta(s_{*t}) - X(A_{t+1})\beta(A_{t+1})||_2^2 + \lambda||\beta(A_{t+1})||_1.$$

Differentiating l_{t+1} with respect to $\beta(s_{*t})$, we have

$$\frac{\partial l_{t+1}}{\partial \beta(s_{*t})} = -2X^T(s_{*t})y + 2X^T(s_{*t})X(s_{*t})\beta(s_{*t}) + 2X^T(s_{*t})X(A_{t+1})\beta(A_{t+1}).$$

Setting the aforementioned derivative to zero, we have that

$$\hat{\beta}(s_{*t}) = [X^T(s_{*t})X(s_{*t})]^{-1}X^T(s_{*t})[y - X(A_{t+1})\beta(A_{t+1})]. \tag{A1}$$

By substituting (A1) into the objective function, we obtain

$$\begin{aligned} l_{t+1} &= ||y - X(s_{*t})\beta(s_{*t}) - X(A_{t+1})\beta(A_{t+1})||_2^2 + \lambda||\beta(A_{t+1})||_1 \\ &= ||y - X(s_{*t})[X^T(s_{*t})X(s_{*t})]^{-1}X^T(s_{*t})[y - X(A_{t+1})\beta(A_{t+1})] - X(A_{t+1})\beta(A_{t+1})||_2^2 + \lambda||\beta(A_{t+1})||_1 \\ &= ||(I - X(s_{*t})[X^T(s_{*t})X(s_{*t})]^{-1}X^T(s_{*t}))y - (I - X(s_{*t})[X^T(s_{*t})X(s_{*t})]^{-1}X^T(s_{*t}))X(A_{t+1})\beta(A_{t+1})||_2^2 \\ &\quad + \lambda||\beta(A_{t+1})||_1. \end{aligned}$$

Hence, minimizing l_{t+1} does not affect $\hat{\beta}(s_{*t})$ and $\hat{\beta}(s_{*t})$ will be almost surely nonzero. □

Now we show the proof of Theorem 4.3. We will make use of standard results about LASSO ℓ_2-error bounds. Recall the following result [44]:

Lemma A.1. *Suppose $y = X\theta^* + w$ $(X \in \mathbb{R}^{n\times d})$ and consider the Lagrangian Lasso with a strictly positive regularization parameter $\lambda_n \geq 2||\frac{X^T w}{n}||_\infty$. Suppose further that θ^* is supported on a subset S of cardinality s, and the design matrix satisfies the $(\kappa;3)$-RE condition over S, then*

$$||\hat{\theta} - \theta^*||_2 \leq \frac{3}{\kappa}\sqrt{s}\lambda_n.$$

We can show that if the design matrix is C-column normalized, i.e.,

$$\max_{j=1,\dots,d} \frac{\|X_j\|_2}{\sqrt{n}} \le C,$$

then the choice $\lambda_n = 2C\sigma(\sqrt{\frac{2\log d}{n}} + \delta)$ is valid with probability at least $1 - 2e^{-\frac{n\delta^2}{2}}$. We thus proceed with the main proof.

Proof. Notice that $\|\frac{\mathbf{X}^T w}{n}\|_\infty$ corresponds to the absolute maximum of d zero-mean Gaussian random variables by definition of infinity norm and each with variance at most $\frac{C^2\sigma^2}{n}$. Hence, from the Gaussian tail bound, we then have

$$\mathbb{P}\left(\left\| \frac{\mathbf{X}^T w}{n} \right\|_\infty \ge C\sigma\left(\sqrt{\frac{2\log d}{n}} + \delta\right)\right) \le 2e^{-\frac{n\delta^2}{2}}. \qquad \square$$

With this particular choice of λ_n, the lemma implies the upper bound

$$\|\hat{\theta} - \theta^*\|_2 \le \frac{6C\sigma}{\kappa}\sqrt{s}\left(\sqrt{\frac{2\log d}{n}} + \delta\right) \tag{A2}$$

with the same high probability [44].

