Model-Based Regression Adjustment with Model-Free Covariates for Network Interference

Kevin Han
Department of Statistics, Stanford University
and
Johan Ugander
Department of Management Science and Engineering, Stanford University

April 22, 2023

Keywords: Causal Inference with Interference, SUTVA, A/B Testing, Regression Adjustment, Social Network Analysis

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 squared 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.

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 assumption of no interference assumption (Cox 1958, Chapter 2) or the stable unit treatment value assumption (SUTVA) Rubin (1974). However, when experimental units interact with each other, SUTVA is often untenable. Violation of SUTVA has been found in many applications, including politics Sinclair et al. (2012), education Hong & Raudenbush (2006), Rosenbaum (2007), economics Sobel (2006), Manski (2013), and public health Halloran & Struchiner (1995). Recently, technology companies developing products with social or market interactions have developed methods to manage the considerable interference in their product experiments Eckles et al. (2017), Pouget-Abadie et al. (2019), Karrer et al. (2021). 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 Aronow & Samii (2017) 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 Lin (2013) and practice Deng et al. (2013). Chin Chin (2019) considers regression adjustment under interference when assuming a linear model for the outcomes, and estimate 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 Harshaw et al. (2022) and interference detection Pouget-Abadie et al. (2019). There has also been literature on new designs that tackle the complication of interference. For example, Ugander et al. (2013) and Ugander & Yin (2020) 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 by using regression adjustment without assuming the true set of features as in Chin (2019). 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 global average treatment effect with model-free covariates. We conclude with simulations, an empirical applications, and a discussion.

2 Setup

Consider a randomized experiment on n units where these 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 pre-treatment covariates for unit i. We assume that the experimental population is the population of interest and hence view pre-treatment covariates as fixed. We only consider binary treatments but note that extensions to non-binary treatments are straightforward. Throughout, we use lower case letters with the appropriate subscript for realizations of the random variables and for non-random quantities.

We work under the Rubin causal model Rubin (1974), Holland (1986), Imbens & Rubin (2015). For every unit i, we associate it with potential outcomes $Y_i(w) \in \mathbb{R}$ for $w \in \{0,1\}^n$. We are interested in the following causal estimand that we call the Global Average Treatment Effect (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 some work Yu et al. (2022), measures the overall effect of the intervention on the experimental units. Under SUTVA, the assignments of other units won't 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 $w \in \{0,1\}^n$,

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

with ϵ_i's being exogenous, i.e. $\mathbb{E}[\epsilon_i|w] = 0$. 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 Aronow & Samii (2017) 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 Basse & Airoldi (2018) has a discussion from an inference perspective.

Example 1 (Linear-in-means model). Consider the structural model Manski (1993), Moffit (2001), Bramoullé et al. (2009)

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

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. Bramoullé et al. (2009) show that under some mild conditions on the coefficients and the graph G, we can rewrite the above model as

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

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 $\mathbb{E}[\sum_{j=0}^{\infty}\beta^j\tilde{A}^{j+1}\boldsymbol{\epsilon}|w] = 0$. 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 Hammond et al. (2011). Such a linear interference assumption is not uncommon Deng et al. (2013), Pouget-Abadie et al. (2019), Chin (2019). Chin Chin (2019) shows how to do inference once we have access to the oracle g while Pouget-Abadie et al. (2019) 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 Bousquet et al. (2004), von Luxburg & Schölkopf (2011), 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

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

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 pre-treatment covariates x_i in our discussion. Extensions to the case of including pre-treatment 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. Karrer et al. (2021). As an alternative, estimates from designs that accounts for network interference (for example, graph cluster randomization) may suffer from sizable variance Ugander & Yin (2020). 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 Chin (2019) provides a complete solution. However, if we don't 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 below and then explain the steps in more detail. We call the procedure ReFeX-LASSO as it builds on the graph mining technique ReFeX Henderson et al. (2011) to generate candidate features while using LASSO Tibshirani (1996) to select features.

