Parallel Computing 47 (2015) 70–87

Contents lists available at ScienceDirect

Parallel Computing

journal homepage: www.elsevier.com/locate/parco

# Spectral clustering for divide-and-conquer graph matching

Vince Lyzinski [a,*], Daniel L. Sussman [d], Donniell E. Fishkind [b], Henry Pao [b], Li Chen [b], Joshua T. Vogelstein [c], Youngser Park [b], Carey E. Priebe [b]

[a] Human Language Technology Center of Excellence, Johns Hopkins University, Baltimore, MD 21211, USA
[b] Department of Applied Mathematics and Statistics, Johns Hopkins University, Baltimore, MD 21218, USA
[c] Department of Biomedical Engineering, Johns Hopkins University, Baltimore, MD 21218, USA
[d] Department of Statistics, Harvard University, Cambridge, MA 02138, USA

ARTICLE INFO

*Article history:*
Available online 12 March 2015

*Keywords:*
Graph matching
Adjacency spectral embedding
Clustering
Stochastic block model

ABSTRACT

We present a parallelized bijective graph matching algorithm that leverages seeds and is designed to match very large graphs. Our algorithm combines spectral graph embedding with existing state-of-the-art seeded graph matching procedures. We justify our approach by proving that modestly correlated, large stochastic block model random graphs are correctly matched utilizing very few seeds through our divide-and-conquer procedure. We also demonstrate the effectiveness of our approach in matching very large graphs in simulated and real data examples, showing up to a factor of 8 improvement in runtime with minimal sacrifice in accuracy.



## 1. Introduction

Graph matching is an increasingly important problem in inferential graph statistics, with applications across a broad spectrum of fields including computer vision [38,10], shape matching and object recognition [4,7], and biology and neuroscience [22,34,36], to name a few. The *graph matching problem* (GMP) seeks to find an alignment between the vertex sets of two graphs that best preserves common structure across graphs. Unfortunately, the GMP is inherently combinatorial, and no efficient exact graph matching algorithms are known. Indeed, even the simpler problem of determining if two graphs are isomorphic is famously of unknown complexity [19,30], and if the graphs are allowed to be loopy, weighted and directed, then the simplest version of GMP is equivalent to the NP-hard quadratic assignment problem. Due to its wide applicability, there exist a vast number of approximating algorithms for GMP; see the paper "30 Years of Graph Matching in Pattern Recognition" [11] for an excellent survey of the existing literature.

When matching across graphs, often we have access to a partial matching of the vertices in the form of a *seeding*. In practice, the assumption of seeds is quite natural in many applications. For example, in aligning social networks actors' user names may often allow for a partial alignment to be known a priori. When matching across brain graphs (connectomes), we have geometric information provided by the brain atlas which provides a soft seeding of the vertices. In many time series graphs, it is common to have a group of invariant vertices across time which act as seeds.

In the *seeded graph matching problem* (SGMP), we leverage the information contained in an available partial matching to match the remaining vertices across graphs. Though the literature on seeded graph matching is comparatively small,

* Corresponding author.
*E-mail addresses:* vlyzins1@jhu.edu (V. Lyzinski), daniellsussman@fas.harvard.edu (D.L. Sussman), def@jhu.edu (D.E. Fishkind), hen.pow@gmail.com (H. Pao), lchen87@jhu.edu (L. Chen), jovo@jhu.edu (J.T. Vogelstein), youngser@jhu.edu (Y. Park), cep@jhu.edu (C.E. Priebe).

http://dx.doi.org/10.1016/j.parco.2015.03.004
0167-8191/

recent results point to significant performance improvements in GM algorithms by incorporating even a modest number of seeds [16,27].

Though a myriad of approximate graph matching algorithms exist, the very large graphs arising in the burgeoning realm of "big data" demand highly scalable algorithms. Roughly speaking, existing state of the art algorithms for approximate graph matching can be divided into two classes: those that seek to bijectively match vertices of graphs of the same order, and those that seek matchings between the vertex sets that are allowed to be many-to-many and many-to-one. The current cutting-edge bijective graph matching algorithms achieve excellent performance in approximately matching graphs with thousands of vertices and with computational complexity $O(n^3)$–$n$ the number of vertices being matched; see for example [34,37,15]. These algorithms often operate directly on the adjacency matrices of the graphs to be matched, utilizing the tools of nonlinear optimization to approximately solve GMP directly. However, owing to their $O(n^3)$ complexity, these algorithms are practically unusable, without significant computation resources, for matching very large graphs ($n \approx 10^5$).

Scalability is often achieved via relaxing the bijection requirement and allowing many-to-many and many-to-one matchings. These graph matching procedures can efficiently match very large graphs, often with $n$ in the tens of thousands; see for example [26,1]. A common approach to these scalable inexact algorithms is that they first match smaller, lower dimensional representative objects (prototype graphs in [1], eigenvectors in [26]) and use these to build the overall matching.

Herein, we propose a new divide-and-conquer approach to *scalable bijective* seeded graph matching. Our algorithm, the Large Seeded Graph Matching algorithm (LSGM, see Algorithm 1), merges the approaches of bijective and non-bijective graph matching and leverages the information in seeded vertices in order to match very large graphs. The algorithm proceeds in two steps: We first spectrally embed the graphs—yielding a low dimensional Euclidean representation of our graph—and then use the information provided by seeded vertices to jointly cluster the vertices of the two embedded graphs. This embedding procedure allows us to employ the powerful theory of adjacency spectral embedding (see for example [31,17]) to prove asymptotically perfect performance in *jointly* clustering stochastic block model random graphs, see Theorem 4.1 for detail.

Once the vertices are jointly clustered, we then match the graphs within the clusters. This matching step is fully parallelizable and flexible in that we can employ any one of a number of matching procedures depending on the properties of the resulting clusters. The flexibility afforded by our procedure in the clustering and matching subroutines can have a dramatic impact on algorithmic scalability. For example, on a 1600 vertex simulated graph our parallelization procedure was able to achieve an factor of 8 improvement in speed at minimal accuracy degradation by increasing the number of clusters and hence the number of cores that were used; see Section 5.2.

Though we are not the first to employ a divide-and-conquer approach to graph matching (see for example [9,38,1]), our focus on the efficient utilization of apriori observed seeded vertices and the theoretical framework for our approach provided by Theorem 4.1 set our algorithm apart from the existing literature.