Now we are ready to prove consistency. First notice that

$$\begin{aligned}
|\hat{\tau} - \tau| &= \left| \frac{1}{n}\sum_{i=1}^{n}[(\hat{\beta}_1 - \beta_1^*)^T u_i^{gt} - (\hat{\beta}_0 - \beta_0^*)^T u_i^{gc}] \right| \\
&\le \frac{1}{n}\sum_{i=1}^{n}|[(\hat{\beta}_1 - \beta_1^*)^T u_i^{gt} - (\hat{\beta}_0 - \beta_0^*)^T u_i^{gc}]| \\
&\le \frac{1}{n}\sum_{i=1}^{n}(\|\hat{\beta}_1 - \beta_1^*\|_2\|u_i^{gt}\|_2 + \|\hat{\beta}_0 - \beta_0^*\|_2\|u_i^{gc}\|_2) \\
&\le C\sqrt{M}(\|\hat{\beta}_1 - \beta_1^*\|_2 + \|\hat{\beta}_0 - \beta_0^*\|_2).
\end{aligned}$$

Let n_0 be the number of control units and n_1 be the number of treated units. Then by strong law of large numbers, $\frac{n_0}{n} \xrightarrow{a.s.} 1 - p$ and $\frac{n_1}{n} \xrightarrow{a.s.} p$. Since the design matrices U^0 and U^1 satisfy the RE condition, both $\|\hat{\beta}_1 - \beta_1^*\|_2$ and $\|\hat{\beta}_0 - \beta_0^*\|_2$ converge to 0 in probability by the bound (A2). Thus, $\hat{\tau} \xrightarrow{\mathbb{P}} \tau$.

Proof of Proposition 4.5. We show that for the setup in Proposition 4.5, the design matrices satisfy RE condition with probability going to 1. In our proof, the first column of the design matrix represents the fraction of treated neighbors, while the second column represents the number of treated neighbors. We introduce one extra notations: for each unit i, we denote by m_i the size of the cluster unit i belongs to. We show the proof for the design matrix for control units, U^0. Similar proof can be done for U^1. After centering, the design matrix we use for estimating β_0 will be

$$\tilde{U}^0 = \begin{bmatrix} \frac{1}{n_0}\sum_{i:W_i=0}(u_i^1 - \bar{u}^1)^2 & \frac{1}{n_0}\sum_{i:W_i=0}(u_i^1 - \bar{u}^1)(u_i^2 - \bar{u}^2) \\ \frac{1}{n_0}\sum_{i:W_i=0}(u_i^1 - \bar{u}^1)(u_i^2 - \bar{u}^2) & \frac{1}{n_0}\sum_{i:W_i=0}(u_i^2 - \bar{u}^2)^2 \end{bmatrix}.$$

Here, $\bar{u}^1 = \frac{1}{n_0}\sum_{i:W_i=0}u_i^1$ and $\bar{u}^2 = \frac{1}{n_0}\sum_{i:W_i=0}u_i^2$. Since the true β_0 is nonzero only for the first feature, $\mathbb{C}_3(S) = \{\Delta \in \mathbb{R}^2 : |\Delta_2| \le 3|\Delta_1|\}$. For such Δ, we have that

$$\frac{1}{n_0}\|\tilde{U}^0\Delta\|_2^2 = \Delta_1^2\frac{1}{n_0}\sum_{i:W_i=0}(u_i^1-\bar{u}^1)^2 + 2\Delta_1\Delta_2\frac{1}{n_0}\sum_{i:W_i=0}(u_i^1-\bar{u}^1)(u_i^2-\bar{u}^2) + \Delta_2^2\frac{1}{n_0}\sum_{i:W_i=0}(u_i^2-\bar{u}^2)^2.$$

Note that since $|\Delta_2| \le 3|\Delta_1|$, $\Delta_1\Delta_2 \ge -|\Delta_1||\Delta_2| \ge -\frac{1}{3}\Delta_2^2$. Therefore,

$$\frac{1}{n_0}\|\tilde{U}^0\Delta\|_2^2 \ge \frac{1}{n_0}\sum_{i:W_i=0}(u_i^1-\bar{u}^1)^2\Delta_1^2 + \left(\frac{1}{n_0}\sum_{i:W_i=0}(u_i^2-\bar{u}^2)^2 - \frac{1}{3}\frac{1}{n_0}\sum_{i:W_i=0}(u_i^1-\bar{u}^1)(u_i^2-\bar{u}^2)\right)\Delta_2^2. \tag{A3}$$