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 (GCNs) Hamilton et al. (2017), Kipf & Welling (2017). 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.

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 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 pre-treatment 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 pre-treatment covariates and G. With ReFeX, the base features are chosen by the analyst. 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 Henderson et al. (2011).

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). 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 above 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 Henderson et al. (2011), 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 get 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 below 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 two-fold. 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 $u_i^1, \cdots, u_i^K$ denote the K covariates returned by ReFeX-LASSO or post-ReFeX-LASSO for unit i and let $u_i = \left[u_i^1, \cdots, u_i^K\right]^T \in \mathbb{R}^K$ 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.

Combing 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 helps 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 prior work on Sequential LASSO Luo & Chen (2014). To this end, we introduce some additional notation. For each iteration t, let $\{u_1^t, u_2^t, \cdots, u_{i_t}^t\}$ be the set

[2]In fact, in the case of no interference, Lin Lin (2013) shows that doing linear adjustment can only improve the precision.

of features generated in the ReFeX step of ReFeX-LASSO and s_{*t} be the selected features at the t-th iteration (note that s_{*t} may contain features that were selected in previous iterations and thus are not in the set $\{u_1^t, u_2^t, \cdots, u_{i_t}^t\}$). Moreover, we let $\mathcal{R}(s)$ to denote the space spanned by features in s.

Proposition 4.1. For $t \geq 1$ and any $j \in \{1, \cdots, i_{t+1}\}$, 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 $s_{*1} \subseteq s_{*2} \subseteq \cdots \subseteq s_{*T}$.

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 Luo & Chen (2014), the proofs of the above two propositions are analogous to those in Luo & Chen (2014). 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 non-selected 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 Luo & Chen (2014) 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 $\{u_i^1, \cdots, u_i^M\}$. We drop the subscript i when we refer to the jth feature vector, i.e., $u^j = [u_1^j, \cdots, u_n^j]^T$. Furthermore, we assume that there exists a subset $S_* \subset \{u^1, \cdots, u^M\}$ 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*

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

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 Wainwright (2019). The restricted eigenvalue (RE) condition in Theorem 4.3 is a standard assumption when proving ℓ_2-error bound on the coefficient vector. It restricts the curvature for a specific subset of vectors in the Euclidean space. It is defined as follows Bickel et al. (2009), van de Geer & Bühlmann (2009), Raskutti et al. (2010)

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

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

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 ν_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 since when all clusters have size 2 then ρ_i and ν_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 co-linearity so in that case we wouldn't consider distinguishing between these two features. While the above 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 Saint-Jacques et al. (2019), Karrer et al. (2021). The above 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 Chin (2019) where one assumes an oracle model, here the true model is unknown. Second, features constructed in Chin (2019) 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 Lee et al. (2016). Lee et al. Lee et al. (2016) consider 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 just 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 don't 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 Efron (1979), Efron & Tibshirani (1994), Cameron et al. (2008). 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 Kojevnikov (2021) demonstrate that there is a version of block bootstrap that does provide theoretical guarantee for certain highlu stylized network processes.

Example 3. Consider the case where our social network G consists of C disjoint cliques $\mathcal{C}_1, \cdots, \mathcal{C}_C$ 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 Basse & Feller (2018). In such a case it is natural to consider sampling all C cliques with replacement to get a bootstrap sample. For network dependent processes satisfying certain technical assumptions, this sampling process is the correct thing to do using arguments in Kojevnikov (2021). Suppose we have a network dependent process $\{Y_n, G_n\}$ that satisfies assumptions in Kojevnikov (2021). To make block bootstrap consistent, i.e., producing a confidence interval that is consistent in level, Assumption 4.1 in Kojevnikov (2021) 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 $\forall s_n \geq \max_c \operatorname{diam}(\mathcal{C}_c)$, since our graph consists of non-overlapping 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 Kojevnikov (2021) 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 Kojevnikov (2021).