**Note:** All graphs considered herein will be simple; in particular there are no multiple edges between two vertices nor are there edges with a single vertex as both endpoints. Modifications for the directed case are quite simple [31,17] but we do not consider them in this manuscript. All vectors considered will be column vectors, and $\vec{1}_m$ is the length-$m$ vector of all 1*s*. When appropriate we drop the subscript and just write $\vec{1}$. Throughout the paper we employ the standard notation $[n] := \{1, 2, \ldots, n\}$ for any $n \in \mathbb{N}$, and to simplify future notation, if $A \in \mathbb{R}^{n\times n}$ and $\tau, \sigma \subset [n]$, then $A(\tau, \sigma)$ will denote the submatrix of $A$ with row indices $\tau$ and column indices $\sigma$. For a matrix $X, X(:, i)$ will denote the $i$th column of $X$ and $X(i, :)$ the $i$th row of $X$. Also for two matrices $X$ and $Y$, $[X|Y]$ will denote the column concatenation of $X$ and $Y$.

**Algorithm 1.** Divide-and-conquer seeded graph matching; the LSGM algorithm

**INPUT:** Symmetric, hollow $A, B \in \{0,1\}^{n\times n}, s \in [n]$, seeding $\phi : [s] \to [s]$
**OUTPUT:** A matching of $G_1$ and $G_2$ given by $\psi$;
**Step 1:** Embed and jointly cluster the graphs according to Algorithm 2
**Step 2:** In parallel
**for** $i = 1$ to $k$ **do**
  Match cluster $i$ across the graphs using, yielding matching $\psi^{(i)}$;
**end for**
**OUTPUT:** $\psi = \oplus_{i=1}^{k} \psi^{(i)}$.

## 2. Background

There are numerous formulations of the graph matching problem, though they all share the same objective heuristic: given two graphs, $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$, GMP seeks an alignment between the vertex sets $V_1$ and $V_2$ that best preserves structure across the graphs. In *bijective* graph matching, we further assume $|V_1| = |V_2| = n$, and the alignment sought by GMP is a bijection between $V_1$ and $V_2$. In *non-bijective* graph matching, we allow for $|V_1| \neq |V_2|$ and for alignments that are not one-to-one.

In the bijective matching setting, GMP is commonly formulated as follows: find a bijection $\psi : V_1 \to V_2$ minimizing the quantity

$$\left|\left\{(i,j) \in V_1 \times V_1 \text{ s.t. } [i \sim_{G_1} j, \psi(i) \nsim_{G_2} \psi(j)] \text{ or } [i \nsim_{G_1} j, \psi(i) \sim_{G_2} \psi(j)]\right\}\right|, \tag{2.1}$$

i.e. the problem seeks to minimize the number of edge disagreements between $G_2$ and "$\psi(G_1)$" (see [34,37,15]). Equivalently stated, if $A$ and $B$ are the respective adjacency matrices of $G_1$ and $G_2$, then this problem seeks to minimize $\|A - PBP^T\|_F^2$, over all permutation matrices $P \in \Pi(n) := \{n \times n \text{ permutation matrices}\}$, with $\|\cdot\|_F$ the matrix Frobenius norm. In the non-bijective matching setting, $V_1$ and $V_2$ need not have equal cardinality. This requires an alternative formulation of GMP, as (2.1) is no longer necessarily well-defined. See [7,12,26,38] for a variety of generalizations of (2.1).

In the seeded graph matching problem (SGMP), we further assume the presence of a latent alignment $\phi$ between the vertex sets of $G_1$ and $G_2$. Our task is to then efficiently leverage the information in a partial observation of the latent alignment, i.e. a *seeding*, to estimate the remaining latent alignment. In bijective SGMP, we are given subsets of the vertices $S_1 \subset V_1$ and $S_2 \subset V_2$ called *seeds* with $|S_1| = |S_2| = s$ and a bijective seeding function $\phi_S : S_1 \to S_2$. Without loss of generality we may reorder the vertices so that $S_1 = S_2 = [s]$ and $\phi_S = \mathrm{id}$ (the identity function on $S_1$). The task then is to use $\phi_S$ to estimate $\phi$ by finding the bijection extending $\phi_S$ which minimizes (2.1). In the non-bijective setting, to accommodate the fact that the latent alignment need not be one–to–one, we define $\phi$ to be a subset of $V_1 \times V_2$, and we are tasked with using a partial observation of $\phi$ to estimate the remaining latent alignment.

## 3. Divide-and-conquer seeded graph matching

We present the details of the LSGM algorithm, Algorithm 1. In Section 3.1, we describe Steps 1–3 of this algorithm which constitute the divide steps. In Section 3.2, we describe the final step of the algorithm which constitutes the conquer step.

### 3.1. Jointly embedding and clustering the graphs

We begin by describing the embedding and clustering subroutine. The input is the symmetric adjacency matrices $A$ and $B$ of the two graphs to be matched ($G_1$ and $G_2$ respectively), the number of seeds $s \in \mathbb{Z}^+$, the seeding function $\phi_S : [s] \to [s]$, the number of clusters $k$, and the embedding dimension $d \in \mathbb{Z}^+$. Note that the procedure can easily be modified to handle directed graphs as well.

**Algorithm 2.** Jointly embedding and clustering the vertices of two graphs, $G_1$ and $G_2$

**INPUT:** Symmetric $A, B \in \{0,1\}^{n\times n}, s \in \mathbb{N}$, seeding $\phi_S : [s] \to [s], d \in \mathbb{N}, k \in [n]$;
**OUTPUT:** A clustering of the $2n$ embedded vertices into $k$ clusters;
**Step 1:** Compute the first $d$ orthonormal eigenpairs of $A$ and $B$, namely $(U_A, S_A)$ and $(U_B, S_B)$ respectively;
**Step 2:** $\widehat{X} \leftarrow U_A S_A^{1/2}, \widehat{Y} \leftarrow U_B S_B^{1/2}$;
**Step 3:** $\widehat{X}_s \leftarrow \widehat{X}([s],:), \widehat{Y}_s \leftarrow \widehat{Y}([s],:), Q \leftarrow \mathrm{argmin}_{W\in W(d)} \|\widehat{X}_s W - \widehat{Y}_s\|_F$;
**Step 4:** Apply the transformation $Q$ to $\widehat{X}$ obtaining the embedding $\widehat{X}Q$ of $A$;
**Step 5:** Cluster the $2n$ embedded points, $\{\widehat{X}Q(i,:), \widehat{Y}(i,:)\}_{i=1}^n$ into $k$ clusters via the $k$-means clustering procedure;

**Step 1**: Compute the first $d$ eigenpairs of $A$ and $B$. Letting the orthonormal eigen-decompositions of $A = [U_A|\widetilde{U}_A](S_A \oplus \widetilde{S}_A)[U_A|\widetilde{U}_A]^T$ and $B = [U_B|\widetilde{U}_B](S_B \oplus \widetilde{S}_B)[U_B|\widetilde{U}_B]^T$, with $U_A, U_B \in \mathbb{R}^{n\times d}, S_A, S_B \in \mathbb{R}^{d\times d}$ and the diagonals of $(S_A \oplus \widetilde{S}_A)$ and $(S_B \oplus \widetilde{S}_B)$ nonincreasing, we compute only $U_A, U_B, S_A, S_B$.