To ease notations, we let ① $=\frac{1}{n_0}\sum_{i:W_i=0}(u_i^1-\bar{u}^1)^2$, ② $=\frac{1}{n_0}\sum_{i:W_i=0}(u_i^2-\bar{u}^2)^2$, and ③ $=\frac{1}{n_0}\sum_{i:W_i=0}(u_i^1-\bar{u}^1)(u_i^2-\bar{u}^2)\Delta_2^2$. Now, we analyze each term separately.

$$\begin{aligned}
① &= \frac{1}{n_0}\sum_{i:W_i=0}(u_i^1-\bar{u}^1)^2 \\
&= \frac{1}{n_0}\sum_{i:W_i=0}(u_i^1)^2 - (\bar{u}^1)^2 \\
&= \frac{n}{n_0}\frac{1}{n}\sum_{i=1}^n(1-W_i)(u_i^1)^2 - (\bar{u}^1)^2 \\
&= \frac{n}{n_0}\frac{1}{n}\sum_{i=1}^n(1-W_i)(u_i^1)^2 - \left(\frac{n}{n_0}\frac{1}{n}\sum_{i=1}^n(1-W_i)u_i^1\right)^2.
\end{aligned}$$

Consider the random variables $\{(1-W_i)(u_i^1)^2\}_{i=1}^n$ and $\{(1-W_i)u_i^1\}_{i=1}^n$. Since we have disjoint clusters and the number of units in each cluster is bounded by M, the sum of covariance term is at most $O(n)$, and hence, weak law of large numbers applies for both sequences. Therefore,

$$\frac{1}{n}\sum_{i=1}^n(1-W_i)(u_i^1)^2 - \left[p(1-p)^2\frac{1}{n}\sum_{i=1}^n\frac{1}{m_i-1} + p^2(1-p)\right] \xrightarrow{\mathbb{P}} 0.$$

Similarly,

$$\frac{1}{n}\sum_{i=1}^n(1-W_i)u_i^1 \xrightarrow{\mathbb{P}} p(1-p).$$

Note that $n/n_0 \xrightarrow{\mathbb{P}} 1/(1-p)$, we obtain

$$① - \left[p(1-p)\frac{1}{n}\sum_{i=1}^n\frac{1}{m_i-1}\right] \xrightarrow{\mathbb{P}} 0.$$

Here, ② can be done similarly:

$$② - \left[p(1-p)\frac{1}{n}\sum_{i=1}^n(m_i-1) + p^2\frac{1}{n}\sum_{i=1}^n(m_i-1)^2 - p^2\left(\frac{1}{n}\sum_{i=1}^n(m_i-1)\right)\right] \xrightarrow{\mathbb{P}} 0.$$

For ③, we have that

$$③ = \frac{1}{n_0}\sum_{i:W_i=0}(u_i^1-\bar{u}^1)(u_i^2-\bar{u}^2) = \frac{1}{n_0}\sum_{i:W_i=0}u_i^1u_i^2 - \bar{u}^1\bar{u}^2.$$

Notice that we have already shown that

$$\bar{u}^1 \xrightarrow{\mathbb{P}} p, \quad \bar{u}^2 \xrightarrow{\mathbb{P}} p\frac{1}{n}\sum_{i=1}^n(m_i-1).$$

Hence, $\bar{u}^1\bar{u}^2 \xrightarrow{\mathbb{P}} p^2\frac{1}{n}\sum_{i=1}^n(m_i-1)$. Moreover,

$$\frac{1}{n_0}\sum_{i:W_i=0}u_i^1u_i^2 = \frac{n}{n_0}\frac{1}{n}\sum_{i=1}^n(1-W_i)u_i^1u_i^2.$$