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 Ugander & Yin (2020), a simple adaptation of the CKR partitioning algorithm Calinescu et al. (2005). 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 above 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 $\mathcal{C}_1, \cdots, \mathcal{C}_c$.
1: for $i \in V$ do
2:     $X_i \leftarrow \mathcal{U}(0,1)$;
3: end for
4: for $i \in V$ do
5:     $i \leftarrow \text{argmax}([X_j \text{ for } j \in B_k(i)])$;
6: end for
7: Return $\mathcal{C}_1, \cdots, \mathcal{C}_c$.
```

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 (since if no feature got selected in the $(T^* + 1)$-th iteration then interference should happen within $(T^* + 1)$-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 $\tau$.
 1: Collect the assignment $w_i$, features $u_i^1, \cdots, u_i^M$ 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
    $\mathcal{C}_1, \cdots, \mathcal{C}_C$.
 3: for $b = 1$ to $B$ do
 4:     Sample $C$ clusters with replacement from $\mathcal{C}_1, \cdots, \mathcal{C}_C$.
 5:     Construct the $b$-th bootstrap sample $(w^b, u^{1,b}, \cdots, u^{M,b}, y^b)$ 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 $\ell$ 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 $\hat{\tau}^1, \cdots, \hat{\tau}^{\ell B}$.
11: Return $\left[q^*_{\alpha/2}, q^*_{1-\alpha/2}\right]$ as the $(1 - \alpha) \times 100\%$ confidence interval for $\tau$.
```

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.

In the above two algorithms, we utilize a randomized graph clustering algorithm that