**Step 2**: Initially embed the vertices of $G_1$ and $G_2$ into $\mathbb{R}^d$ as $\widehat{X} := U_A S_A^{1/2}$ and $\widehat{Y} := U_B S_B^{1/2}$ respectively.

**Step 3**: Let $\widehat{X}_s := \widehat{X}([s],:)$ and $\widehat{Y}_s := \widehat{Y}([s],:)$ be the initial embedding of the seeded vertices. Align the embedded seeded vertices via the orthogonal Procrustes fit problem: for $W(d) := \{W \in \mathbb{R}^{d\times d} : W^T W = I\}$, we set $Q = \mathrm{argmin}_{W\in W(d)} \|\widehat{X}_s W - \widehat{Y}_s\|_F$.

**Step 4**: Align the two embedded adjacency matrices; i.e. we apply the transformation $Q$ to $\widehat{X}$ and obtaining the transformed embedding $\widehat{X}Q$.

**Step 5**: Cluster the $2n$ embedded vertices, $\widehat{X}Q$ and $\widehat{Y}$, into $k$ clusters with the *k-means* algorithm [23]. Let the corresponding cluster centroids be labeled $\{\mu_i\}_{i=1}^k$.

**Remark 3.1.1.** The above procedure can be implemented on very large graphs using efficient SVD algorithms (see for example [6]). Indeed, as we are only interested in the first $d \ll n$ eigenpairs, these can be computed in $O(n^2 d)$ steps for $d \leqslant \sqrt{n}$. In the sparse regime, fast partial singular value decompositions (e.g. IRLBD in [2]) can be effectively implemented on arbitrarily

large graphs. Paired with fast clustering procedures (here, each iteration of $k$-means has complexity $O(dkn)$, and in practice excellent performance can often be achieved with significantly less than $n$ iterations), the above procedure can be effectively run on extremely large sparse graphs.

We do not implement parallelized versions of the SVD procedure or clustering procedure in our algorithm; indeed, even for the large graphs we considered, the partial SVD and direct $k$-means were directly and efficiently computable. Note that there is an extensive literature devoted to parallel SVD and clustering implementations, see [5,3] for more detail. Empirically, we see that the matching step is the most computationally intensive step of our procedure, and the runtime gains possible by parallelizing the SVD and clustering procedures are relatively small compared to the gains achieved by matching in parallel. See Section 5.4 for detail.

Additionally, the orthogonal Procrustes problem in Step 3 can be solved in $O(nd^2)$ time as it involves computing the singular value decomposition of $\widehat{X}_s^T\widehat{Y}_s = USV^T \in \mathbb{R}^{d\times d}$ and setting $Q = U^TV$.

**Remark 3.1.2.** Model selection, more specifically choosing $d$ and $k$, is a difficult hurdle to overcome in spectral clustering (see [32,29] for instance). One way to estimate $d$ is via automated profile likelihood procedures such as [39]. Unfortunately, the procedure in [39] requires computation of the full spectrum, which is computationally intensive. In our simulation examples we assume $d$ is known, and in the real data examples, we use the ideas of [8] to estimate the embedding dimension from a partial SCREE plot. We expect our procedure to work well as long as $d \ll \sqrt{n}$ (see Lemma 4.2 for detail) which we see is the case in our simulated and real data examples.

Our procedure is insensitive to our choice of $k$ provided that

1. The procedure consistently clusters across the graphs—if the optimal matching of $G_1$ and $G_2$ is given by $\phi : V_1 \mapsto V_2$ (in the bijective case), then for all $v \in V_1$, $v$ and $\phi(v)$ are in the same cluster. This is essential for ensuring the accuracy of the subsequent matching step.
2. The clusters are modestly sized (for implementing the subsequent matching procedure).

Note that in practice it is impossible to ensure that the clustering is consistent, and we explore the impact of different values for $k$ (and misclustered vertices) in Section 5.2. Indeed, the accuracy of the algorithm is limited by the initial clustering, and we are presently working to understand the consistency of different clustering procedures in different model settings.

**Remark 3.1.3.** Practically, the particular choice of clustering procedure utilized in Step 5 of Algorithm 2 is of secondary importance. Indeed, we choose the $k$-means clustering procedure (using *Matlab*'s built in k-means solver) because of its ease of implementation and theoretical tractability. The particular clustering procedure can be chosen to optimize speed and accuracy given the properties of the underlying data. See [13] for a review of clustering procedures. Also note that although in many applications a natural $k$ is dictated by the data, we do not need to exactly find $k$. For our graph matching exploitation task we do not need to finely cluster the vertices of our graphs; a gross but consistent clustering would still achieve excellent performance.

**Remark 3.1.4.** While our algorithm is presented for undirected unweighted graphs, we could adapt our approach to directed graphs (we would embed the vertices as in [31]), or weighted graphs (the SVD can easily be run on weighted graphs). We plan to theoretically explore this further in future work.

### 3.2. Matching within clusters

When the desired matching is bijective, we first must resolve disagreements in cluster sizes and adjust the clusters accordingly. More specifically, we need to address the fact that within each cluster, we may have an unequal number of vertices from each of the two graphs. We do this as follows:

(i) Suppose that for each $i = 1, 2, \ldots, k$, cluster $i$ has $c_i$ total vertices (from both graphs combined) with $c_1 \geqslant c_2 \geqslant \cdots \geqslant c_k$. Within cluster $i$, suppose there are $c_i^{(1)}$ vertices from $G_1$ and $c_i^{(2)}$ vertices from $G_2$.