Again by weak law of large numbers,

$$\frac{1}{n}\sum_{i=1}^{n}(1-W_i)u_i^1u_i^2-\left[p(1-p)^2+p^2(1-p)\frac{1}{n}\sum_{i=1}^{n}(m_i-1)\right]\xrightarrow{\mathbb{P}}0.$$

Hence, ③ $-\left[p(1-p)+p^2\frac{1}{n}\sum_{i=1}^{n}(m_i-1)\right]\xrightarrow{\mathbb{P}}0$. Put all these pieces together, we obtain

$$\text{RHS of (12)}-\left\{\left[p(1-p)\frac{1}{n}\sum_{i=1}^{n}\frac{1}{m_i-1}\right]\Delta_1^2+\left[p(1-p)\frac{1}{n}\sum_{i=1}^{n}(m_i-1)+p^2\frac{1}{n}\sum_{i=1}^{n}(m_i-1)^2-p^2\left(\frac{1}{n}\sum_{i=1}^{n}(m_i-1)\right)\right.\right.$$
$$\left.\left.-\frac{1}{3}\left(p(1-p)+p^2\frac{1}{n}\sum_{i=1}^{n}(m_i-1)\right)\right]\Delta_2^2\right\}\xrightarrow{\mathbb{P}}0.$$

Notice that $m_i\geq 3$ and $m_i\leq M$ for each i, we conclude that for $\kappa=\min\{\frac{p(1-p)}{M-1},\frac{5}{3}p-\frac{1}{3}p^2\}$,

$$\frac{1}{n_0}||\tilde{U}^0\Delta||_2^2\geq\kappa||\Delta||_2^2\quad\text{w.p.}\ \to 1.$$ □

B Supplementary materials

For completeness, we give a full definition of the nonlinear model used in the main text.

Definition B.1. (The nonlinear model in simulations) Suppose the assignment vector is w, then for each unit i, the response is

$$y_i(w)=-5+2z_iw_i+0.03v_i+\frac{1}{1+0.001\exp(-0.03v_i+9)}+\frac{10}{3+\exp(-8\rho_i+3.2)}+\varepsilon_i.$$

Here, $z_i,\varepsilon_i\overset{\text{i.i.d}}{\sim}\mathcal{N}(0,1)$, ρ_i is the fraction of treated neighbors for unit i and v_i is the number of treated neighbors for unit i.

B.1 Additional simulation results

In this section, we present further simulation results with distinct network generating processes, specifically considering the stochastic block model and the small-world network. These two network models have been widely explored in previous studies. For the stochastic block models, see [60–63]. For the small-world networks, refer to the previous studies [64–68]. Since the networks here are less dense compared to the Swarthmore college network, we use Models 2a and 2b instead of Models 1 and 2.

In Tables A1 and A2, we present the RMSE and bias of various estimators under linear interference for a network generated using a stochastic block model. Specifically, we maintain a fixed number of nodes at 1,600 and four distinct communities, comprising 300, 600, 250, and 450 members, respectively. The probability matrix P governing the formation of edges between vertices from two different groups is:

$$P=\begin{bmatrix}0.05&0.01&0.02&0.03\\0.01&0.05&0.01&0.02\\0.02&0.01&0.05&0.03\\0.03&0.02&0.03&0.05\end{bmatrix}.$$

The true GATEs for the different models are 2.00, 6.12, 11.69, 15.00, 20.00, 15.00, and 35.00. The results for the Hájek estimator are absent due to the fact that in this particular scenario, no unit has more than 80% or less than 20% of treated neighbors. This, in turn, underscores a significant limitation associated with the Hájek estimator. Specifically, a Hájek estimator under such a fractional neighborhood exposure condition may not be well defined in situations where data are collected from a randomized experiment on a network with a Bernoulli design.