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 $\tau$.
 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 $\mathcal{C}_1, \cdots, \mathcal{C}_C$.
 3: for $b = 1$ to $B$ do
 4:     Sample $C$ clusters with replacement from $\mathcal{C}_1, \cdots, \mathcal{C}_C$.
 5:     Construct the $b$-th bootstrap sample with units from sampled clusters.
 6:     Rerun ReFeX-LASSO with the original sample for feature generation and the boot-
        strap sample for feature selection.
 7:     Use the covariates returned from last step as well as the bootstrap sample to get
        estimate of $\tau$, $\hat{\tau}^b$.
 8: end for
 9: Repeat line 2-8 for $\ell$ 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 $\hat{\tau}^1, \cdots, \hat{\tau}^{\ell B}$.
11: Return $\left[q^*_{\alpha/2}, q^*_{1-\alpha/2}\right]$ as the $(1-\alpha) \times 100\%$ confidence interval for $\tau$.
```

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 Nishimura & Ugander (2013), Spielman & Teng (2013), Awadelkarim & Ugander (2020), Shi & Chen (2020) 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 (2021). Though we are not considering the same problem as in Kojevnikov (2021), given that we have a more complicated setup, we do not expect that weaker assumptions than those in Kojevnikov (2021) would be sufficient for good coverage in our case. Therefore, we view assumptions in Kojevnikov (2021) as what we should expect to have in order to make our block bootstrap consistent.

The first scenario that we consider is when we have $O(n)$ clusters with non-constant sizes. Then the second absolute central moment of block sizes may be non-vanishing 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 Kojevnikov (2021). 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 Kojevnikov (2021) is certainly violated. In general, we don't 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 Kojevnikov (2021) 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 Algorithm 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 Traud et al. (2012) of real-world social networks. The networks in this dataset are complete online friendship networks for one hundred 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. We use this network to demonstrate that even for such a network, we are still able to get 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 Table 1 and 2. We will discuss in detail about these outcome models in Section 5.2. Our 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 in Section 5.1 and demonstrate that it has good performance in terms of root mean squared error. 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 Manski (2013). Difference-in-mean estimator calculate 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

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

Here we will consider a version of Hájek estimator that accounts for interference. Manski Manski (2013) 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. Ugander et al. (2013) 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 are 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:

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

and

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

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

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

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 one has is usually quite large and under independent Bernoulli assignment we wouldn't 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 describing the outcome models we use in our simulation study. We carry forward the notation from 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 Pouget-Abadie et al. (2019) and Chin (2019). 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 $J > 1$.

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

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

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

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

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 respectively.

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 echos 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.

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

[3]We document this model in the Appendix B.

Estimator	$\hat{\tau}^{\mathrm{DM}}$	$\hat{\tau}^{\text{Hájek}}_{0.8,0.2}$	$\hat{\tau}_{\mathrm{frac}}$	$\hat{\tau}_{\mathrm{num}}$	post-ReFeX-LASSO	ReFeX-LASSO	$\hat{\tau}_{\mathrm{oracle}}$
Model 0	0.004	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

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

5.3 Simulation results

We study both the bias and the root mean squared error (RMSE) of each estimator under these varied models. Table 3 and Table 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. $\hat{\tau}_{\mathrm{frac}}$ is the regression adjustment estimator that adjusts for the fraction of treated neighbors and $\hat{\tau}_{\mathrm{num}}$ adjusts for the number of treated neighbors. They are also considered in Chin (2019). We also show the oracle adjustment estimator $\hat{\tau}_{\mathrm{oracle}}$ as a reference, which marks the best we can do with full knowledge of the 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 $\hat{\tau}^{\text{Hájek}}_{0.8,0.2}$ 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 Model 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. 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 below shows our results there. In this case, $\hat{\tau}_{\text{frac}}$ and $\hat{\tau}_{\text{num}}$ 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 $\hat{\tau}_{\text{frac}}$, $\hat{\tau}_{\text{num}}$ 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.

