1. Introduction

Graphs are widely used to model data in various fields wherein vertices represent entities or objects of interest and the edges represent pairwise relationships between the vertices. For example, in social network graphs the vertices represent individuals with the edges showing the communication between these individuals. Another example is citation network where the vertices represent articles and the (directed) edges represent citations between the articles. Finally, in many neuroscience applications, the vertices represent brain regions of interest and the edges summarize the inter-connectivity between these regions. Due to the prevalence of network data, there has been a great deal of research done recently in statistical inference on graphs, including, but not limited to, estimation of graph parameters (Lloyd et al. 2012; Xu 2017), one-sample and multi-sample hypothesis testing (Moreno and Neville 2013; Tang et al. 2017), graph clustering and classification (Kudo, Maeda, and Matsumoto 2005; Schaeffer 2007; Yin et al. 2017; Zhang et al. 2018), and vertex nomination (Fishkind et al. 2015; Yoder et al. 2020).

Vertex nomination is the graph analog of recommender systems for general tabular data. The simplest and most widely studied variant of this problem is in the single graph setting wherein, given a network and a subset of interesting vertices whose identities are partially known, the task is to identify, using the known interesting vertices, the remaining vertices of interest. The number of interesting vertices is, in general, much smaller than the total number of vertices in the graphs, and vertex nomination algorithms usually seek to output a list of candidate vertices (that are deemed interesting) with the aim that the remaining true but unknown vertices of interest are concentrated near the top of the list.

The vertex nomination problem in the single graph setting appears, at first blush, to be similar to the more widely studied community detection problem (Duch and Arenas 2005; Newman 2006; Fortunato 2010); however, there are important conceptual and practical differences between the two. More specifically, community detection is concerned with clustering or partitioning all the vertices of a network into communities or clusters; a cluster is, roughly speaking, a group of vertices exhibiting a different connectivity pattern within the cluster as compared to the connectivity between clusters. In contrast, as we alluded to earlier, vertex nomination is only concerned with identifying a small subset of vertices of interest, and furthermore, these vertices of interest might not form a cluster in the usual sense, for example, their intraconnectivity does not need to be qualitatively different from their connectivity to other “noninteresting” vertices. Due to this reason, vertex nomination
is also not the same as local graph clustering (Spielman and Teng 2013; Yin et al. 2017) as local graph clustering is concerned with clustering the vertices around the neighborhood of a given seed vertex; the unknown vertices of interest might or might not be included in that neighborhood.

Before continuing with our exposition we emphasize that, similar to recommender systems for tabular data or community detection in networks, vertex nomination is intrinsically an unsupervised learning problem. It is universally accepted that these type of problems generally do not have clearly defined solutions. For example, given a collection of data points in \( \mathbb{R}^d \), there can be numerous different ways to group these data points into clusters and furthermore all of these clusterings can be valid, and the preference for one clustering over another clustering depends on the setting and/or objective of the data analysis at hand. Analogously, given a graph \( G \), what characterizes a vertex \( v \) or a collection of vertices \( S \) in \( G \) as being interesting is in general not well-defined and could vary between applications and/or between users. Therefore, to present a relevant notion of “interestiness,” it is usually assumed that there exists a generative process underlying the observed graph(s); the process itself could be latent or partially observed. This is similar to the use of stochastic block models or its variants in the context of community detection, the use of the Bradley–Terry model in ranking problems, and the assumption of low-rank factor models in recommender systems. Furthermore, even when we assume that the latent generative model belongs to a collection of models denoted by \( \mathcal{M} \), if \( \mathcal{M} \) is not sufficiently restricted then there does not exist a vertex nomination algorithm that is well-behaved for all models in \( \mathcal{M} \), that, for any algorithm \( A \) there exists a model \( M_0 \in \mathcal{M} \) such that the nomination accuracy of \( A \) when given data generated from \( M_0 \) is no better than random guessing; see Lyzinski, Levin, and Priebe (2019) for a more precise formulation and statement of this result. In light of the above discussion, in this article we shall present our methodology in the context of a generative model for which there are latent but unobserved features associated with the vertices and, from these latent features, we can define an appropriate notion of “interestiness.” Similarly, our real data analysis examples are motivated by datasets where there are one-to-one correspondence between (a subset of) the vertices.

Typical applications of vertex nomination include predicting group membership in social networks (Coppersmith and Priebe 2012), searching and indexing in databases (Levin 2017), and identifying specific type of neurons (e.g., motor neurons) in neuroscience (Fishkind et al. 2015). A sizable number of techniques have been developed for vertex nomination in the single graph setting, including methods based on likelihood maximization, Bayesian MCMC, and spectral decomposition of the adjacency matrices, see Fishkind et al. (2015), Yoder et al. (2020), Lyzinski et al. (2016), Lee and Priebe (2012), Coppersmith and Priebe (2012), Sun, Tang, and Priebe (2012) and the references therein. Among these diverse techniques, the spectral decomposition approach is one of the most practical as it is computationally efficient and can be scaled to handle reasonably large and sparse networks.

The current article focuses on another important, albeit much less studied, variant of the vertex nomination problem, namely vertex nomination across graphs. More specifically, given two networks \( G_1 \) and \( G_2 \) and a vertex of interest \( x \) in the network \( G_1 \), our task is to find the corresponding vertex of \( x \) in \( G_2 \) (if it exists). We emphasize that \( G_1 \) and \( G_2 \) need not have the same number of vertices and that the correspondence between the vertices of \( G_1 \) and \( G_2 \) is largely unknown. The following is a concrete practical example of the above problem. Consider the pair of high school friendship networks (Patsolic et al. 2017). The first social network \( G_1 \), having 156 vertices, represents a Facebook network, in which two vertices are adjacent if the pair of individuals are friends on Facebook. The second social network \( G_2 \) with 134 vertices, is created based on the result of survey, and two vertices are adjacent if the students report they are friends. There are 82 students appearing in both social network \( G_1 \) and \( G_2 \). Given a student of interest in \( G_1 \), our goal is to identify the corresponding student in \( G_2 \). We may know the corresponding relationship of some other shared students, and the information can help us to nominate the student of interest in \( G_2 \). This is a typical example, and similar examples could be constructed including (a) identifying which user in Instagram corresponding to some specific user in Facebook, (b) identifying topics of interest across graphical knowledge bases (Sun and Priebe 2013) and (c) identifying structural signal across connectomes (Sussman et al. 2020).

Two algorithms were recently proposed for this multi-graph vertex nomination problem. The authors of Agterberg et al. (2020) proposed an algorithm based on spectral graph embedding wherein (a) the graphs are spectrally embedded into Euclidean space, (b) the embedded points are aligned via orthogonal Procrustes transformation, (c) the embedded points are simultaneously clustered via Gaussian mixture modeling (Fraley and Raftery 1998), and (d) output the candidate vertices using the resulting clustering. In contrast, the authors of Patsolic et al. (2017) proposed an algorithm based on seeded graph matching (Lyzinski, Fishkind, and Priebe 2014) wherein they graph match induced subgraphs generated around neighbourhoods of the vertices with known correspondence in each network.

The proposed algorithm in this article also uses spectral graph embedding. It is noted, empirically, that spectral graph embedding approaches are much faster and more scalable, computationally, as compared to graph matching approaches. Our approach is similar to Agterberg et al. (2020) in that we also spectrally embed the graphs into a common Euclidean space. However, in contrast to the Gaussian mixture modeling of Agterberg et al. (2020), our nomination lists are based on solving a penalized linear assignment problem, and are thus not dependent on tuning parameters such as the number of clusters and the shape/orientation of these clusters, both of which are hard to tune and could have significant impacts on the ordering in the nomination lists. Indeed, if the embedding dimension \( d \) is moderately large, a Gaussian mixture model with arbitrary covariance matrices requires estimation of \( O(Kd^2) \) parameters where \( K \) is the maximum number of Gaussian component.

We also prove consistency results about our scheme when the number of vertices goes to infinity under mild assumptions for a wide class of popular random graph models. Furthermore, for a class of random graph model where the edges of the two graphs are pairwise correlated, we analyze how the magnitude of this correlation influences the consistency.
2. Methodology

We first introduce some notations. Let the two graphs be denoted as $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$ where $V_1$ and $V_2$ are the vertices sets and $E_1$ and $E_2$ are the edges sets. We shall assume that our graphs are undirected. We also partition the vertices sets $V_1$ and $V_2$ as $V_1 = U_1 \cup J_1$ and $V_2 = U_2 \cup J_2$ where $U_1$ and $U_2$ denote the sets of shared vertices between $G_1$ and $G_2$. We express that the correspondence between the vertices in $U_1$ and $U_2$ are only partially known, that is, we further partition $U_1$ and $U_2$ as $U_1 = \{x\} \cup S_1 \cup W_1$ and $U_2 = \{x\} \cup S_2 \cup W_2$ where $x$ denote the known vertex of interest in $G_1$; $\sigma : U_1 \to U_2$ is a bijection such that for any $v \in U_1$, $\sigma(v)$ is its corresponding vertex in $U_2$; $S_1$ and $S_2$ are the seed sets with $|S_1| = |S_2| = K$, where we already know the corresponding relationship, that is, we know the bijection $\sigma|_{S_1} : S_1 \to S_2$; and $W_1$ and $W_2$ are the remaining vertices of interest. We shall assume that the mapping $\sigma$ from $W_1$ to $W_2$ is unknown, and furthermore, while the recovery of this correspondence between $W_1$ and $W_2$ is important, it is also potentially less pressing than the recovery of $\sigma(x)$. In summary, we only know the correspondence between $S_1$ and $S_2$, and given the known vertex of interest $x$, we are interested in finding its unknown correspondence $\sigma(x) \in V_2 \setminus S_2$. Our goal is thus to seek a nomination list of the vertices in $V_2 \setminus S_2$, ranked according to our confidence in how similar they are to $x$.