(ii) Resize cluster $i$ to be of size

$$\widetilde{c}_i = 2\left\lceil \frac{c_i^{(1)} + c_i^{(2)}}{2} \right\rceil - 2 \cdot \mathbb{1}\left\{ \sum_{j=1}^{k} \left\lceil \frac{c_j^{(1)} + c_j^{(2)}}{2} \right\rceil \geqslant i + n \right\}. \tag{3.1}$$

To parse out Eq. (3.1), note that ideally we would resize the clusters to be of size $\left\lceil \frac{c_i^{(1)}+c_i^{(2)}}{2} \right\rceil$, but $\sum_i \left\lceil \frac{c_i^{(1)}+c_i^{(2)}}{2} \right\rceil$ may be greater than $n$ (note that it is never greater than $n + 2k$). To account for this, we sequentially (starting from the smallest cluster and working up) remove 2 vertices from each cluster until $\sum_i \widetilde{c}_i = n$.

(iii) Designating all vertices as unassigned, sequentially for $i = 1, 2, \ldots, k$, assign the $\widetilde{c}_i/2$ unassigned vertices from each graph closest (in the $L^2$ sense) to $\mu_i$ to be in cluster $i$.

Note that if the desired output is a non-bijective matching, the above procedure for ameliorating cluster sizes need not be implemented.

Once the cluster sizes are resolved, we can match the two graphs within a cluster using any number of bijective matching algorithms. See Section 5 for performance comparisons of various matching procedures. These matching sub-routines can be run fully in parallel, and if the matching within cluster $i$ is denoted $\psi_i$, then the final output of our algorithm is the full matching $\psi = \oplus_{i=1}^{K}\psi_i$, an approximate solution to the SGMP. To further parallelize our approach, one could implement a multi-thread graph matching procedure as in [25]. However, to run their procedure one needs a machine with a NUMA architecture and OpenMP installed, whereas we focus on a scalable procedure able to be run on a typical computer cluster, without any specialized hardware/software.

**Remark 3.2.1.** First, note that the distances needed to resize the cluster have already been computed by the $k$-means clustering procedure so that the cost incurred by reassigning the vertices is computationally minimal (see Section 5 for empirical evidence of this). Second, we do not focus on modifying existing $k$-means procedures to automatically make the clusters be of commensurate sizes. We view our resizing as a refinement of the original $k$-means procedure, and not as providing a new clustering of the vertices. In practice, our reassigned clusters are very similar to the original $k$-means clusters, often differing in only a few vertices.

**Remark 3.2.2.** In the event that one of the $k$-means clusters is composed of a large majority of vertices from a *single* graph, bijective graph matching might not be sensible. In this case, we can non-bijectively match within each cluster by padding the adjacency matrices with empty vertices to make the graphs of commensurate size (as suggested in [37]), and match the resulting graphs. Vertices matched to isolates could be treated as unmatched, or we could iteratively remove the matched vertices in the larger graph and rematch the graphs, yielding a many-to-many matching.

**Remark 3.2.3.** In these matching procedures, it is not surprising that we obtain best results if we use the seeded vertices to not only cluster but also match the graphs (via the SGM algorithm of [16,27]). We recognize that the other bijective matching procedures [37,15] have not been modified in the literature to accommodate seeded vertices, and we do not pursue the modification here. Our results point to the need for modifying these algorithms to handle seedings, and we expect them to achieve excellent performance when thus modified.

### 3.3. Computational cost of LSGM

The many executions of the bijective matching subroutine can be run in parallel, and if $\tilde{c}$ is the size of the largest cluster of the points, then this step has computational complexity $O\left((\tilde{c}+s)^3\right)$ (assuming that we use all seeds in the matching procedure). If the executions are run in sequence then this step would have complexity $O\left(k(\tilde{c}+s)^3\right)$. If $\tilde{c} = \Theta(n)$ then the computational cost of this step is $O(n^3)$, and we have the same computational bound as the algorithms of [34,37,15]. To deal with this issue of load balancing, we re-cluster any overly large clusters by re-running our embedding and clustering procedure with the same seeding function $\phi$ on (where $\ell_i$ is the set of indices of the unseeded vertices in cluster $i$)

$$A_i = \begin{pmatrix} A'([s],[s]) & A'([s],\ell_i) \\ (A'([s],\ell_i))^T & A'(\ell_i,\ell_i) \end{pmatrix},$$

and $B_i$ (defined analogously) for all $i$ such that the size of the corresponding cluster is overly large. If we are unable to reduce these cluster sizes further, then our algorithm cannot improve upon the existing $O(n^3)$ computational complexity, though we achieve a significantly better lead constant. In this case, we might overcome this hurdle by non-bijectively matching any overly large clusters, as these procedures are often highly scalable.

**Remark 3.3.1.** If there exists an $\alpha > 0$ such that $s = o(n^{1-\alpha}), k = \Omega(n^{\alpha})$ and each cluster is size $O(n^{1-\alpha})$, then the computational cost of the LSGM algorithm is $O(n^2 d)$ for $\alpha \leqslant 1/3$ and $O(n^{3(1-\alpha)})$ for $\alpha > 1/3$ when the matching subroutines are fully parallelized. Hence, a modest number of modestly sized clusters – $\alpha \approx 1/3$ – yields a $O(n^2 d)$ running time for the LSGM algorithm.

### 3.4. Active seed selection

If the number of seeds is large and if the seeds are all used in the matching procedures (i.e. we use SGM to match the clusters), the LSGM algorithm may be computationally unwieldy. To remedy this, we formulate a procedure for active seed selection that aims to optimally choose a computationally tractable number of seeds from $S$ to match across each cluster. If we are matching cluster $i$ of size $c_i$ across $G_1$ and $G_2$, and computationally we can only handle an additional $s_i$ seeds in the

SGM subroutine–so that we are matching $c_i + s_i$ total vertices–then ideally we would want to pick the "best" $s_i$ seeds to use. Luckily, the results of [27] provide a useful heuristic for what defines "best" in this setting.

Ideally, columns of the seed to non-seed adjacency matrix in $G_1$ and $G_2$ would be enough to uniquely identify the unseeded vertices in each graph and this can be achieved with a logarithmic number of randomly chosen seeds [27]. Though this is a limiting result, the result (and its proof) offers insight into how to select the "best" seeds in a finite resource setting. Specifically, we seek to have the columns of the seed to non-seed adjacency matrix maximally distinguish the unseeded vertices. Mathematically, this translates to choosing seeds that have the maximum entropy in their collection of seed-nonseed adjacency vectors. To this end, we formulate the following seed selection algorithm for selecting the seeds to use when matching across cluster $i$ (for $i$ fixed).