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
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

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

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 Chin (2019) for some network features ($\hat{\tau}_{\text{frac}}$ and $\hat{\tau}_{\text{num}}$ 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 Cai et al. (2015) where we have a larger and sparser network compared to the Swarthmore College network. In fact, the average size of 2-hop neighborhoods in Swarthmore network is 1092.65 and the average size of 3-hop neighborhoods in 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 Cai et al. (2015) 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 Model 3-6 as well as Model 2a and 2b. We fix $\ell = 3$, $B = 100$ and the coverage is calculated by repeating the whole process 100 times. Table 7 and 8 show

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

the coverage and the average length of the 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 get 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

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

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 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	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

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

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 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 Tablr 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.

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

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.

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 Cai et al. (2015). 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. (2015) found that the diffusion of insurance knowledge drove network effects in product adoption. Hence, this data is 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 Aronow & Samii (2017) is difficult. Hence, analysis done in Chin (2019) is limited since there only four pre-specified 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 Cai et al. (2015) the intensive information sessions were offered in two separate rounds, leading to four separate treatment arms. For our purpose, following Chin (2019), we simplify the experiment by viewing the two intensive information sessions as the same treatment arm. Hence, we reduce the original field experiment to a binary randomized experiment. As in Cai et al. (2015), 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 Chin (2019).

Table 10 shows the resulting GATE estimates, where $\hat{\tau}_{\text{chin}}$ is the estimator in Chin (2019) that adjusts for four covariates: the fraction of treated neighbors, the number of treated

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

Table 10: Estimates and standard errors of different estimators for the global average treatment effect on insurance adoption Cai et al. (2015).

neighbors, the fraction of treated neighbors in 2-hop neighborhoods, 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. 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. We 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. Compared to $\hat{\tau}_{\text{chin}}$, $\hat{\tau}_{\text{refex-lasso}}$ has smaller standard error and 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 as a smaller standard error. The reasons are twofold. First, bootstrap in general is conservative. Second, ReFeX estimate should have larger variance as we have a random selection procedure involved.

7 Discussion

In this paper, we have developed a method to do estimation and inference for the global average treatment effect (GATE) when network interference is present. We develop a procedure that can be used to estimate the GATE without pre-specifying 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 manuscript. First, our results only consider designs that satisfy the uniformity assumption (e.g., Bernoulli design): 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.

Acknowledgements

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

References

Aronow, P. M. & Samii, C. (2017), 'Estimating average causal effects under general interference, with application to a social network experiment', *Annals of Applied Statistics* **11**(4), 1912–1947.

Awadelkarim, A. & Ugander, J. (2020), Prioritized restreaming algorithms for balanced graph partitioning, *in* 'Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining', KDD '20, Association for Computing Machinery, New York, NY, USA, p. 1877–1887.

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

Basse, G. W. & Airoldi, E. M. (2018), 'Limitations of design-based causal inference and a/b testing under arbitrary and network interference', *Sociological Methodology* **48**(1), 136–151.

Bickel, P. J., Ritov, Y. & Tsybakov, A. B. (2009), 'Simultaneous analysis of Lasso and Dantzig selector', *The Annals of Statistics* **37**(4), 1705 – 1732.

Bousquet, O., Boucheron, S. & Lugosi, G. (2004), *Introduction to Statistical Learning Theory*, Springer Berlin Heidelberg, Berlin, Heidelberg, pp. 169–207.