The following example can help to understand how the partitioning of the vertices in each graph arise. We consider the high school friendship network dataset containing two observed graphs. The first graph $G_1$ is extracted from the Facebook social network and the second graph $G_2$ is created based on a survey of the students’ friendship. These two networks share some students in common. For the first social network $G_1$, we partition the students set $V_1$ as the union of the set $U_1$ of shared students and the set $J_1$ of remaining students. Similarly, we also have the partition $V_2 = U_2 \cup J_2$ for $G_2$. There is a bijection between the shared students of the two social networks, but we only observe this bijection for a subset of the vertices; we denote these sets as $S_1 \subset U_1$ and $S_2 \subset U_2$. We have interest in the unknown corresponding relationship. For a specific student of interest $x \in U_1 \setminus S_1$, our goal is to find which vertex $\sigma(x)$ in the second graph corresponds to the same student. We then use $W_1 = U_1 \setminus \{x\} \setminus S_1$ and $W_2 = U_2 \setminus (\sigma(x)) \setminus S_2$ to denote the remaining students. More details for the high school friendship network data can be found in Section 4.2.1.

We now describe our algorithm for finding $\sigma(x)$. Our algorithm proceeds in three main steps. In the first step we spectrally embed each graph into some $d$-dimensional Euclidean space. We next aligned these embeddings either via solving an orthogonal Procrustes problem in the case when $K$, the number of seeds vertices, is at least as large as the embedding dimension $d$, or via solving a point set registration problem. Finally we solve a quadratic program, using the pairwise distances between the embedded points, to map each vertex $v \in V_1 \setminus S_1$ to some ordered subset $\ell(v) \subset V_2 \setminus S_2$; $\ell(v)$ serves as the nomination list of the vertices in $V_2 \setminus S_2$ most similar to $v$. We now describe these steps in detail.

2.1. Adjacency Spectral Embedding

We spectrally embed the graphs by truncating their eigenvalue decomposition. More specifically, given an $n \times n$ adjacency matrix $A$ of a graph and a positive integer $d$ for the embedding dimension, we compute

$$A = \sum_{i=1}^{n} \lambda_i u_i u_i^T,$$

where $|\lambda_1| \geq |\lambda_2| \geq \ldots$ are the eigenvalues and $u_1, u_2, \ldots, u_n$ are the corresponding eigenvectors. The adjacency spectral embedding of $A$ (into $\mathbb{R}^d$) is then the $n \times d$ matrix

$$\tilde{X} = [|\lambda_1|^{1/2}u_1, |\lambda_2|^{1/2}u_2, \ldots, |\lambda_d|^{1/2}u_d].$$

The rows of $\tilde{X}$ represent the (low-dimensional) embedding of the vertices of $A$ into $\mathbb{R}^d$.

In practice, we can choose $d$ by looking at the eigenvalues of the adjacency matrix. A ubiquitous and principled method is to examine the so-called scree plot and look for “elbow” or “knees” defining the cutoff between the top (signal) $d$ dimensions and the noise dimensions. Zhu and Ghodsi (2006) provides an automatic dimensionality selection procedure to look for the “elbow” by maximizing a profile likelihood function. Han, Yang, and Fan (2019) suggests another universal approach to rank inference via residual subsampling for estimating rank $d$. We can also determine $d$ by eigenvalue ratio test (Ahn and Horenstein 2013) or by empirical distribution of eigenvalues (Onatski 2010).

2.2. Orthogonal Procrustes and Point Set Registration

We applied adjacency spectral embedding to the graphs $G_1$ and $G_2$, thereby obtaining the $n \times d$ matrices $\tilde{X}_1$ and $\tilde{X}_2$, respectively. As the rows of $\tilde{X}_1$ and $\tilde{X}_2$ represent the low-dimensional embeddings of the vertices in $G_1$ and $G_2$, we should expect that, for similar vertices, these rows are close in $\ell_2$ distance. This is, however, not necessarily the case as the embeddings $\tilde{X}_1$ and $\tilde{X}_2$ are not unique, that is, $\tilde{X}_1$ and $\tilde{X}_2$ are only defined up to some orthogonal transformations as the eigendecomposition of $A_1$ is not, in general, unique. We thus need to align $\tilde{X}_1, \tilde{X}_2$ by an orthogonal transformation $W$ to eliminate this potential nonidentifiability. We describe two methods for finding $W$. The first method is applicable when $K$, the number of seed vertices, is larger than or equal to $d$, the embedding dimension; the second method, which is more general but possibly less accurate, is applicable for $K < d$, including the important case of $K = 0$.

2.2.1. Orthogonal Procrustes (when $K \geq d$)

When the number of seed vertices is greater than or equal to the embedding dimension, we find the orthogonal transformation $W$ by solving the orthogonal Procrustes problem (Schönemann 1966) to align the seeded vertices across graphs, that is,

$$\hat{W} = \arg \min_{W \in O(d)} ||(\tilde{X}_1)_{S_1} W - (\tilde{X}_2)_{S_1}||_F,$$

where $O(d)$ is the set of all $d \times d$ orthogonal matrices and $(\tilde{X}_1)_{S_1}$ is a $|S_1| \times d$ matrix whose rows are the rows of $\tilde{X}_1$ indexed by the seed set $S_1$. With $S_2 = \{\sigma(v) : v \in S_1\}$, $(\tilde{X}_2)_{S_2}$ is defined similarly, so that the $i$th row of $(\tilde{X}_1)_{S_1}$ corresponds to the $i$th
row of $(\hat{X}_2)_S$. The minimizer $\hat{W}$ has an explicit solution as $\hat{W} = UV^\top$ where $UDV^\top$ is the singular value decomposition of the $d \times d$ matrix $(\hat{X}_2)_S^\top \hat{X}_1)_S$. After finding $\hat{W}$ we set $\hat{X}_1 = \hat{X}_1 \hat{W}$.

### 2.2.2. Adaptive Rigid Point Set Registration (when $K < d$)

Even when we do not have enough seed vertices or even no seed set ($K = 0$), as long as we have a reasonable number of vertices that are shared between the two graphs, we can apply the coherent point drift algorithm in Myronenko and Song (2010) to align $\hat{X}_1$ and $\hat{X}_2$. The algorithm in Myronenko and Song (2010) finds an affine transformation to best align the centroids of the clusters of $\hat{X}_1$ to the centroids of the clusters of $\hat{X}_2$. More specifically, given a $n \times d$ matrix $\hat{X}_1$ and $m \times d$ matrix $\hat{X}_2$, we find $s \in \mathbb{R}$, $t \in \mathbb{R}^d$ and $W \in \mathbb{O}_d$ that minimize the following objective function

$$Q(\hat{W}, t, s, \sigma^2) = \frac{1}{\sigma^2} \sum_{i=1}^n \sum_{j=1}^m P(i \mid \hat{X}_2)_j \left | \left | \hat{X}_2)_j - sW^\top (\hat{X}_1)_i - t \right | \right |^2 + \frac{n \sigma^2}{2} \log \sigma^2,$$

where $(\hat{X}_1)_i$ represents the $i$th row of $\hat{X}_1$, that is, the $i$th vertex's embedding of $G_1$; $(\hat{X}_2)_j$ is defined similarly. Here $N_p = \sum_{i=1}^n \sum_{j=1}^m P(i \mid \hat{X}_2)_j$ is a normalizing constant, with $P(i \mid \hat{X}_2)_j$ the correspondence probability between two vertices' embeddings $(\hat{X}_1)_i$ and $(\hat{X}_2)_j$, defined as the posterior probability of the centroid given the vertex's embedding $(\hat{X}_2)_j$, that is,

$$P(i \mid \hat{X}_2)_j = \frac{\exp \left ( \frac{1}{\sigma^2} \left | \left | \hat{X}_2)_j - sW^\top (\hat{X}_1)_i - t \right | \right |^2 \right )}{c + \sum_{k=1}^m \exp \left ( \frac{1}{\sigma^2} \left | \left | \hat{X}_2)_j - sW^\top (\hat{X}_1)_k - t \right | \right |^2 \right )},$$

for some constant $c$ (where $c$ is a function of the model parameters defined above; see Myronenko and Song 2010). The minimization of $Q$ is done via an EM-algorithm. For more details, please refer to Myronenko and Song (2010).

The resulting minimizer $(\hat{s}, \hat{t}, \hat{W})$ yield an affine transformation $T$ of $\hat{X}_1$ via $T((\hat{X}_1)_i) = sW^\top (\hat{X}_1)_i + t$. We note, however, that in the context of our current work, the alignment of $\hat{X}_1$ and $\hat{X}_2$ does not require the scaling $s$ and the translation $t$. We thus make a few minor adjustments to the EM algorithm in Myronenko and Song (2010). In particular, (a) we always set $s = 1$ and $t = 0$; (b) we iteratively update $W$ via $W = UV^\top$ instead of $W = UCV^\top$; (c) we initialize $W$ as a diagonal matrix with 1 or $\lambda$ diagonal elements so that the initial error in the EM approach $\lambda^2 = \frac{1}{\lambda} \sum_{i=1}^n \sum_{j=1}^m \left | \left | (\hat{X}_1)_i W_j - (\hat{X}_1)_j \right | \right |^2$ is as small as possible. Once we get the final orthogonal matrix $\hat{W} \in \mathbb{R}^{d \times d}$, we set $\hat{X}_1 = \hat{X}_1 \hat{W}$.

### 2.3. Quadratic Program

We now formulate a quadratic program to find, for each vertex $v \in V_1$, a collection of vertices $\ell(v) \subset V_2 \setminus S_2$ that are "most similar" to $v$. Here similarity between $v \in V_1$ and $u \in V_2$ is measured in terms of the Euclidean distances between their embeddings. In other words, given a query vertex $v$ in the first graph, our proposed algorithm outputs a nomination list $\ell(v)$ of vertices in the second graph that are most similar to $v$: these vertices are deemed "interesting" in the context of the query vertex $v$.

Our quadratic program is described as follows. Given the aligned embeddings $\hat{X}_1$ and $\hat{X}_2$, we find $\hat{D}$ to minimize the following objective function

$$\hat{D} = \arg \min _{\hat{D} \in \mathbb{R}^{n \times m}} \sum _{i=1}^n \sum _{j=1}^m ||(\hat{X}_1)_i - (\hat{X}_2)_j||^2 \cdot D_{ij} + \lambda ||D||_F^2,$$  \tag{1}

subject to the constraints that

(i) $\sum_{j=1}^m D_{ij} = m$, for all $1 \leq i \leq n$,
(ii) $\sum_{i=1}^n D_{ij} = n$, for all $1 \leq j \leq m$,
(iii) $D_{ij} \geq 0$, for all $1 \leq i \leq n, 1 \leq j \leq m$,
(iv) $D(S_1, S_2) = \min[m, n]$, for all $1 \leq k \leq K$.