Suppose that the desired number of seeds for matching cluster $i$ is $s_i$. To have the columns of the seed to non-seed adjacency matrix maximally distinguish the unseeded vertices, we seek seeds that have maximum entropy contained in their collection of seed-nonseed adjacency vectors. We propose to accomplish this greedily by repeatedly maximizing the (average across the two graphs) entropy increase possible by adding a *single* inactive seeded vertex to our active seed set. Abusing notation, define

$$H^j(\mathcal{S}_i) = H\left[A_j(\mathcal{S}_i, \ell_i^j)\right], \tag{3.2}$$

to be the Shannon entropy of the binary column vectors of the seed to nonseed adjacency matrix in graph $G_j$ with seed set $\mathcal{S}_i \subset \mathcal{S}$ and unseeded vertices $\ell_i^j$ and $H$ is the Shannon entropy function. Initialize $\mathcal{S}_i^{(0)} = \emptyset$ and for $t = 1, 2, \ldots, s_i$, we set $\mathcal{S}_i^{(t)}$ to $\mathcal{S}_i^{(t-1)} \cup \{i_t\}$ where

$$i_t \in \operatorname{argmax}_{i \in [s] \setminus \mathcal{S}_i^{(t-1)}} \left( H^1\left(\mathcal{S}_i^{(t-1)} \cup \{i\}\right) + H^2\left(\mathcal{S}_i^{(t-1)} \cup \{i\}\right)\right). \tag{3.3}$$

Finally, set $\mathcal{S}_i = \mathcal{S}_i^{(s_i)}$.

For example, suppose that we have 4 seeded vertices and 4 unseeded vertices and seed to nonseed adjacency given by:

$$A\left([s], C_i^1\right) = \begin{pmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 \\ 1 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 \end{pmatrix}, \quad B\left([s], C_i^2\right) = \begin{pmatrix} 1 & 1 & 1 & 0 \\ 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 \end{pmatrix}.$$

If we were choosing 3 seeds for subsequent matching, we would choose (in this order): $i_1 = 2$, then $i_2 = 1$ (seed 3 could also have been chosen as there are two maximizers of the entropy), then $i_3 = 3$.

## 4. LSGM and the stochastic block model

In as much as we can partition the vertices of $G_1$ and $G_2$ into consistent clusters, it is natural to model $G_1$ and $G_2$ using the *stochastic block model* (SBM) of [24,35] (details of the model are presented shortly). We then define the clustering criterion for clustering the rows of $\left[\widehat{Y}^T | (\widehat{X}Q)^T\right]^T$ into $k$ clusters via

$$(\widehat{C}, \hat{b}) := \operatorname{argmin}_{C \in \mathbb{R}^{k \times d},\ b:[2n] \to [k]} \sum_{i=1}^{2n} \left\| \left[ \begin{pmatrix} \widehat{Y} \\ \widehat{X}Q \end{pmatrix} \right](i,:) - C(b(i),:) \right\|_2^2, \tag{4.1}$$

where the rows of $\widehat{C}$ are the centroids of the $k$ clusters and $\hat{b}$ is the cluster assignment function. Note that $k$-means attempts to solve (4.1). In Theorem 4.1 we show that, under some mild conditions on the underlying SBM, the optimal cluster assignment $\hat{b}$ almost surely perfectly clusters the vertices of both $G_1$ and $G_2$. We present the necessary background below.

A $d$-dimensional stochastic block model random graph, $G$, has the following parameters: an integer $K \geqslant 2$, a vector of nonnegative integers $\vec{n} \in \mathbb{N}^K$, and a latent–position matrix $X \in [0,1]^{n \times d}$ with $K$ distinct rows. The random graph's vertex set $V$ is the union of the *blocks* $V_1, V_2, \ldots, V_K$, which are disjoint sets with respective cardinalities $n_1, n_2, \ldots, n_K$. For each $v \in V$, let $b(v)$ denote the block of $v$, i.e. $v \in V_{b(v)}$. Lastly, for each pair of vertices $\{v, v'\} \in \binom{V}{2}$, the adjacency of $v$ and $v'$ is an independent Bernoulli trial with probability of success $D(v, v')$, where $D := XX^T$.

Two independent SBM graphs may have no correlation structure between them, and there is no natural bijective alignment of their vertices. To induce this alignment, we introduce correlation between the graphs. We say that two (matched) random graphs $G_1$ and $G_2$ from this model have correlation $\rho \in [0,1]$ if the set of indicator random variables

$$\left\{ \mathbb{1}_{v \sim_{G_1} v'}, \mathbb{1}_{w \sim_{G_2} w'} \right\}_{\{v,v'\},\{w,w'\} \in \binom{V}{2}}$$

are mutually independent except that for each $\{v, v'\} \in \binom{V}{2}$, the indicator random variables $\mathbb{1}_{v \sim_{G_1} v'}$ and $\mathbb{1}_{v \sim_{G_2} v'}$ have Pearson product-moment correlation coefficient $\rho$. Such correlated graphs can be easily constructed by realizing $G_1$ from the underlying SBM and then, for each $\{v, v'\} \in \binom{V}{2}$, $\mathbb{1}_{v \sim_{G_2} v'}$ is an independent Bernoulli trial with probability of success $D(v, v') + \rho(1 - D(v, v'))$ if $v$ and $v'$ are adjacent in $G_1$, and probability of success $D(v, v')(1 - \rho)$ if $v$ and $v'$ are not adjacent in $G_1$. If $G_1$ and $G_2$ are thus correlated, then there is a natural latent alignment between the vertices of the graphs, namely the identity function $\mathrm{id}_n$.

Given $\vec{m} \in \mathbb{N}^K$ such that $\vec{m} \leqslant \vec{n}$ coordinate-wise and $\|\vec{m}\|_1 = s$ (the number of seeds), the random graphs $G_1$ and $G_2$ from the $d$-dimensional stochastic block model parameterized with $K, \vec{n}, X$, and having correlation $\rho$, are $\vec{m}$-seeded if, a priori for each $i = 1, 2, \ldots, K$, $m_i$ of the $n_i$ vertices from block $V_i$ function as seeds for LSGM, i.e. their across graph correspondence is known.