Table A1: RMSE of estimators of the GATE assuming linear interference (simple linear interference and truncated linear-in-means) outcome models. The underlying network is generated from a stochastic block model

RMSE, Stochastic block model							
Estimator	$\hat{\tau}^{\mathrm{DM}}$	$\hat{\tau}^{\mathrm{Hájek}}_{0.8,0.2}$	$\hat{\tau}_{\mathrm{frac}}$	$\hat{\tau}_{\mathrm{num}}$	**Post-ReFeX-LASSO**	**ReFeX-LASSO**	$\hat{\tau}_{\mathrm{oracle}}$
Model 0	0.05	NA	0.33	0.22	1.83	1.24	0.05
Model 2a	2.78	NA	0.32	1.23	1.92	2.10	0.32
Model 2b	6.46	NA	0.41	1.22	2.75	2.22	0.32
Model 3	10.00	NA	0.36	6.11	2.26	2.46	0.36
Model 4	15.01	NA	0.42	9.15	2.52	2.62	0.42
Model 5	9.89	NA	4.86	7.95	2.47	2.45	2.45
Model 6	29.58	NA	19.57	25.67	2.59	2.58	2.62

Table A2: Empirical bias of estimators of the GATE assuming linear interference (simple linear interference and truncated linear-in-means) outcome models. The underlying network is generated from a stochastic block model

Bias, stochastic block model							
Estimator	$\hat{\tau}^{\mathrm{DM}}$	$\hat{\tau}^{\mathrm{Hájek}}_{0.8,0.2}$	$\hat{\tau}_{\mathrm{frac}}$	$\hat{\tau}_{\mathrm{num}}$	**Post-ReFeX-LASSO**	**ReFeX-LASSO**	$\hat{\tau}_{\mathrm{oracle}}$
Model 0	–0.001	NA	–0.013	0.006	0.573	–0.028	–0.001
Model 2a	–2.784	NA	–0.035	–1.21	0.352	0.043	–0.010
Model 2b	–6.456	NA	–0.141	–1.200	0.379	0.815	0.019
Model 3	–10.001	NA	–0.007	–6.102	0.163	–0.022	–0.007
Model 4	–15.008	NA	–0.008	–9.142	0.388	0.287	–0.008
Model 5	–9.892	NA	–4.850	–7.944	0.027	0.011	–0.012
Model 6	–29.575	NA	–19.569	–25.663	0.022	–0.011	0.059

Tables A3 and A4 show the RMSE and bias of several different estimators under linear interference with a network generated from the small-world network. Specifically, a Watts-Strogatz model was utilized to generate a network consisting of 1,500 vertices, with a rewiring probability of 0.1 and a neighborhood size of 15. The true GATEs under the seven models are 2.00, 5.75, 9.50, 15.00, 20.00, 15.00, and 35.00.

Upon examination of these four results tables, it is evident that our estimators exhibit consistent performance across a broad range of model configurations for the two distinct network models. These supplementary simulations underscore the robustness of our estimators irrespective of the underlying network generating process.

Table A3: RMSE of estimators of the GATE assuming linear interference (simple linear interference and truncated linear-in-means) outcome models. The underlying network is generated from a Watts–Strogatz model of small worlds

RMSE, Watts–Strogatz model							
Estimator	$\hat{\tau}^{\mathrm{DM}}$	$\hat{\tau}^{\mathrm{Hájek}}_{0.8,0.2}$	$\hat{\tau}_{\mathrm{frac}}$	$\hat{\tau}_{\mathrm{num}}$	**Post-ReFeX-LASSO**	**ReFeX-LASSO**	$\hat{\tau}_{\mathrm{oracle}}$
Model 0	0.05	NA	0.31	0.29	0.12	0.43	0.05
Model 2a	2.53	NA	0.29	0.42	1.49	0.58	0.29
Model 2b	5.01	NA	0.32	0.43	1.45	0.72	0.32
Model 3	10.02	NA	0.42	1.74	0.48	0.58	0.42
Model 4	15.02	NA	0.56	2.47	0.66	0.79	0.56
Model 5	9.84	NA	3.10	4.19	0.90	1.45	0.81
Model 6	29.35	NA	12.29	15.05	1.69	1.91	1.53

Table A4: Empirical bias of estimators of the GATE assuming linear interference (simple linear interference and truncated linear-in-means) outcome models. The underlying network is generated from a Watts–Strogatz model of small-worlds