Here $||[(\hat{X}_1)_i - (\hat{X}_2)_j||$ is the Euclidean distance between the $i$th vertex's embedding of $G_1$ and the $j$th vertex's embedding of $G_2$, $\lambda > 0$ is a penalty parameter, and $(S_1)_k, (S_2)_k$ represents the index of the $k$th seed vertex in $G_1$ and $G_2$, respectively.

The motivation behind solving the above optimization problem is as follows. The constraints on $\hat{D}$ state that (a) each vertex $i \in V_1$ is mapped to some collection of vertices in $j \in V_2$, namely those for which $D_{ij} > 0$ (constraint iii.); (b) since for any $i \in V_1$, $\sum_{j} D_{ij} = m$ where $m = |V_2|$, larger values of $D_{ij}$ indicates more "similarity" between $i \in V_1$ and $j \in V_2$ (constraints i. and ii.); and (c) a seed vertex $s \in S_1$ will get mapped to its unique correspondence $\sigma(s) \in S_2$ (constraint iv.).

We now consider the objective function. The first part of the objective function indicates that the similarity between the $i$th vertex in $V_1$ and the $j$th vertex in $V_2$ is based on the Euclidean distance $||[(\hat{X}_1)_i - (\hat{X}_2)_j||$, that is larger distance should lead to smaller $\hat{D}_{ij}$. We can then consider, for each $i \in V_1$, the nomination list for $i$ as being the vertices in $V_2 \setminus S_2$ arranged according to decreasing values of $D_{ij}$ for $V_2 \setminus S_2$. The second part of the objective function, that is, the penalty term $\lambda ||D||_F^2$, is to discourage sparsity of $\hat{D}$. More specifically, removal of the penalty term $\lambda ||D||_F^2$ leads to a linear programming problem for which the minimizer $\hat{D}$ may lie on the boundary of the feasibility region, that is, the elements of $\hat{D}$ take values only in $\{0, m, n\}$. If $D_{ij} = 0$, then the $i$th vertex in $V_1$ is mapped to the $j$th vertex in $V_2$ and if $D_{ij} = m$ then the $j$th vertex in $V_2$ is mapped to the $i$th vertex in $V_1$. This gives a nomination list with a single candidate. This type of nomination list, when accurate, can significantly reduces the burden of post-processing and checking/verifying multiple candidates. However, it is also likely to be nonrobust. By adding the penalty term, we encourage the elements in each row of $\hat{D}$ to be more uniform since, for any vector $x \in \mathbb{R}^m$, $||x||_F \geq m^{-1/2} ||x||_1$ with equality if and only if all the elements of $x$ are the same. We note that the optimization problem in Equation (1) is analogous to the quadratically regularized optimal transport problem between the point masses induced by $\hat{X}_1$ and $\hat{X}_2$.

The resulting optimization problem is a quadratic program with linear constraints, and the coefficient matrix of the quadratic term is positive definite. Thus, for a fixed $\lambda > 0$, the optimization function is strongly convex and hence there exists a unique global minimizer $\hat{D}$ for any given $\hat{X}_1$ and $\hat{X}_2$. In particular, for a fixed $\lambda > 0$, the solution $\hat{D}_2$ of Equation (1) is
of the form (Blondel, Seguy, and Rolet 2018)
\[
D_\lambda(i,j) = \frac{1}{\lambda} \left[ ||\tilde{X}_1 - \tilde{X}_2||^2 - \alpha_i - \beta_j \right],
\]
where \([z]_+ = \max\{z, 0\}\), and \(\tilde{\alpha} = (\tilde{\alpha}_1, \ldots, \tilde{\alpha}_n)\) and \(\tilde{\beta} = (\tilde{\beta}_1, \ldots, \tilde{\beta}_m)\) solve the unconstrained dual problem
\[
\max_{a \in \mathbb{R}^n, b \in \mathbb{R}^m} r_a^T + m b^T - \frac{1}{2\lambda} \sum_{i=1}^n \sum_{j=1}^m \left[ ||\tilde{X}_1 - \tilde{X}_2|| - \alpha_i - \beta_j \right]_+.
\]

The optimal solution can be found, theoretically, in polynomial time using the ellipsoid algorithm of Kozlov, Tarasov, and Khachiyan (1980). In practice we use the Gurobi solver (Incorporated 2015), which is based on an interior-point algorithm.

There is no simple and universal approach for choosing \(\lambda\) in practice. If there are seed vertices, then we can do cross-validation by leaving out a subset (or all) of the seed vertices and choose \(\lambda\) for which the nomination on the seed vertices has smallest MRR or MNR. In general, as vertex nomination is an unsupervised learning problem, the issue of choosing tuning parameter is, in a sense, unsolved (at least without additional information such as having seed vertices or a known collection of pair of vertices that should not be matched together). Nevertheless, we note that, empirically, the nomination list found by our algorithm is not overly sensitive to the choice of the \(\lambda\), as \(\lambda\) mainly influences the magnitudes of the elements in \(D\) but does not change their relative ordering too much.

The quadratic program considered in Equation (1) is motivated by the two-dimensional linear assignment problem. The most obvious formulation of the current vertex nomination problem to three or more graphs will lead to an optimization problem that is similar to the multi-dimensional assignment problem (Pierskalla 1968), which is NP-hard. We thus leave the detailed study of multi-sample vertex nomination for future work.

### 2.4. Computational Complexity

We now describe the computational complexity for our proposed algorithm. First, the embedding step is roughly \(O(n^2 + m^2)d)\) where \(d\) is the embedding dimension (Huffel 1990) and \(n\) and \(m\) are the number of vertices in the first and second graph, respectively. The orthogonal Procrustes is roughly \(O(\min\{m, n\}d^2)\). The adaptive point set registration can be done iteratively with each iteration having computational complexity of \(O(2^d \text{nnz}d)\), and generally speaking \(\ell \leq 50\) iterations suffice from empirical observations. The solution of the quadratic programming problem can also be solved iteratively with each iteration having a complexity of \(O(nm)\) operations; the number of iterations is (empirically) generally bounded and is also independent of the data dimension. As such, the expected empirical complexity of the quadratic programming step is also \(O(nm)\) (Dessein, Papadakis, and Rouas 2018).

Empirically, we record the running time of the simulation experiments in Section 4.1 on \(n = 300\) vertices and \(n = 1000\) vertices, respectively (here we set \(m = n\)). For our algorithm with orthogonal Procrustes, the total running time for the experiments on \(n = 300\) vertices for various choices of \(\rho\) (Figure 1 or Figure 2) is about 2 hr while the total running time for \(n = 1000\) vertices (Figure C10 or Figure C11 in Appendix C, supplementary materials) is about 23 hr. The running time of our algorithm with the adaptive point set registration procedure also has the similar ratio. In summary, we see that the running time of our algorithm does scale (approximately) quadratically with \(n\), the number of vertices, which is consistent with the above analysis.

### 3. Theoretical Results

We now investigate the theoretical properties of our proposed algorithm. For simplicity we will only consider the case where \(G_1\) and \(G_2\) have the same number of vertices; the analysis presented here will also extend to the case when \(G_1\) and \(G_2\) have different number of vertices provided that the number of common vertices is sufficiently large. We first formulate a generative model for generating pairs of random graphs \((G_1, G_2)\) with underlying latent correspondence \(\sigma\) between the vertices \(V_1\) of \(G_1\) and \(V_2\) of \(G_2\). We then show consistency results about our scheme under the generative model and analyze the accuracy of the nomination list obtained by our algorithm. We next investigate the impact of having seed vertices. In particular, we propose a procedure for re-ranking the nomination lists in the presence of seed vertices. The empirical results in Section 4 indicates that if the number of seed vertices is not too small, then this optional re-ranking step outputs an improved nomination list \(\ell(x)\) compared to the nomination list obtained directly from the quadratic program solution.

Our generative model for pairs of random graphs depends on the following notion of the generalized random dot product graphs (Rubin-Delanchy et al. 2017; Young and Scheinerman 2007).

**Definition 1 (Generalized random dot product graphs).** Let \(d \geq 1\) be given and let \(X\) be a subset of \(\mathbb{R}^d\) such that \(x^T I_{p,q} y \in [0, 1]\). Here \(I_{p,q}\) is a \(d \times d\) diagonal matrix with diagonal entries containing \(p + 1\) and \(q - 1\) for integers \(p, q \geq 0, p + q = d\). For a given \(n \geq 1\), let \(X\) be a \(n \times d\) matrix with rows \(X_i \in X\) for \(i = 1, 2, \ldots, n\). A random graph \(G\) is said to be an instance of a generalized random dot product graph with latent positions \(X\) if the adjacency matrix \(A\) of \(G\) is a symmetric matrix whose upper triangular entries \(\{A(i,j)\}_{i \leq j}\) are independent Bernoulli random variables with
\[
A(i,j) \sim \text{Bernoulli}(X_i^T I_{p,q} X_j).
\]

We use GRDPG\((P)\) to represent such graph, where \(P = XI_{p,q}X^T\).

Generalized random dot product graphs are a special case of latent position graphs or graphons (Hoff, Raftery, and Handcock 2002; Diaconis and Janson 2008; Lovász 2012). In the general latent position graph model, each vertex \(v_i\) is associated with a latent or unobserved vector \(X_i\) and, given the collection of latent vectors \(\{X_i\}\), the edges are conditionally independent Bernoulli random variables with \(P[v_i \sim v_j] = \kappa(X_i, X_j)\) for some symmetric link function \(\kappa\). Generalized random dot product graphs can be used to model any latent position graphs where the link function \(\kappa\) is finite-dimensional, that is, \(\kappa\) is such that for any \(n\)
and for any collection of latent vectors \( \{X_i\}_{i=1}^{n} \), the \( n \times n \) matrix \( P \) with \( P(i,j) = \kappa(X_i, X_j) \) has rank at most \( d \) for some arbitrary but fixed \( d \) not depending on \( n \). Indeed, as \( P \) is a symmetric matrix with rank at most \( d \), \( P \) has an eigen decomposition as \( P = U A U^\top \). Hence, taking \( X = U |A|^{1/2} \) and letting \( p \) and \( q \) be the number of positive and negative eigenvalues of \( P \), we obtained a representation of \( P \) as a GRDPG.

Generalized random dot product graphs include, as special cases, the popular class of stochastic block model graphs and their degree-corrected and mixed-membership variants (Holland, Laskey, and Leinhardt 1983; Airoldi et al. 2008; Karrer and Newman 2011).