Bramoullé, Y., Djebbari, H. & Fortin, B. (2009), 'Identification of peer effects through social networks', *Journal of Econometrics* **150**(1), 41–55.

Cai, J., De Janvry, A. & Sadoulet, E. (2015), 'Social networks and the decision to insure', *American Economic Journal: Applied Economics* **7**(2), 81–108.

Calinescu, G., Karloff, H. & Rabani, Y. (2005), 'Approximation algorithms for the 0-extension problem', *SIAM Journal on Computing* **34**(2), 358–372.

Cameron, A. C., Gelbach, J. B. & Miller, D. L. (2008), 'Bootstrap-Based Improvements for Inference with Clustered Errors', *The Review of Economics and Statistics* **90**(3), 414–427.

Chin, A. (2019), 'Regression adjustments for estimating the global treatment effect in experiments with interference', *Journal of Causal Inference* **7**(2), 20180026.

Cox, D. R. (1958), *Planning of experiments*, New York, Wiley.

Deng, A., Xu, Y., Kohavi, R. & Walker, T. (2013), 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, Association for Computing Machinery, New York, NY, USA, pp. 123–132.

Eckles, D., Karrer, B. & Ugander, J. (2017), 'Design and analysis of experiments in networks: Reducing bias from interference', *Journal of Causal Inference* **5**(1), 20150021.

Efron, B. (1979), 'Bootstrap Methods: Another Look at the Jackknife', *The Annals of Statistics* **7**(1), 1 – 26.

Efron, B. & Tibshirani, R. (1994), *An Introduction to the Bootstrap*, Chapman & Hall/CRC Monographs on Statistics & Applied Probability, Taylor & Francis.

Halloran, M. E. & Struchiner, C. J. (1995), 'Causal inference in infectious diseases', *Epidemiology* **6**(2), 142–151.

Hamilton, W., Ying, Z. & Leskovec, J. (2017), Inductive representation learning on large graphs, *in* I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan & R. Garnett, eds, 'Advances in Neural Information Processing Systems', Vol. 30, Curran Associates, Inc.

Hammond, D. K., Vandergheynst, P. & Gribonval, R. (2011), 'Wavelets on graphs via spectral graph theory', *Applied and Computational Harmonic Analysis* **30**(2), 129–150.

Harshaw, C., Sävje, F., Eisenstat, D., Mirrokni, V. & Pouget-Abadie, J. (2022), 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, Association for Computing Machinery, New York, NY, USA, p. 606.

Henderson, K., Gallagher, B., Li, L., Akoglu, L., Eliassi-Rad, T., Tong, H. & Faloutsos, C. (2011), 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, Association for Computing Machinery, New York, NY, USA, pp. 663–671.

Holland, P. W. (1986), 'Statistics and causal inference', *Journal of the American Statistical Association* **81**(396), 945–960.

Hong, G. & Raudenbush, S. W. (2006), 'Evaluating kindergarten retention policy', *Journal of the American Statistical Association* **101**(475), 901–910.

Imbens, G. W. & Rubin, D. B. (2015), *Causal Inference for Statistics, Social, and Biomedical Sciences: An Introduction*, Cambridge University Press.

Karrer, B., Shi, L., Bhole, M., Goldman, M., Palmer, T., Gelman, C., Konutgan, M. & Sun, F. (2021), Network experimentation at scale, *in* 'Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining', KDD '21, Association for Computing Machinery, New York, NY, USA, pp. 3106–3116.

Kipf, T. N. & Welling, M. (2017), Semi-supervised classification with graph convolutional networks, *in* 'International Conference on Learning Representations'.

Kojevnikov, D. (2021), 'The bootstrap for network dependent processes', *arXiv preprint arXiv:2101.12312* .

Lee, J. D., Sun, D. L., Sun, Y. & Taylor, J. E. (2016), 'Exact post-selection inference, with application to the lasso', *The Annals of Statistics* **44**(3), 907 – 927.

Lin, W. (2013), 'Agnostic notes on regression adjustments to experimental data: Reexamining Freedman's critique', *The Annals of Applied Statistics* **7**(1), 295 – 318.

Luo, S. & Chen, Z. (2014), 'Sequential lasso cum ebic for feature selection with ultra-high dimensional feature space', *Journal of the American Statistical Association* **109**(507), 1229–1240.

Manski, C. F. (1993), 'Identification of endogenous social effects: The reflection problem', *The Review of Economic Studies* **60**(3), 531–542.

Manski, C. F. (2013), 'Identification of treatment response with social interactions', *The Econometrics Journal* **16**(1), S1–S23.

Moffit, R. A. (2001), Policy Interventions, Low-Level Equilibria, and Social Interactions, *in* 'Social Dynamics', The MIT Press.

Nishimura, J. & Ugander, J. (2013), 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, Association for Computing Machinery, New York, NY, USA, p. 1106–1114.

Pouget-Abadie, J., Saint-Jacques, G., Saveski, M., Duan, W., Ghosh, S., Xu, Y. & Airoldi, E. M. (2019), 'Testing for arbitrary interference on experimentation platforms', *Biometrika* **106**(4), 929–940.

Raskutti, G., Wainwright, M. J. & Yu, B. (2010), 'Restricted eigenvalue properties for correlated gaussian designs', *Journal of Machine Learning Research* **11**(78), 2241–2259.

Rosenbaum, P. R. (2007), 'Interference between units in randomized experiments', *Journal of the American Statistical Association* **102**(477), 191–200.

Rubin, D. B. (1974), 'Estimating causal effects of treatments in randomized and nonrandomized studies.', *Journal of educational Psychology* **66**(5), 688.

Saint-Jacques, G., Varshney, M., Simpson, J. & Xu, Y. (2019), 'Using ego-clusters to measure network effects at linkedin'.