Let $G_1$ and $G_2$ be $\rho$-correlated, $\vec{m}$-seeded (with $\vec{m}^T\vec{1} = s$), $d$-dimensional SBM's parametrized by $K, \vec{n}$, and $X$. Let their respective adjacency matrices be $A$ and $B$, and let their respective block membership functions be $b_A$ and $b_B$. Without loss of generality, let the true alignment function be $\mathrm{id}_n$ and let $b := b_A = b_B$. Consider the transformed (as in Step 4 of Algorithm 2) adjacency spectral embeddings of $G_1$ and $G_2, \widehat{X}Q$ and $\widehat{Y}$, and assume that we have clustered the rows of $\left[\widehat{Y}^T | (\widehat{X}Q)^T\right]^t$ via the optimal $(\hat{C}, \hat{b})$ of (4.1). Adopting the notation of Algorithm 1, define (where again $C_i^j$ is the set of unseeded indices in $G_j$ corresponding to cluster $i$ and $c_i = |C_i^j|$)

$$\psi_s^{(i)} := \mathrm{argmin}_{P \in \Pi(s+c_i)} \left\| \begin{pmatrix} A_s & A\left([s], C_i^1\right) \\ A\left([s], C_i^1\right)^T & A^{(i)} \end{pmatrix} - \begin{pmatrix} I_s & 0 \\ 0 & P \end{pmatrix} \begin{pmatrix} B_s & B\left([s], C_i^2\right) \\ B\left([s], C_i^2\right)^T & B^{(i)} \end{pmatrix} \begin{pmatrix} I_s & 0 \\ 0 & P^T \end{pmatrix} \right\|_F, \tag{4.2}$$

$$\psi_n^{(i)} := \mathrm{argmin}_{P \in \Pi(c_i)} \|A^{(i)} - PB^{(i)}P^T\|_F \tag{4.3}$$

to be the respective optimal seeded and unseeded matchings of cluster $i$ across the two graphs. When appropriate, we will drop the subscript and refer to the matching of cluster $i$ as simply $\psi^{(i)}$.

We shall hereto forth be considering a sequence of growing models with $n = 1, 2, \ldots$ vertices. In the next theorem, we prove that under modest assumptions, we have that for all but finitely many $n, \hat{b} = b$, and all of the vertices are perfectly clustered across the two graphs. The results of [27] immediately give that $\psi_s^{(i)} = \{I_{s+c_i}\}$ a.a.s. and $\psi_n^{(i)} = \{I_{c_i}\}$ a.a.s. for all $i = 1, 2, \ldots, K$ and the above procedure (when perfected implemented) correctly aligns the two SBM graphs.Although this result is asymptotic in nature, it provides hope that our two-step procedure will be effective in approximating the true but unknown alignment across a broad spectrum of graphs.

**Theorem 4.1.** *With notation as above, let $G_1$ and $G_2$ be $\vec{m}$-seeded (with $\vec{m}^T\vec{1} = s$), $d$-dimensional SBM's parametrized by $K, \vec{n}$, and $X$. Although we assume $G_1$ and $G_2$ have the same block structure, we make no assumptions about the correlation structure. Let their respective adjacency matrices be $A$ and $B$, and without loss of generality let the true alignment function be $\mathrm{id}_n$, so that the block membership function is $b := b_A = b_B$. Adopting the notation of Section 3, if the following assumptions hold:*

(i) *There exist constants $\epsilon_1, \epsilon_2 > 0$ such that $K = O(n^{1/3-\epsilon_1})$ and $\min_i \vec{n}(i) = \Omega(n^{2/3+\epsilon_2})$;*

(ii) *Defining*

$$\delta_d := \min_{i,j \leqslant d+1, i \neq j} \left| \lambda_i(XX^T) - \lambda_j(XX^T) \right| / n, \tag{4.4}$$

*and*

$$\beta := \beta(n, d, \delta_d) = \frac{260d\log(n)}{\delta_d n^{1/2}}, \tag{4.5}$$

*if $i, j \in [n]$ are such that $X(i,:) \neq X(j,:)$ then $\|X(i,:) - X(j,:)\|_2 > 6n^{1/6}\beta$;*

(iii) *Without loss of generality, let $\{X(i,:)\}_{i=1}^s$ be the latent positions corresponding to the seeded vertices, then we assume there exists an $\alpha$ satisfying $\alpha > 4\beta$ and $\sqrt{n}\beta/\alpha = o(n^{\epsilon_2/2}d/\delta_d)$ such that*

$$\min_{v: \|v\|_2 = 1} \left\| X([s], :)v^T \right\|_2 \geqslant \alpha\sqrt{s}; \tag{4.6}$$

*then for all but finitely many n, the $\hat{b}$ of (4.1) satisfies $\hat{b} = b$.*

Regardless of the correlation structure, Theorem 4.1 implies that our joint clustering procedure yields a canonical nonbijective matching of the vertices (where the matching is given by the clustering).

Our proof of this theorem will proceed as follows. First we will state some key results proved elsewhere. Then we will bound $\left\|\widehat{X}Q-\widehat{Y}\right\|_{2\to\infty}:=\max_i\left\|(\widehat{X}Q-\widehat{Y})_i\right\|_2$ and will then have that the $2n\times d$ matrix $\left[\widehat{Y}^T|(\widehat{X}Q)^T\right]^T$ is close to a specified transformation of the $\left[X^T|X^T\right]^T$ (recalling from [28] that for a matrix $M\in\mathbb{R}^{a\times b}$, $\|M\|_{2\to\infty}=\max_i\|M(i,:)\|_2$). Finally, we will use this to show that the clustering will perfectly cluster the vertices in the two graphs into the $K$ true blocks.(See Fig. 1)

Let $D=\left[U_D|\widetilde{U}_D\right]\left[S_D\oplus\widetilde{S}_D\right]\left[U_D|\widetilde{U}_D\right]^T$ be the orthonormal eigen-decomposition of $D$ with $U_D\in\mathbb{R}^{n\times d}, S_D\in\mathbb{R}^{d\times d}$, and ordered so that the diagonals of $\left[S_D\oplus\widetilde{S}_D\right]$ are nondecreasing. The next lemma collects some necessary results from [31,28] which will be needed in the sequel.

**Lemma 4.2.** *With notation as above, let* $W_A=\operatorname{argmin}_{W\in W(d)}\left\|\widehat{X}-XW\right\|_F$ *and* $W_B=\operatorname{argmin}_{W\in W(d)}\left\|\widehat{Y}-XW\right\|_F$. *If* $d=o(\sqrt{n})$, *then it holds with probability one that for all but finitely many n that*

$$\left\|\widehat{X}-XW_A\right\|_{2\to\infty}\leqslant\beta \text{ and } \left\|\widehat{Y}-XW_B\right\|_{2\to\infty}\leqslant\beta. \tag{4.7}$$

We are now ready to prove the following.