**Definition 2.** (Stochastic block model random graphs). We say a random graph \( G \) with adjacency matrix \( A \) is distributed as a stochastic block model random graph with parameters \( L, \theta, B \) if

1. The vertex set \( V \) of \( G \) is partitioned into \( L \) blocks, \( V = V_1 \cup V_2 \cup \ldots \cup V_L \).
2. The function \( \theta \) is a mapping from \( V \) to \( \{1, \ldots, L\} \) with \( \theta(i) \) denoting the block label of vertex \( i \) in \( V \).
3. The matrix \( B \in R_{+}^{L \times L} \) is a symmetric matrix of block probabilities. More specifically, given \( B \), the entries \( A(i,j) \) for \( i \leq j \) are conditionally independent Bernoulli random variables with \( A(i,j) \sim \text{Bernoulli}(B_{\theta(i),\theta(j)}) \).

We denote a stochastic block model graph as \( G \sim \text{SBM}(L, \theta, B) \).

A \( L \)-blocks stochastic block model graph (Holland, Laskey, and Leinhardt 1983) corresponds to a GRDPG where \( \mathcal{X} \) is a mixture of \( L \) point masses. Similarly, a \( L \)-blocks degree-corrected stochastic block model and a \( L \)-blocks mixed-membership stochastic block model correspond to a GRDPG where \( \mathcal{X} \) is supported on a mixture of \( L \) rays and on a convex hull of \( L \) points, respectively. See Rubin-Delanchy et al. (2017) for a more detailed description of these relationships. The perspective of representing a mixed membership SBM as a GRDPG also gives us a general approach to constructing the domain \( \mathcal{X} \) for a GRDPG in Definition 1. More specifically, we can construct \( \mathcal{X} \) by first finding some collection of \( K \) vectors \( S = \{X_1, X_2, \ldots\} \) for which \( X_i^\top I_p X_i \in \{0, 1\} \) for all \( X_i, X_j \in \mathcal{S} \) and then construct \( \mathcal{X} \) as the convex hull of the \( S \).

Our generative model for pairs of random graphs extends the GRDPG model for the single graph setting to the setting of two graphs that share a common set of vertices with edges that are possibly correlated.

**Definition 3 (\( \rho \)-correlated GRDPG).** Assume the notation in Definition 1. Let \( \rho \in [-1, 1] \) be given. A pair of random graphs \( (G_1, G_2) \) is said to be an instance of a \( \rho \)-correlated generalized random dot product graphs with latent positions \( X \) if the pair of adjacency matrices \( A_1 \) and \( A_2 \) satisfy the following conditions:

1. Marginally \( G_1 \sim \text{GRDPG}(P) \) and \( G_2 \sim \text{GRDPG}(P) \).
2. Given \( P \), the bivariate random variables \( (A_1(i,j), A_2(i,j)) \) \( \forall 1 \leq i, j \leq n \) are collectively independent and \( \text{corr}(A_1(i,j), A_2(i,j)) = \rho \), for any \( 1 \leq i < j \leq n \).

The correlation \( \rho \) in Definition 3 induces a notion of correspondence between the vertices in \( G_1 \) and \( G_2 \). More specifically, if \( \rho = 0 \) then for any two arbitrary pairs of vertices \( (i, j) \) and \( (k, \ell) \), the edges \( A_1(i,j) \) and \( A_2(k, \ell) \) are independent. In contrast, if \( \rho \neq 0 \) then \( A_1(i,j) \) and \( A_2(k, \ell) \) are independent if and only if \( \{i, j\} \neq \{k, \ell\} \). Now suppose that \( \rho 
eq 0 \). Then given a vertex of interest \( v \) in \( G_1 \), we can define the true correspondence of \( v \) in \( G_2 \) as the unique vertex \( \sigma(v) \in G_2 \) such that the edges \( A_1(v, u) \) and \( A_2(\sigma(v), \sigma(u)) \) are correlated for all \( u \in G_1 \). In summary, if \( (G_1, G_2) \) is a pair of \( \rho \)-correlated GRDPG graphs with \( \rho \neq 0 \) then there exists a canonical correspondence between the vertices of \( G_1 \) and the vertices of \( G_2 \). We use this correspondence to define our notion of “interestingness” when evaluating the proposed methodology on graphs generated from the \( \rho \)-GRDPG model, that is, given a query vertex \( v \in G_1 \), we wish to find \( \sigma(v) \in G_2 \). See Lyzinski, Fishkind, and Priebe (2014), Patsolic et al. (2017), and Agterberg et al. (2020) for further discussion of the relationship between \( \rho \) and its induced correspondence in graph matching and vertex nomination problems.

We now state our first theoretical result. The following result provides an error bound for the \( 2 \rightarrow \infty \) norm difference between the adjacency spectral embeddings of \( G_1 \) and \( G_2 \); given a \( n \times p \) matrix \( M \) with rows \( M_i \), the \( 2 \rightarrow \infty \) norm of \( M \) is the maximum \( \ell_2 \) norm of the rows \( M_i \), that is,

\[
||M||_{2 \rightarrow \infty} = \max_{i=1, \ldots, n} ||M_i|| = \max_{i=1, \ldots, n} ||M_i||. 
\]

The main feature of the following \( 2 \rightarrow \infty \) bound is that it is monotone decreasing in both \( \rho > 0 \) and \( n \), that is, larger correlation and/or larger number of vertices in each graph lead to smaller error bound that holds uniformly for all vertices of \( G_1 \) and \( G_2 \). We emphasize that previous bounds for \( \min_{w \in \Omega_q} ||\hat{X}_1 W - X||_2 \rightarrow \infty \) and \( \min_{w \in \Omega_q} ||\hat{X}_2 W - X||_2 \rightarrow \infty \) (see, e.g., Rubin-Delanchy et al. 2017) do not depend on the correlation \( \rho \), and thus will lead to a potentially sub-optimal bound of the form

\[
\min_{w \in \Omega_q} ||\hat{X}_1 W - \hat{X}_2||_2 \rightarrow \infty = O_p(n^{-1/2}).
\]

Below, we will characterize the error between asymptotic latent position estimations under the assumption that \( P = \gamma \cdot X_l p q X_l^\top \) where \( \gamma \) is a sparsity factor.

**Theorem 1.** Let \( (G_1, G_2) \sim \rho \)-GRDPG(P), where \( P = \gamma \cdot X_l p q X_l^\top \in [0, 1]^{n \times n} \) is symmetric with rank \( P = p + q = d \). Suppose that (a) \( \max_i \sum_j (1 - P_{ij}) \geq C \log^4 n \) for some universal constant \( C \); (b) the latent positions satisfy

\[
\frac{1}{n} \sum_{i=1}^{n} X_i X_i^\top I_{p q} \xrightarrow{a.s.} \Gamma \text{ as } n \to \infty. \text{ Here } \Gamma \text{ is a fixed } d \times d \text{ matrix not depending on } n. \text{ Denote by } A_1, A_2 \in \mathbb{R}^{n \times n} \text{ the adjacency matrices for } G_1 \text{ and } G_2, \text{ respectively. Let } \hat{X}_1 \text{ and } \hat{X}_2 \text{ be the adjacency spectral embedding of } A_1 \text{ and } A_2 \text{ into } \mathbb{R}^d, \text{ respectively. Then there exists a constant } c > 0 \text{ such that}
\]

\[
\min_{w \in \Omega_q} ||\hat{X}_1 W - \hat{X}_2||_2 \rightarrow \infty = (1 - \rho)^{1/2} \cdot O_p(n^{-1/2}) + O_p((\log n)^2 n^{-1/2 - c}).
\]

Condition (a) in the statement of Theorem 1 is a condition on the sparsity of the graph. In particular, Condition (a) is satisfied provided that the maximum degree of \( P \) is of order \( \Omega(\log^4 n) \). Condition (b) is a condition on the homogeneity of
In summary, estimates of the optimization problem in Equation (1) reduces to a linear programming. Suppose also that $c_{ij} > 0$ whenever $i \neq j$ and $c_{ii} = 0$, that is, the latent positions $\{X_i\}$ are unique. Then as $n \to \infty$, by the above bounds for $|\hat{c}_{ij} - c_{ij}|$, we have $\hat{D}_{ij} = n$ for all $1 \leq i \leq n$ and hence, for any vertex of interest in $G_1$, our algorithm will give the nomination list with the true $\sigma(x)$ at the top of the list.

Next consider the case where $\lambda > 0$. Define $C$ and $\hat{C}$ as the $n \times n$ matrices whose elements are $c_{ij}$ and $\hat{c}_{ij}$, respectively. Let $D = \{D : D \in \mathbb{R}^{n \times n}, D_{1n} = (n, \ldots, n)^T, D_{T1} = (n, \ldots, n)^T\}$. The optimization problem in Equation (1) is then equivalent to

$$\arg\min_{D \in D} \|C - \hat{C}\|_F^2 = \arg\min_{D \in D} \|D + \frac{1}{2\lambda} \hat{C}\|_F^2, \quad \lambda > 0.$$ 

Let $\hat{D}_\lambda$ be the unique solution of the above problem. Then $\hat{D}_\lambda$ is the projection of $-\frac{1}{2\lambda} \hat{C}$ onto the convex set $D$. Now consider the solution of Equation (1) where we replaced $C$ by $\hat{C}$ and define that unique solution as $D_\lambda$. From Equation (2) we have $\|\hat{C} - C\|_F^2 = \sum_{ij}(\hat{c}_{ij} - c_{ij})^2 = \mathcal{O}_p(n)$ and hence $\|\hat{C} - C\|_F = \mathcal{O}(n^{1/2})$.

Next recall that the projection in Frobenius norm onto convex sets is $1$-Lipschitz. We therefore have

$$\|D_\lambda - \hat{D}_\lambda\|_F \leq \frac{1}{2\lambda} \|\hat{C} - C\|_F = \frac{1}{2\lambda} \mathcal{O}(n^{1/2}).$$ 

Since $\|D\|_F \geq n$ for all $D \in D$, we see that $\|D_\lambda - D_\lambda\|_F = \mathcal{O}(\|D\|_F)$ for all $\lambda \gg n^{-1/2}$. In other words, provided that $\lambda$ is not too small, the solutions of Equation (1) using the true cost matrix $C$ and using the estimated cost matrix $\hat{C}$ are close, that is, the relative error between $D_\lambda$ and $\hat{D}_\lambda$ could be made arbitrarily small for sufficiently large $n$.