Shi, L. & Chen, B. (2020), 'Comparison and benchmark of graph clustering algorithms', *arXiv preprint arXiv:2005.04806* .

Sinclair, B., McConnell, M. & Green, D. P. (2012), 'Detecting spillover effects: Design and analysis of multilevel experiments', *American Journal of Political Science* **56**(4), 1055–1069.

Sobel, M. E. (2006), 'What do randomized studies of housing mobility demonstrate?', *Journal of the American Statistical Association* **101**(476), 1398–1407.

Spielman, D. A. & Teng, S.-H. (2013), 'A local clustering algorithm for massive graphs and its application to nearly linear time graph partitioning', *SIAM Journal on Computing* **42**(1), 1–26.

Tibshirani, R. (1996), 'Regression shrinkage and selection via the lasso', *Journal of the Royal Statistical Society. Series B (Methodological)* **58**(1), 267–288.

Traud, A. L., Mucha, P. J. & Porter, M. A. (2012), 'Social structure of facebook networks', *Physica A: Statistical Mechanics and its Applications* **391**(16), 4165–4180.

Ugander, J., Karrer, B., Backstrom, L. & Kleinberg, J. M. (2013), Graph cluster randomization: network exposure to multiple universes, *in* I. S. Dhillon, Y. Koren, R. Ghani, T. E. Senator, P. Bradley, R. Parekh, J. He, R. L. Grossman & R. Uthurusamy, eds, 'The 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD 2013, Chicago, IL, USA, August 11-14, 2013', ACM, pp. 329–337.

Ugander, J. & Yin, H. (2020), 'Randomized graph cluster randomization', *arXiv preprint arXiv:2009.02297* .

van de Geer, S. A. & Bühlmann, P. (2009), 'On the conditions used to prove oracle results for the Lasso', *Electronic Journal of Statistics* **3**(none), 1360 – 1392.

von Luxburg, U. & Schölkopf, B. (2011), *Statistical Learning Theory: Models, Concepts, and Results*, Vol. 10, Elsevier North Holland, Amsterdam, Netherlands, pp. 651–706.

Wainwright, M. J. (2019), *High-Dimensional Statistics: A Non-Asymptotic Viewpoint*, Cambridge Series in Statistical and Probabilistic Mathematics, Cambridge University Press.

Yu, C. L., Airoldi, E. M., Borgs, C. & Chayes, J. T. (2022), 'Estimating the total treatment effect in randomized experiments with unknown network structure', *Proceedings of the National Academy of Sciences* **119**(44), e2208975119.

A Proofs

The proofs of Proposition 4.1 and Proposition 4.2 will be exactly the same as the proofs of Proposition 1 and Proposition 2 in Luo & Chen (2014) 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 Luo & Chen (2014), i.e., though our sequential selection procedure is different from that in Luo & Chen (2014), 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 $\{u_1^{t+1}, u_2^{t+1}, \cdots, u_{i_{t+1}}^{t+1}\}$.

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.

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

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

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 above derivative to zero, we have that

$$\hat{\beta}(s_{*t}) = [X^T(s_{*t})X(s_{*t})]^{-1}X^T(s_{*t})[y - X(A_{t+1})\beta(A_{t+1})]. \tag{10}$$

Substituting (10) into the objective function, we obtain

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

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

Now we show the proof Theorem 4.3. We will make use of standard results about LASSO ℓ_2-error bounds. Recall the following result Wainwright (2019):

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 $\lambda_n \geq 2\|\frac{\mathbf{X}^T w}{n}\|_\infty$. 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.,

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

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

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

$\square$

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

$$\|\hat{\theta} - \theta^*\|_2 \leq \frac{6C\sigma}{\kappa}\sqrt{s}\left(\sqrt{\frac{2\log d}{n}} + \delta\right) \tag{11}$$

with the same high probability Wainwright (2019).

Now we are ready to prove consistency. First notice that

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

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 (11). 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 fration 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 non-zero 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,

$$\begin{aligned} \frac{1}{n_0}\|\tilde{U^0}\Delta\|_2^2 \ge & \frac{1}{n_0}\sum_{i:W_i=0}(u_i^1 - \bar{u}^1)^2\Delta_1^2 \\ & + \left(\frac{1}{n_0}\sum_{i:W_i=0}(u_i^2 - \bar{u}^2)^2 - \frac{1}{3}\frac{1}{n_0}\sum_{i:W_i=0}(u_i^1 - \bar{u}^1)(u_i^2 - \bar{u}^2)\right)\Delta_2^2. \end{aligned} \tag{12}$$

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

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

Here $\textcircled{2}$ can be done similarly:

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

For $\textcircled{3}$, we have that

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

Notice that we have already shown that

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

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

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

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

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

□

B Supplementary Materials

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

$$y_i(w) = -5 + 2z_i w_i + 0.03\nu_i + \frac{1}{1 + 0.001\exp(-0.03\nu_i + 9)} + \frac{10}{3 + \exp(-8\rho_i + 3.2)} + \epsilon_i.$$

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