Bias, Watts-Strogatz model							
Estimator	$\hat{\tau}^{\mathrm{DM}}$	$\hat{\tau}^{\mathrm{Hájek}}_{0.8,0.2}$	$\hat{\tau}_{\mathrm{frac}}$	$\hat{\tau}_{\mathrm{num}}$	**Post-ReFeX-LASSO**	**ReFeX-LASSO**	$\hat{\tau}_{\mathrm{oracle}}$
Model 0	0.000	NA	0.002	−0.006	−0.004	0.034	0.000
Model 2a	−2.526	NA	0.013	−0.315	0.018	−0.035	0.014
Model 2b	−5.005	NA	−0.018	−0.316	−0.052	−0.115	0.003
Model 3	−10.018	NA	−0.049	−1.690	−0.073	−0.067	−0.049
Model 4	−15.016	NA	0.054	−2.409	0.051	0.040	0.054
Model 5	−9.838	NA	−3.069	−4.164	−0.130	−0.185	0.003
Model 6	−29.344	NA	−12.254	−15.016	−0.190	−0.245	0.047

Finally, we conclude this section with simulation results for the farmers network [21]. As previously emphasized in the main body of this article, the farmers network is less dense than the Swarthmore college network, which simplifies the estimation process. We also use the Models 2a and 2b as in Section 5.4 since the network is relatively sparse. The RMSE and empirical bias of various estimators under different model specifications are presented in Tables A5 and A6. To reiterate, our estimators exhibit consistent performance across all model specifications. Furthermore, our estimators occasionally outperform the oracle regression adjustment estimator in terms of RMSE due to more stable variance.

Table A5: RMSE of estimators of the GATE assuming linear interference (simple linear interference and truncated linear-in-means) outcome models. The underlying network is the farmers network in ref. [21]

RMSE, farmers network							
Estimator	$\hat{\tau}^{\mathrm{DM}}$	$\hat{\tau}^{\mathrm{Hájek}}_{0.8,0.2}$	$\hat{\tau}_{\mathrm{frac}}$	$\hat{\tau}_{\mathrm{num}}$	**Post-ReFeX-LASSO**	**ReFeX-LASSO**	$\hat{\tau}_{\mathrm{oracle}}$
Model 0	0.03	0.16	0.07	0.05	0.03	0.03	0.03
Model 2a	2.12	0.54	0.07	1.09	0.07	0.07	0.07
Model 2b	2.69	0.64	0.17	1.09	0.12	0.17	0.07
Model 3	9.98	2.19	0.12	5.43	0.12	0.12	0.12
Model 4	14.97	3.27	0.18	8.15	0.19	0.18	0.18
Model 5	9.10	4.32	3.33	6.35	0.20	0.20	0.22
Model 6	26.45	15.20	13.37	20.02	0.45	0.46	0.53

Table A6: Empirical bias of estimators of the GATE assuming linear interference (simple linear interference and truncated linear-in-means) outcome models. The underlying network is the farmers network in ref. [21]

Bias, farmers network							
Estimator	$\hat{\tau}^{\mathrm{DM}}$	$\hat{\tau}^{\mathrm{Hájek}}_{0.8,0.2}$	$\hat{\tau}_{\mathrm{frac}}$	$\hat{\tau}_{\mathrm{num}}$	**Post-ReFeX-LASSO**	**ReFeX-LASSO**	$\hat{\tau}_{\mathrm{oracle}}$
Model 0	0.000	0.015	−0.004	−0.002	0.000	0.000	0.000
Model 2a	−2.119	−0.524	−0.027	−1.089	−0.027	−0.027	−0.002
Model 2b	−2.694	−0.611	−0.150	−1.085	0.013	−0.011	0.001
Model 3	−9.975	−2.178	0.000	−5.427	0.000	0.000	0.000
Model 4	−14.966	−3.255	2.179	−8.146	0.022	0.022	−0.009
Model 5	−9.101	−4.312	−3.323	−6.349	0.036	0.034	0.085
Model 6	−26.452	−15.187	−13.362	−20.016	0.072	0.085	0.251