Finally, we consider how the solution $D_\lambda$ using the true cost matrix $C$ will look like as $\lambda \to 0$. Let $P_0$ be the linear programming problem $\min_{D \in D} \|C - D\|_F$. Suppose $n$ is now fixed. Then if $\lambda \to 0$, $D_\lambda$ will converge to the optimal solution with minimum Frobenius norm among the set of all optimal solutions of $P_0$, that is, letting $\xi_\lambda$ be the minimum objective value of $P_0$, we have

$$D_\lambda \to \arg\min_{D \in D} \|D\|_F : (C, D) = \xi_\lambda.$$ 

We note that there could be multiple solutions of $\{D \in D : (C, D) = \xi_\lambda\}$. Nevertheless, in the event that $P_0$ has a unique minimizer, then since $c_{ij} = 0$ for all $i$, this unique minimizer will be given by $D_\lambda = \text{diag}(n, n, \ldots, n)$. Therefore, by the continuity of the optimization problem, there exists a $\lambda > 0$ such that $D_\lambda = D_\lambda$. The previous bound for $\|D_\lambda - D_\lambda\|_F$ suggests that $D_\lambda \to D_\lambda$ as $\lambda \to 0$. A precise statement of this result, however, requires a more detailed analysis of the relationship between $\lambda$ and $n$. Indeed, the relative error bound for $\|D_\lambda - D_\lambda\|_F$ currently requires $n$ sufficiently large and $\lambda \gg n^{-1/2}$ while the convergence of $D_\lambda$ to $D_\lambda$ currently requires $\lambda \to 0$ with $n$ fixed. We leave this analysis for future work. Finally, if we can assume that for sufficiently large $n$ we also have $\min_{i \neq j} c_{ij} = \omega(n^{-1/2})$, then with high probability $D_\lambda = \text{diag}(n, n, \ldots, n)$ and we exactly recover the true correspondence for all vertices. The condition $\min_{i \neq j} c_{ij} = \omega(n^{-1/2})$ is likely to be too restrictive; we can relax this condition by assuming that the indices $\{1, 2, \ldots, n\}$ can be partitioned into $K$ distinct groups such that $c_{ij} > c_{ij} + \omega(n^{-1/2})$ for all triplets $(i, j, f)$ where $i$ and $j$ are in the same group and $i$ and $f$ are in different groups. Then under this milder condition, we can guarantee that with high probability, $D_\lambda$ satisfies $D_\lambda(i, j) = 0$ whenever $i$ and $j$ are in different groups. As a special case if $G_1, G_2$ are $\rho-$correlated stochastic block model graphs then $D_\lambda$ will be block-diagonal and hence, for sufficiently large $n$, $D_\lambda$ will also be block-diagonal with high probability. Thus, for $\rho-$correlated stochastic block models, our algorithm will generally assign each vertex $u \in G_1$ to another vertex $v \in G_2$ from the same block as $u$.

We summarize the above discussion in the following result.

**Proposition 1.** Let $P_\lambda$ and $P_\lambda$ be the optimization problem in Equation (1) with cost matrices $C = (\|X_i - X_j\|)$ and $\hat{C} = (\|\hat{W}(X_i) - (X_j)\|)$, respectively. Then for sufficiently large $n$,

$$\frac{\|D_\lambda - \hat{D}_\lambda\|_F}{\|D_\lambda\|_F} = \frac{1}{\lambda} \mathcal{O}(n^{-1/2}).$$
For a fixed \( n \), as \( \lambda \to 0 \), we have
\[
\mathbf{D}_\lambda \rightarrow \arg\min_{\mathbf{D} \in \mathcal{D}} \{ ||\mathbf{D}||_{\mathcal{F}} : \langle \mathbf{C}, \mathbf{D} \rangle = \xi_\star \},
\]
where \( \xi_\star \) is the minimum value achieved in \( P_0 \). Furthermore, suppose that \( \lambda = 0 \) and, for sufficiently large \( n \) the vertices of \( G_1 \) and \( G_2 \) can be partitioned into \( K \) distinct groups such that \( c_i < c_j \) with \( c_j - c_i = \omega(n^{-1/2}) \) for all triplets \((i, j, j')\) with \( i \) and \( j \) being the same group and \( i \) and \( j' \) being in different groups. Then with high probability \( \mathbf{D}_0 \) is a block diagonal matrix, that is, \( \mathbf{D}_0(i, j) = 0 \) whenever \( i \) and \( j \) are in different groups.

Note that the simulation results in Section 4 are more accurate than what Proposition 1 suggests, that is, the correlation structure between the two graphs lead to better nomination than just nominating vertices from the same block.

### 3.1. Reranking Based on Likelihood

The algorithm in Section 2 output a nomination list \( \ell(x) \) for the vertex of interest. When the pair \((G_1, G_2)\) is an instance of a \( \rho \)-correlated generalized random dot product graph, then for any fixed \( \rho > 0 \), Theorem 1 guarantees that \( \sigma(x) \) is located at the top of the nomination list \( \ell(x) \), that is, \( \mathbf{r}(\sigma(x))/n \to 0 \) as \( n \to \infty \), and furthermore, if \( \rho = 1 \) then \( \mathbf{r}(\sigma(x)) = 1 \) asymptotically almost surely.

We now describe a procedure for refining the nomination list so that \( \mathbf{r}(\sigma(x)) = 1 \) even when \( \rho < 1 \), provided that we have enough seed vertices. Let \( x \in V_1 \) be given and let \( v \in V_2 \) be arbitrary. Then for any seed vertex \( w \in S_1 \) with correspondence \( \sigma(w) \in S_2 \), have, by the assumptions on the \( \rho \)-correlation structure
\[
\begin{align*}
\mathbb{P}(A_1(x, w) = 1, A_2(v, \sigma(w)) = 1) &= \mathbb{P}(x, w)^2 + \rho \mathbb{P}(x, w)(1 - \mathbb{P}(x, w)), \\
&\quad \text{if } v = \sigma(x), \\
\mathbb{P}(A_1(x, w) = 1, A_2(v, \sigma(w)) = 1) &= \mathbb{P}(x, w)\mathbb{P}(v, \sigma(w)), \\
&\quad \text{if } v \neq \sigma(x).
\end{align*}
\]
Let \( C_{vw} = A_1(x, w)A_2(v, \sigma(w)) \in \{0, 1\} \), and let \( p_{vw}(\rho) \) be
\[
p_{vw}(\rho) = \mathbb{P}(x, w)\mathbb{P}(v, \sigma(w)) + \rho \mathbb{P}(x, w)(1 - \mathbb{P}(x, w)).
\]
Then the collection \( \{C_{vw}\}_{w \in S_1} \) are independent Bernoulli random variables with mean parameters \( \{p_{vw}(\rho)\}_{w \in S_1} \) where \( \rho \neq 0 \) if and only if \( v = \sigma(x) \). For a fixed \( \rho \neq 0 \), the likelihood of observing \( \{C_{vw}\}_{w \in S_1} \), assuming the edge probabilities \( \{p_{ij}\}_{i \neq j} \) are known, is then
\[
L(\rho; \{C_{vw}\}_{w \in S_1}) = \prod_{w \in S_1} \left( p_{vw}(\rho) \right)^{c_{vw}}(1 - p_{vw}(\rho))^{1 - c_{vw}}.
\]
Deciding between \( v = \sigma(x) \) and \( v \neq \sigma(x) \) is thus analogous to test \( H_0 : \rho = 0 \) against \( H_A : \rho \neq 0 \). For our problem, the edge probabilities \( \{p_{ij}\}_{i < j} \) are unknown. Nevertheless, Theorem 1 guarantees that \( \{P_{ij}\}_{i < j} \) can be estimated uniformly well by \( \{\hat{P}_{ij}\}_{i < j} \). In summary, our procedure for refining the nomination list \( \ell(x) \) is as follows.

- For every \( v \) in top ranked part of \( \ell(x) \), find \( \hat{\rho}_v \in [-1, 1] \) that maximizes the likelihood \( L(\rho; \{C_{vw}\}_{w \in S_1}) \); here the true edge probabilities \( \{P_{ij}\} \) defining \( p_{vw}(\rho) \) are replaced by their estimates \( \{\hat{P}_{ij}\}_{i < j} \).
- Reorder these \( v \in \ell(x) \) according to decreasing values of \( |\hat{\rho}_v| \).

### 4. Simulation and Real Data Experiments

We illustrate the performance of our algorithm by synthetic and real data experiments. For the synthetic data experiments, we generate two types of synthetic data which correspond to two special cases of \( \rho \)-correlated GRDPG model. For the real data experiments, we use the high-school friendship data from Moreno and Neville (2013) and Microsoft Bing entity graph transitions data from Agterberg et al. (2020). We evaluate the performance of our algorithm using the following two criteria.

- **Mean reciprocal rank (MRR)**
  The reciprocal rank (RR) of a nomination list \( \ell(x) \) is a measure of how far down a ranked list one must go to find the true corresponding vertex of interest \( \sigma(x) \), that is, with a slight abuse of notation,
  \[
  \mathrm{RR}(x) = \mathbf{r}(\sigma(x))^{-1} \in (0, 1],
  \]
  where \( \mathbf{r}(\sigma(x)) \) is the rank of \( \sigma(x) \) in \( \ell(x) \), with ties broken randomly. For Monte Carlo experiments, we also consider the mean reciprocal rank, that is, the reciprocal rank averaged over the Monte Carlo replicates; we denote this as MRR. Larger values of RR or MRR indicate better performance.

- **Mean normalized rank (MNR)**
  The normalized rank (NR) of a nomination list \( \ell(x) \) is another measure of the rank of \( \sigma(x) \) in \( \ell(x) \), and is defined as
  \[
  \mathrm{NR}(x) = \frac{\mathbf{r}(\sigma(x)) - 1}{|V_2 \setminus S_2| - 1} \in [0, 1],
  \]
  where \( |V_2 \setminus S_2| \) is the set of all possible non-seed candidates. Note that \( \mathrm{NR}(x) = 0 \) if and only if \( \mathbf{r}(\sigma(x)) = 1 \). For Monte Carlo experiments we also consider the mean normalized rank (MNR). Smaller values of NR or MNR indicate better performance.