**Lemma 4.3.** *For all but finitely many n it holds that* $\left\|\widehat{X}Q-\widehat{Y}\right\|_{2\to\infty}\leqslant 8\beta/\alpha+2\beta$.

**Proof.** As in Section 3, let $Q:=\operatorname{argmin}_{W\in W(d)}\left\|\widehat{X}([s],:)W-\widehat{Y}([s],:)\right\|_F$ and let $\widetilde{Q}=W_A^\top W_B$. It immediately follows from Eq. (4.7) that $\left\|\widehat{X}\widetilde{Q}-\widehat{Y}\right\|_{2\to\infty}\leq 2\beta$. Clearly

$$\left\|\widehat{X}([s],:)Q-\widehat{Y}([s],:)\right\|_F\leqslant\left\|\widehat{X}([s],:)\widetilde{Q}-\widehat{Y}([s],:)\right\|_F\leqslant 2\beta\sqrt{s}. \tag{4.8}$$

and working in the other direction

$$2\beta\sqrt{s}\geqslant\left\|\widehat{X}([s],:)Q-\widehat{Y}([s],:)\right\|_F\geqslant\left\|\widehat{X}([s],:)(Q-\widetilde{Q})\right\|_F-\left\|\widehat{X}([s],:)\widetilde{Q}-\widehat{Y}([s],:)\right\|_F\geqslant\left\|\widehat{X}([s],:)(Q-\widetilde{Q})\right\|_F-2\beta\sqrt{s}. \tag{4.9}$$

If we let the SVD of $Q-\widetilde{Q}$ be $V_1SV_2^\top$ then

$$\begin{aligned}\left\|\widehat{X}([s],:)(Q-\widetilde{Q})\right\|_F&\geqslant\left\|X([s],:)W_A(Q-\widetilde{Q})\right\|_F-\left\|\left(\widehat{X}([s],:)-X([s],:)W_A\right)(Q-\widetilde{Q})\right\|_F\\&\geqslant\left(\sum_{i=1}^{s}\sum_{j=1}^{d}\langle X(i,:),W_AV_1(:,j)\rangle S(j,j)^2\right)^{1/2}-2\beta\sqrt{s}\left\|Q-\widetilde{Q}\right\|_F\geqslant(\alpha-2\beta)\sqrt{s}\left\|Q-\widetilde{Q}\right\|_F\end{aligned} \tag{4.10}$$

by the assumption (Eq. 4.6) that $\min_{\|v\|_2=1}\|X([s],:)v\|_2^2\geqslant\alpha^2 s$ and Eq. (4.9). Combined with Eq. (4.8), we have $\left\|Q-\widetilde{Q}\right\|_{2\to 2}\leqslant\left\|Q-\widetilde{Q}\right\|_F\leqslant\frac{4\beta}{\alpha-2\beta}$. Hence, we have that

$$\left\|\widehat{X}Q-\widehat{Y}\right\|_{2\to\infty}\leqslant\left\|\widehat{X}(Q-\widetilde{Q})\right\|_{2\to\infty}+\left\|\widehat{X}\widetilde{Q}-\widehat{Y}\right\|_{2\to\infty}\leqslant\left\|\widehat{X}\right\|_{2\to\infty}4\beta/(\alpha-2\beta)+2\beta\leqslant 8\beta/\alpha+2\beta. \tag{4.11}$$

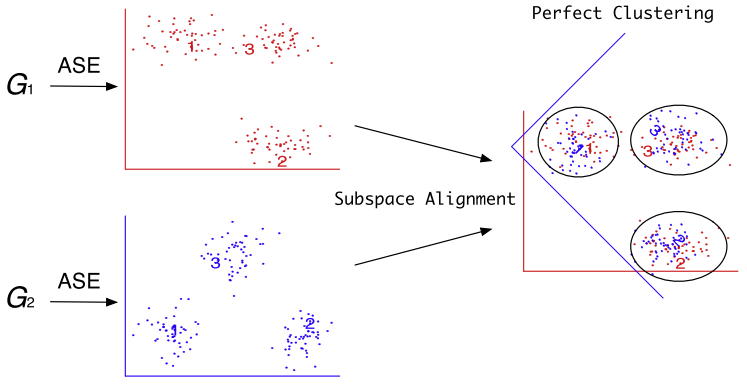



**Fig. 1.** Visual proof sketch of Theorem 4.1. The graphs are first embedded using adjacency spectral embedding (ASE), aligned using the seeded vertices, and perfectly clustered in the aligned space.

since $\left\|\widehat{X}\right\|_{2\to\infty} \leqslant 1$ and $\alpha > 4\beta$. □

**Lemma 4.4.** *For all but finitely many n, it holds that*

$$\left\|\begin{pmatrix}\widehat{Y}\\ \widehat{X}Q\end{pmatrix} - \begin{pmatrix}XW_B\\ XW_B\end{pmatrix}\right\|_{2\to\infty} \leqslant \frac{8\beta}{\alpha} + 3\beta.$$

**Proof.** We have

$$\left\|\begin{pmatrix}\widehat{Y}\\ \widehat{X}Q\end{pmatrix} - \begin{pmatrix}XW_B\\ XW_B\end{pmatrix}\right\|_{2\to\infty} = \max\{\|\widehat{Y} - XW_B\|_{2\to\infty}, \|\widehat{X}Q - XW_B\|_{2\to\infty}\}. \tag{4.12}$$

The first term in Eq. (4.12) is bounded by $\beta$ by Eq. (4.7). For the second term we have from Eq. (4.11) that $\left\|\widehat{X}Q - XW_B\right\|_{2\to\infty} \leqslant \left\|\widehat{X}Q - \widehat{Y}\right\|_{2\to\infty} + \left\|\widehat{Y} - XW_B\right\|_{2\to\infty} \leqslant \frac{8\beta}{\alpha} + 3\beta$. □

*Pf of Main thm:* Let $\mathcal{B}_1, \mathcal{B}_2, \ldots, \mathcal{B}_K$ be the $L^2$-balls of radius $r := n^{1/6}\beta$ around the $K$ distinct rows of $XW_B$. If $X(i,:) \neq X(j,:)$, then by assumption

$$6n^{1/6}\beta \leqslant \|X(i,:) - X(j,:)\|_2 = \|(X(i,:) - X(j,:))W_B\|_2, \tag{4.13}$$

and the $\mathcal{B}_i$'s are disjoint.