#### 4.1. Simulation Experiments

We consider two special cases of the \( \rho \)-correlated GRDPG model. We first assumes that the matrix of edge probabilities is positive semidefinite, that is, that \( q = 0 \) in the GRDPG model. We refer to this as the \( \rho \)-RDGP or \( \rho \)-correlated random dot product graphs model. The second assumes that the edge probabilities matrix is that of the popular stochastic block model graphs Holland, Laskey, and Leinhardt (1983).

In the case of \( \rho \)-RDGP, we generate pairs of graphs \((G_1, G_2)\) on \( n = 300 \) vertices where the latent positions \( \{X_i\} \) are sampled uniformly on the unit sphere in \( \mathbb{R}^3 \). We then choose, uniformly at random, a vertex \( x \in V_1 \) and use our algorithm to find a nomination list \( \ell(x) \subset V_2 \). Recall that the edges of the graphs are pairwise correlated. This correlation structure then yields a canonical notion of correspondence between the vertices in \( G_1 \) and those in \( G_2 \). In other words, given any vertex \( v \in G_1 \) with latent position \( X_v \), the most “interesting” or “similar” vertex to \( v \) in \( G_2 \) is simply the vertex \( u \in G_2 \) with latent position \( X_u = X_v \). We evaluate our algorithm using the mean reciprocal rank (MRR) and the mean normalized rank (MNR) calculated from 500 Monte Carlo replicates. The results, as a function of the correlation \( \rho \in \{0, 0.3, 0.5, 0.7, 1\} \), are presented.
in Figure 1. The embeddings of the graphs are aligned either via orthogonal Procrustes (see Section 2.2) or via the adaptive point set registration procedure of Myronenko and Song (2010).

The setup for $\rho$-SBM($L, b, B$) is similar. We generate pairs of graphs on $n = 300$ vertices with $L = 3$ blocks and 100 vertices in each block. The block probabilities matrix is

$$ B = \begin{bmatrix} 0.7 & 0.3 & 0.4 \\ 0.3 & 0.7 & 0.2 \\ 0.4 & 0.2 & 0.7 \end{bmatrix}. $$

The mean reciprocal rank (MRR) and the mean normalized rank given in Figure 2.

For these two settings we choose $d = 3$ for the adjacency spectral embedding step, that is, we embed the graphs into $\mathbb{R}^3$. We recall that our algorithm requires alignment of these embeddings either via orthogonal Procrustes or via an adaptive point set registration. These two choices lead to two slightly different quadratic program formulation. More specifically, as we are embedding into $\mathbb{R}^3$, the orthogonal Procrustes procedure needs at least three seed vertices to align the embeddings. These seed vertices can then be incorporated into the quadratic program in Section 2.3. In contrast, if the embeddings are aligned using adaptive point set registration, then seed vertices are not necessary and hence the quadratic program is formulated with no seeds. When using orthogonal Procrustes, we also explore the impact of $K$, the number of seed vertices. We find that increasing $K$ does improve our algorithm, but that the improvement is not overly substantial in the regime where $K$ is small. For example, in the $\rho$-RDPG setting with $\rho = 0.5$, increasing $K$ from 3 to 9 increases the MRR from 0.28 to 0.37 for the orthogonal Procrustes step, thus, for simplicity of presentation, we fixed $K = 6$ and show the results.

The mean reciprocal rank and mean normalized rank of our algorithm, as a function of the correlation coefficient $\rho$, are presented in Figures 1 and 2. Our algorithm is generally quite accurate. In particular, even when $\rho = 0$ our algorithm still performs substantially better than the baseline. We also note that the performance of orthogonal Procrustes (using $K = 6$ seeds) and adaptive point set registration (with no seeds) are similar, with the difference being even less pronounced in the $\rho$-SBM setting. We posit that the more obvious community structure in the SBM setting helps the adaptive point set registration procedure to align the embeddings more accurately, thereby reducing the need for seed vertices.

We next explore how the reranking step in Section 3.1 can improve the performance of our algorithm, especially when there are enough seed vertices. More specifically, we set $\rho = 0.7$ and vary the number of seed vertices $K$ from 10 to 50. These seed vertices are incorporated into both the quadratic program formulation and the reranking step. We then compare the performance of our algorithm with and without the reranking step. The MRR averaged over 500 Monte Carlo replicates are presented in Figure 3 for the $\rho$-RDPG setting and in Figure 4 for the $\rho$-SBM setting. These figures indicate that the reranking step leads to significant improvement even for small values of $K$, for example, compare the MRR in the $\rho$-SBM setting with $K = 10$ seeds.

Appendix C, supplementary materials contains additional simulation results illustrating how the choice of embedding dimension $d$, the sparsity parameter $\gamma$, and the penalty parameter $\lambda$ affects the performance of our algorithm. In particular, Figures C1 and C2, supplementary materials show that our algorithm is relatively robust to the choice of $d$ while Figure C5, Table C2, and Table C3, supplementary materials show that our algorithm is also relatively robust to the choice of $\gamma$ and $\lambda$, respectively. In addition, we also include in Appendix C, supplementary materials comparisons between our algorithm and the embedding followed by Gaussian mixture modeling algorithm of Agterberg et al. (2020), and we see from Figures C7 and C8, supplementary materials that the accuracy of Agterberg et al.’s (2020) method and our algorithm with orthogonal Procrustes are very similar but we note the running time of Agterberg et al.’s (2020) method is roughly six times slower than ours, and we see our algorithm with the adaptive point set registration procedure needs no seeds information but also has comparable accuracy.

4.2. Real Data Experiments

We now explore the practical application of our algorithms on real data. In Section 4.2.1, we consider a pair of high-school friendship networks containing some of the same vertices and in which we would like to identify the same individuals across the two networks. In Section 4.2.2, we explore the graphs derived from Microsoft Bing entity graph transitions.

4.2.1. High School Friendship Networks

We first focus on the high school friendship network data from Mastrandrea, Fournet, and Barrat (2015). This dataset contains two observed graphs and, for each graph, the vertices represent students and the edges represent their friendship. The first graph is extracted from the Facebook social network, that is, if two individuals are friends on Facebook, then they are adjacent. The second graph is created based on the result of a survey of the students; for every pair of students, they are considered adjacent if at least one of the students in this pair reports that they are friends with another student. There are 156 vertices in the first graph, 134 vertices in the second graph, and 82 vertices shared between the two graphs. These 82 shared vertices will induce the notion of interestingness for our subsequent analysis. In other words, given a query vertex $x$ in one graph, with $x$ being one of the 82 shared vertices, we are interested in finding the same vertex $x$ in the second graph. This application is thus analogous to that of network deanonymization.

As the number of unshared vertices is reasonably large, we consider two experimental setups. In the first setup we used only the subgraphs induced by the 82 shared vertices while in the other setup we used the full graphs on 156 and 134 vertices. For the adjacency spectral embedding step we set $d = 2$. Orthogonal Procrustes alignment of the embeddings then requires at least 2 seed vertices.

For the experiment using only the shared vertices we iteratively consider each vertex as the vertex of interest. For each vertex of interest we choose a pair of seed vertices, align the embeddings using orthogonal Procrustes, and then solve a quadratic program to obtain a nomination list (the seed vertices are not
Figure 1. Performance of our algorithm for pairs of $\rho$-RDPG graphs on $n = 300$ vertices. The mean reciprocal rank (MRR) and mean normalized rank (MNR) are computed based on 500 Monte Carlo replicates. The MRR and MNR are plotted for different values of the correlation coefficient $\rho$. The red and green lines correspond to the case where the graphs embeddings are aligned via orthogonal Procrustes and via the adaptive point set registration procedure, respectively. The dotted blue lines correspond to the baseline MRR and MNR for a nomination list chosen uniformly at random.

Figure 2. Performance of our algorithm for pairs of $\rho$-SBM graphs on $n = 300$ vertices. The mean reciprocal rank (MRR) and mean normalized rank (MNR) are computed based on 500 Monte Carlo replicates. See the caption to Figure 1 for further descriptions of the various colored lines.

Figure 3. Performance of our algorithm with and without the reranking step for pairs of $\rho$-RDPG graphs on $n = 300$ vertices and correlation $\rho = 0.7$. The mean reciprocal rank (MRR) are computed based on 500 Monte Carlo replicates. The red and green lines correspond to the case where the graphs embeddings are aligned via orthogonal Procrustes and via the adaptive point set registration procedure, respectively. In each plot, the corresponding dashed red or green line describes the result after the reranking step. The dotted blue lines correspond to the baseline MRR for a nomination list chosen uniformly at random.

used in the quadratic program). We repeat this procedure 100 times for each vertex of interest, each time choosing a random pair of seed vertices. Figure 5 then illustrates, for each of the 82 possible vertex of interest $x$, how often $NR(x) \in \{0, (0,0.2), (0.2,0.5), (0.5,1)\}$; the mean normalized rank for a nomination list chosen uniformly at random is 0.5. Figure 5 indicates that the nomination lists obtained by our algorithm are in general quite accurate; indeed, the normalized rank values are small for most of the nomination lists, with a significant portion of the nomination lists even having normalized rank values of 0, that is, the true correspondence of the vertex of interest is at the top of the nomination list.

We next consider the impact of increasing the number of seed vertices $K$. For simplicity, we present our analysis for a randomly chosen vertex of interest $x = 27$ as an example. Similar results hold for other vertices. We vary $K$ from 2 to 10 and run 500
Figure 4. Performance of our algorithm with and without the reranking step for pairs of $\rho$-SBM($X$) graphs on $n = 300$ vertices and correlation $\rho = 0.7$. See the caption to Figure 3 for further descriptions.

Figure 5. Performance of our algorithm for vertex nomination between the two high-school networks. Here we consider only the subgraphs induced by the 82 shared vertices. The graphs embeddings are aligned via orthogonal Procrustes transformation using two randomly selected seeds; these seeds are only used for the alignment and are not incorporated into the quadratic programming step. For each $x \in V_1$ we repeat this random seeds selection 100 times and record the normalized rank of its correspondence $\sigma(x) \in V_2$. The four categories correspond to the case when the normalized rank (NR) is equal to 0, lying between 0 and 0.2, lying between 0.2 and 0.5, or larger than 0.5.

Monte Carlo replicates to compute the MNR. We tabulate how often $\text{NR}(x) \in [0, 0.2], (0.2, 0.5), (0.5, 1]$ in Figure 6. We see from Figure 6 that $K = 7$ seed vertices is sufficient for the NR of the nomination lists for $x = 27$ to be between 0 and 0.2 always.

Analogous results are available when we align the embeddings using adaptive point set registration procedure. However, since adaptive point set registration does not use any seed vertex, it lead to more robust performance when compared to using orthogonal Procrustes. Finally, we note in passing that our algorithm is quite computationally efficient, for example, generating Figure 5 takes us only about 7 min on a normal laptop.

We now consider the setup using the full graphs on 134 and 156 vertices. Once again we use orthogonal Procrustes to align the embeddings. We then consider each of the 82 shared vertices as the vertex of interests $x$ and find the nomination list $\ell(x)$ using the same procedure as that outlined above for the setup using the induced subgraphs. Note that the main difference between the current setup and that of the induced subgraphs is that, for each vertex of interest, there are more candidate vertices in the current setup; this make the task harder and hence the performance of our algorithm is likely to be worse in the current setup. The experiment results in Figure 7 confirmed this speculation. Indeed, comparing Figures 5 and 7, we see that the number of times in which the obtained nomination list is no better than chance increases. Nevertheless, our algorithm is still quite accurate since, for almost all of the vertex of interests, the true correspondences do appear frequently at the top of the nomination lists.

4.2.2. Microsoft Bing Entity Graph Transitions

In this section, we consider graphs derived from one month of Bing entity graph transitions. The dataset for this example is from Agterberg et al. (2020) and contains two graphs on the same set of vertices; these vertices denote entities. The (weighted) edges in each graph represent transition rates between the entities during an internet browsing session, but the types of transitions differ between the two graphs. More specifically, the edges in the first graph $G_1$ represents transitions that were made using a suggestion interface while the transitions in the second graph $G_2$ were made independently of any suggestion interface. As the suggestion interface can only suggest a few entities at a time, the edges in $G_1$ are much more constrained than those in $G_2$. The first and second graphs both have 13,535 vertices and approximately $5.2 \times 10^5$ and $5.9 \times 10^5$ edges, respectively. There is, once again, a one-to-one correspondence between the vertices in both networks and we use this correspondence to define our notion of interestingness,
For applying the algorithm with orthogonal Procrustes to subgraphs of high-school network generated by shared vertices, using $x = 27$ as the vertex of interest, we vary the number of seed vertices $K$ from 2 to 10, uniformly at random generate 500 sets of seed vertices and plot NR.

Figure 6.

**Figure 7.** Performance of our algorithm for vertex nomination between the two high-school networks. Here we consider the graphs with full vertices. The graphs embeddings are aligned via orthogonal Procrustes transformation using two randomly selected seeds; these seeds are only used for the alignment and are not incorporated into the quadratic programming step. See the caption to Figure 5 for further descriptions of the experiment.

**Figure 8.** Performance of our algorithm for vertex nomination between the two Microsoft Bing entities transition networks on $n = 1000$ vertices. For each $x \in V_1$ we randomly selected 10 seeds and record the normalized rank (NR) of its correspondence $\sigma(x) \in V_2$. The red and green histogram of NR correspond to the case where the graph embeddings are aligned via orthogonal Procrustes and via the adaptive point set registration procedure, respectively. The last green figure corresponds to the case for adaptive point set registration procedure without any seeds.

that is, given a vertex $x$ in one graph, we are interested in finding the same vertex in the other graph.

For our first analysis we subsample the graphs and only consider the subgraphs induced by the first 1000 vertices. These induced subgraphs are also unweighted, that is, two vertices are adjacent in a induced subgraph if the corresponding transition rate in the original graph is nonzero. Denoting by $G_1$ and $G_2$ the resulting induced subgraphs, $G_1$ and $G_2$ have 8365 edges and 10,247 edges, respectively. We emphasize that there is a 1-to-1 correspondence between the vertex sets of $G_1$ and $G_2$.

We now explore the performance of our algorithm for vertex nomination between $G_1$ and $G_2$. In particular, we sequentially consider each vertex $x \in G_1$ as the vertex of interest, and for a given vertex of interest we randomly select 10 other vertices as seeds. After computing the NR for all vertices, we present the histogram of NR to show the distribution. The results are given in Figure 8 for both the cases where the graph embeddings are aligned via orthogonal Procrustes and via adaptive point set registration. We emphasize that there are two variants of adaptive rigid point set registration used here. In the first variant
the 10 seed vertices are used in the quadratic programming formulation while in the second variant the seed vertices are not used at all. Figure 8 indicates that the normalized rank values are generally quite small and hence the nomination lists returned by our algorithm are accurate. Figure 8 also indicates that there is almost no difference between using orthogonal Procrustes and using adaptive point set registration and, more importantly, our algorithm perform well even when there are no seeds information, that is, the performance of adaptive point set registration with no seeds is virtually identical to that of orthogonal Procrustes and adaptive point set registrations with 10 seeds. Indeed, Table 1 summarizes the quantiles of the NR for different variants of our algorithm and we see from these quantiles that the performance of the three variants are virtually indistinguishable.

For our second analysis we do not sub-sample the graphs and hence $G_1$ contains 13,535 vertices and 519,389 edges while $G_2$ contains the same 13,535 vertices and 595,047 edges. Once again we sequentially consider each vertex $x \in G_1$ as the vertex of interest. For a given vertex of interest we randomly select 10 other vertices as seeds. We then take the induced subgraph of interest. For a given vertex of interest we randomly select 10 seeds and our algorithms are applied on the 1-neighborhood of these vertices set registration procedure, respectively.

Table 1. Quantile levels of normalized rank (NR) values for vertex nomination with the Bing entity networks on $n = 1000$ vertices

|                      | 1% | 5% | 10% | 25% | 50% | 75% | 95% | 99% |
|----------------------|----|----|-----|-----|-----|-----|-----|-----|
| Procrustes (10 seeds) | 0.003 | 0.013 | 0.030 | 0.074 | 0.196 | 0.387 | 0.750 | 0.876 |
| Set registration (10 seeds) | 0.002 | 0.012 | 0.025 | 0.073 | 0.196 | 0.387 | 0.750 | 0.877 |
| Set registration (no seeds) | 0.002 | 0.013 | 0.025 | 0.073 | 0.196 | 0.386 | 0.757 | 0.876 |

Table 2. Quantile levels of normalized rank (NR) values for vertex nomination with the Bing entity networks on $n = 13,535$ vertices

|                      | 1% | 5% | 10% | 25% | 50% | 75% | 95% | 99% |
|----------------------|----|----|-----|-----|-----|-----|-----|-----|
| Procrustes (10 seeds) | 0.000 | 0.006 | 0.018 | 0.073 | 0.211 | 0.429 | 0.869 | 0.991 |
| Set registration (10 seeds) | 0.000 | 0.008 | 0.022 | 0.081 | 0.220 | 0.440 | 0.894 | 0.992 |

the Bing data; see Figure C3, Table C1, Figure C4, and Table C4, supplementary materials. In addition, Figure C6 and Table C5, supplementary materials illustrate how the reranking step also improves the performance for these real data applications; for example for the Microsoft Bing data, Table C5 in Appendix C.4, supplementary materials shows that with 100 seeds the 25th percentile of the normalized rank changes from 0.073 to 0.027 after we apply the reranking step. It means that without the reranking step, 25% of the time we can find the corresponding vertex among the first 73 vertices of the nomination list, and with the reranking step, 25% of the time we can find the corresponding vertex among the first 27 vertices of the nomination list. This a substantial improvement and indicates that while we do not expect the Microsoft Bing entity graph transitions data to follows a $\rho$-correlated GRDPG model, the $\rho$-correlated GRDPG model still provides a useful surrogate for analyzing the pairwise correlations between the edges of the two Bing graphs. Finally, Appendix C, supplementary materials also presents comparisons between our algorithm and the algorithm of Agterberg et al. (2020). For the high school friendship data, we see from Figure C9, supplementary materials that our algorithm is much more accurate and we note the running time of our algorithm with orthogonal Procrustes is only about 7 min compared to their running time which is roughly 200 min on the same laptop. Meanwhile, for the Bing data, the algorithm in Agterberg et al. (2020) and our algorithm using either the orthogonal Procrustes alignment or adaptive point set registration alignment have similar normalized rank distribution (see Table C6), but Agterberg et al. (2020) algorithm is roughly eight times slower than our algorithm.

5. Conclusion

In summary, the current article provides an algorithm for solving the vertex nomination problem in the multi-graphs setting.
Our algorithm depends on adjacency spectral embedding and followed by solving a quadratic programming. To eliminate nonidentifiability of spectral embedding for different graphs, besides an approach based on orthogonal Procrustes, we propose a method using adaptive point set registration to align the embedding that also work without needing any information about seed vertices. Under mild assumption, we establish theoretical guarantee on the consistency of our nomination scheme. The empirical results on the simulation and real data analysis demonstrate that our algorithm is generally quite accurate even when there are only a few seeds or even no seed vertices. As we allude to in the introduction of this article, vertex nomination is an unsupervised learning problem and thus evaluation of a vertex nomination algorithm usually requires some underlying ground truth. The real data analysis examples of this article are based on pairs of graphs with shared vertices and we used these shared vertices to define our groundtruth; the resulting analysis is thus similar to network deanonymization. When there is no known groundtruth, then our proposed methodology can be used for exploratory data analysis or for suggesting possible matches between a query vertex $x$ in one graph and vertices most “similar” to $x$ in the second graph. To evaluate the accuracy of the resulting nominations will, however, require additional domain knowledge or domain experts. We believe that our chosen examples are simple to describe and yet sufficiently rich in scope, thereby providing a clear and compelling illustration of the effectiveness of our proposed methodology.

While the proposed algorithm is reasonably computationally efficient, there are still technical challenges in applying the algorithm to large graphs. For example, the Ring graphs analyzed in this article are on the order of $10^4$ vertices and $10^5$ to $10^6$ edges and our algorithm takes roughly 30 min for one full analysis when running on a consumer laptop. For larger-scale graphs, such as those on $10^5$ vertices and $10^7$ edges, our algorithm breaks down. In particular, the EM steps in the adaptive point set registration algorithm can be quite slow to converge and thus might require subsampling of the embedded points before performing the alignment. Furthermore, the quadratic programming step requires keeping track of the assignment matrix $D$; a naive approach of storing $D$ will require too much memory, especially since $D$ is likely to be sparse throughout the optimization. Development of iterative procedures for storing and updating $D$ is thus essential for scaling our algorithm to large graphs. We leave these investigations for future work.

**Supplementary Materials**

**Appendix:** Proof of Theorem 1 and proof of Proposition 1. (pdf file)

**R code for experiments:** R code to perform the algorithm for experiments described in the article. (R files)

**Bing data set:** Dataset used in Section 4.2.2. (RData file)

**Funding**

VL research is sponsored by the Air Force Research Laboratory and DARPA under agreement number FA8750-20-2-1001. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright notation thereon. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of the Air Force Research Laboratory and DARPA or the U.S. Government.

**References**

Agterberg, J., Park, Y., Larson, J., White, C., Priebe, C. E., and Lyzinski, V. (2020), “Vertex Nomination, Consistent Estimation, and Adversarial Modification,” *Electronic Journal of Statistics*, 14, 3230–3267. [1255,1259,1261,1262,1264,1266]

Ahn, S. C., and Horenstein, A. R. (2013), “Eigenvalue Ratio Test for the Number of Factors,” *Econometrica*, 81, 1203–1227. [1256]

Airoldi, E. M., Blei, D. M., Fienberg, S. E., and Xing, E. P. (2008), “Mixed Membership Stochastic Blockmodels,” *Journal of Machine Learning Research*, 9, 1981–2014. [1259]

Blondel, M., Seguy, V., and Rolet, A. (2018), “Smooth and Sparse Optimal Transport,” in *Proceedings of the 21st International Conference on Artificial Intelligence and Statistics*, pp. 880–889. [1258]

Coppersmith, G. A., and Priebe, C. E. (2012), “Vertex Nomination via Content and Context,” arXiv preprint. Available at [http://arxiv.org/abs/1201.4118](http://arxiv.org/abs/1201.4118). [1255]

Duch, J., and Arenas, A. (2005), “Regularized Optimal Transport and the Rot Mover’s Distance,” *The Journal of Machine Learning Research*, 19, 590–642. [1258]

Diaconis, P., and Janson, S. (2008), “Graph Limits and Exchangeable Random Graphs,” *Rendiconti di Matematica, Serie VII*, 28, 33–61. [1258]

Duch, J., and Arenas, A. (2005), “Community Detection in Complex Networks Using Extremal Optimization,” *Physical Review E*, 72, 027104. [1254]

Fishkind, D. E., Lyzinski, V., Pao, H., Chen, L., and Priebe, C. E. (2015), “Vertex Nomination Schemes for Membership Prediction,” *Annals of Applied Statistics*, 9, 1510–1532. [1254,1255]

Fortunato, S. (2010), “Community Detection in Graphs,” *Physics Reports*, 486, 75–174. [1254]

Fraley, C., and Raftery, A. E. (1998), “Mclust: Software for Model-based Cluster and Discriminant Analysis,” Technical report 342, Department of Statistics, University of Washington: Technical Report. [1255]

Han, X., Yang, Q., and Fan, Y. (2019), “Universal Rank Inference via Residual Subsampling with Application to Large Networks,” arXiv preprint arXiv:1912.11583. [1256]

Hoff, P. D., Raftery, A. E., and Handcock, M. S. (2002), “Latent Space Models: First Steps,” *Journal of the American Statistical Association*, 97, 1090–1098. [1258]

Holland, P. W., Laskey, K. B., and Leinhardt, S. (1983), “Stochastic Blockmodels: First Steps,” *Social Networks*, 5, 109–137. [1259,1261]

Hufel, S. V. (1990), “Partial Singular Value Decomposition Algorithm,” *Journal of Computational and Applied Mathematics*, 33, 105–112. [1258]

Incorporated, G. O. (2015), “Gurobi Optimizer Reference Manual,” Available at [http://www.gurobi.com](http://www.gurobi.com). [1258]

Karrer, B., and Newman, M. E. J. (2011), “Stochastic Blockmodels and Community Structure in Networks,” *Physical Review E*, 83, 016107. [1259]

Kozlov, M. K., Tarasov, S. P., and Khachiyan, L. G. (1980), “The Polynomial Solvability of Convex Quadratic Programming,” *USSR Computational Mathematics and Mathematical Physics*, 20, 223–228. [1258]

Kudo, T., Maeda, E., and Matsumoto, Y. (2005), “An Application of Boosting to Graph Classification,” in *Advances in Neural Information Processing Systems*, pp. 729–736. [1254]

Lee, D. S., and Priebe, C. E. (2012), “Bayesian Vertex Nomination,” arXiv preprint. Available at [http://arxiv.org/abs/1205.5082](http://arxiv.org/abs/1205.5082). [1255]

Levin, K. (2017), “Graph Inference with Applications to Low-Resource Audio Search and Indexing,” Ph.D. thesis, Johns Hopkins University. [1255]

Lloyd, J., Orbanz, P., Ghahramani, Z., and Roy, D. M. (2012), “Random Function Priors for Exchangeable Arrays with Applications to Graphs and Relational Data,” in *Advances in Neural Information Processing Systems*, pp. 998–1006. [1254]

[1254,1255,1259,1261,1262,1264,1266]
Lovász, L. (2012), *Large Networks and Graph Limits*, Providence, RI: American Mathematical Society. [1258]

Lyzinski, V., Fishkind, D. E., and Priebe, C. E. (2014), “Seeded Graph Matching for Correlated Erdős–Rényi Graphs,” *Journal of Machine Learning Research*, 15, 3513–3540. [1255,1259]

Lyzinski, V., Levin, K., Fishkind, D. E., and Priebe, C. E. (2016), “On the Consistency of the Likelihood Maximization Vertex Nomination Scheme: Bridging the Gap Between Maximum Likelihood Estimation and Graph Matching,” *Journal of Machine Learning Research*, 17, 1–34. [1255]

Lyzinski, V., Levin, K., and Priebe, C. E. (2019), “On Consistent Vertex Nomination Schemes,” *Journal of Machine Learning Research*, 20, 1–39. [1255]

Mastrandrea, R., Fournet, J., and Barrat, A. (2015), “Contact Patterns in a High School: A Comparison Between Data Collected Using Wearable Sensors, Contact Diaries and Friendship Surveys,” *PloS One*, 10, e0136497. [1262]

Moreno, S., and Neville, J. (2013), “Network Hypothesis Testing Using Mixed Kronecker Product Graph Models,” in *IEEE 13th International Conference on Data Mining*, pp. 1163–1168. [1254,1261]

Myronenko, A., and Song, X. (2010), “Point Set Registration: Coherent Point Drift,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32, 2262–2275. [1257,1262]

Newman, M. E. J. (2006), “Finding Community Structure in Networks Using the Eigenvectors of Matrices,” *Physical Review E*, 74, 036104. [1254]

Onatski, A. (2010), “Determining the Number of Factors from Empirical Distribution of Eigenvalues,” *The Review of Economics and Statistics*, 92, 1004–1016. [1256]

Patsolic, H. G., Park, Y., Lyzinski, V., and Priebe, C. E. (2017), “Vertex Nomination via Seeded Graph Matching,” arXiv preprint. Available at [http://arxiv.org/abs/1705.00674.](http://arxiv.org/abs/1705.00674) [1255,1259]

Pierskalla, W. P. (1968), “The Multidimensional Assignment Problem,” *Operations Research*, 16, 422–431. [1258]

Rubin-Delanchy, P., Priebe, C. E., Tang, M., and Cape, J. (2017), “A Statistical Interpretation of Spectral Embedding: The Generalised Random Dot Product Graph,” arXiv preprint. Available at [http://arxiv.org/abs/1709.05506.](http://arxiv.org/abs/1709.05506) [1258,1259]

Schaeffer, S. E. (2007), “Graph Clustering,” *Computer Science Review*, 1, 27–64. [1254]

Schönemann, P. H. (1966), “A Generalized Solution of the Orthogonal Procrustes Problem,” *Psychometrika*, 31, 1–10. [1256]

Spielman, D. A., and 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, 1–26. [1255]

Sun, M., and Priebe, C. E. (2013), “Efficiency Investigation of Manifold Matching for Text Document Classification,” *Pattern Recognition Letters*, 34, 1263–1269. [1255]

Sun, M., Tang, M., and Priebe, C. E. (2012), “A Comparison of Graph Embedding Methods for Vertex Nomination,” in *11th International Conference on Machine Learning and Applications*, pp. 398–403. [1255]

Sussman, D., Park, Y., Priebe, C. E., and Lyzinski, V. (2020), “Matched Filters for Noisy Induced Subgraph Detection,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 42, 2887–2900. [1255]

Tang, M., Athreya, A., Sussman, D. L., Lyzinski, V., and Priebe, C. E. (2017), “A Semiparametric Two-Sample Hypothesis Testing Problem for Random Dot Product Graphs,” *Journal of Computational and Graphical Statistics*, 26, 344–354. [1254]

Xu, J. (2017), “Rates of Convergence of Spectral Methods for Graphon Estimation,” arXiv preprint. Available at [http://arxiv.org/abs/1709.03183.](http://arxiv.org/abs/1709.03183) [1254]

Yin, H., Benson, A. R., Leskovec, J., and Gleich, D. F. (2017), “Local Higher-Order Graph Clustering,” in *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 555–564. [1254,1255]

Yoder, J., Chen, L., Pao, H., Bridgeford, E., Levin, K., Fishkind, D. E., Priebe, C., and Lyzinski, V. (2020), “Vertex Nomination: The Canonical Sampling and the Extended Spectral Nomination Schemes,” *Computational Statistics & Data Analysis*, 145, 106916. [1254,1255]

Young, S. J., and Scheinerman, E. R. (2007), “Random Dot Product Graph Models for Social Networks,” in *International Workshop on Algorithms and Models for the Web-Graph*, pp. 138–149. Springer. [1258]

Zhang, M., Cui, Z., Neumann, M., and Chen, Y. (2018), “An End-to-End Deep Learning Architecture for Graph Classification,” in *32nd AAAI Conference on Artificial Intelligence*, pp. 4438–4445. [1254]

Zhu, M., and Ghodsi, A. (2006), “Automatic Dimensionality Selection from the Scree Plot via the Use of Profile Likelihood,” *Computational Statistics & Data Analysis*, 51, 918–930. [1256]