Let $\widehat{Z} = \left[\widehat{Y}^T|(\widehat{X}Q)^T\right]^T$ and let $Z = \left[(XW_B)^T|(XW_B)^T\right]$. Let $\left(\widehat{C}, \hat{b}\right)$ be the optimal clustering of the rows of $\widehat{Z}$ from (4.1). Suppose there is an index $i \in [2n]$ such that $\left\|X(i,:)W_B - \widehat{C}(\hat{b}(i),:)\right\| > 2r$. This would imply that $\left\|\widehat{Z} - \widehat{C}\circ\hat{b}\right\|_F > \sqrt{\min_j \vec{n}(j)}(2r - \beta)$ (where $\widehat{C}\circ\hat{b}$ is the $2n \times d$ matrix whose $i$th row is $\widehat{C}(\hat{b}(i),:)$). As $\min_j \vec{n}(j) = \Omega(n^{2/3+\epsilon_2})$ for a constant $\epsilon_2 > 0$, we would then have that

$$\left\|\widehat{Z} - \widehat{C}\circ\hat{b}\right\|_F = \Omega\left(\frac{n^{\epsilon_2/2}d}{\delta_d}\right). \tag{4.14}$$

Lemma 4.4 yields that

$$\left\|\widehat{Z} - Z\right\|_F \leqslant \sqrt{2n}\left(\frac{8\beta}{\alpha} + 3\beta\right) = o\left(\frac{n^{\epsilon_2/2}d}{\delta_d}\right), \tag{4.15}$$

where the final equality follows from assumption (iii). Combined with Eq. (4.14), this contradicts the minimality of $(\widehat{C}, \hat{b})$, and therefore $\|Z - \widehat{C}\circ\hat{b}\|_{2\to\infty} \leq 2r$.

From (4.7) we have $\left\|\widehat{Z} - \widehat{C}\circ\hat{b}\right\|_{2\to\infty} \leqslant 2r + \beta = (2 + o(1))r$. If $i,j \in [n]$ are such that $\widehat{C}(\hat{b}(i),:) \neq \widehat{C}(\hat{b}(j),:)$, then $\|Z(i,:) - Z(j,:)\|_2 > 6r$, and it follows that

$$\left\|\widehat{Z}(i,:) - \widehat{C}(j,:)\right\|_2 > 4r - \beta = (4 + o(1))r. \tag{4.16}$$

It follows that for all but finitely many $n$, $\hat{b} = [b^T|b^T]^T$. Stated simply,

$$\min_{\pi\in S_K}\left|\left\{v \in V(G_1)\cup V(G_2) : b_n(v) \neq \pi(\hat{b}_n(v))\right\}\right| = 0. \tag{4.17}$$

Now [27, Theorem 1] immediate implies that for all but finitely many $n$, $\psi^{(i)} = \{I_{u_i}\}$ for all $i \in [K]$ and the proof is complete. □

**Remark 4.5.** The implication of assumption (iii) in Theorem 4.1 is that in order for the scaled Procrustes fit of the embedded seeded vectors to align the entire embedding, it is sufficient that the latent positions corresponding to the seeded vectors cannot concentrate too heavily in one direction. We note that analogous assumptions are made in the literature on sparse subspace clustering, see [14] for example and detail.

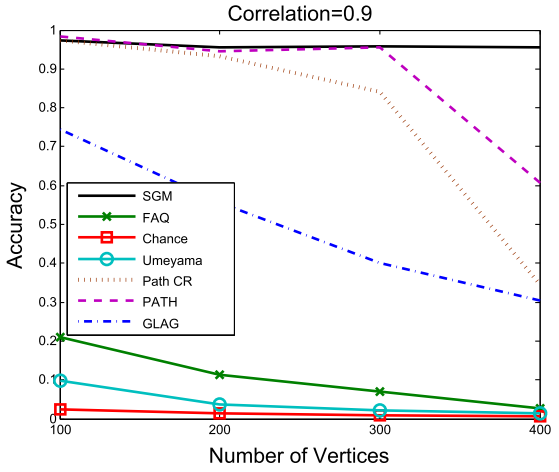
**Remark 4.6.** If there exist constants $\epsilon_1, \epsilon_2 > 0$ such that $K = O(n^{1/3-\epsilon_1})$ and $\min_i \vec{n}(i) = \Omega(n^{2/3+\epsilon_2})$, then the results of [28] demonstrate that the optimal clustering for the one graph analogue of (4.1) perfectly clusters the vertices of a single SBM.

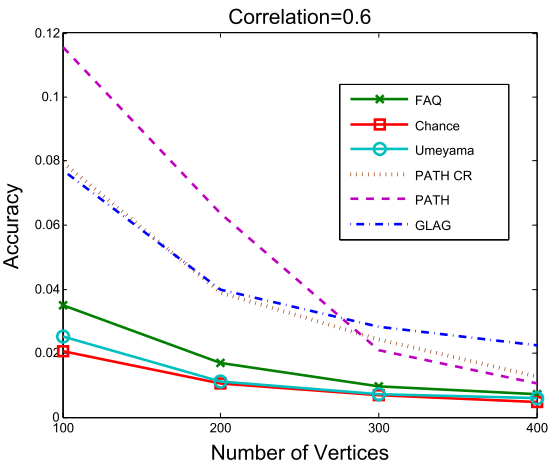
## 5. Empirical results

We next explore the effectiveness of our divide-and-conquer approach on simulated and real data examples. When comparing across graph matching algorithms, we measure effectiveness via the matching accuracy (since we assume a true latent alignment, this amounts to the fraction of vertices which were correctly aligned) and runtime of the algorithms. Across both runtime and accuracy, our algorithm achieves excellent performance: achieving significantly better accuracy than existing scalable bijective matching algorithms (Umeyama's spectral approach [33]), and achieving significantly better accuracy and runtime than the existing state-of-the-art (in terms of accuracy) matching procedures (PATH [37], GLAG [15], FAQ [34]). Unless otherwise specified, all of our experiments are run on a 2x Intel(R) Xeon(R) CPU E5-2660 0 @ 2.20 GHz (with 32 virtual cores and 16 physical cores). We implement all of our code in the software package Matlab limited to 12 parallel threads. Additionally, the code needed to run our algorithm (in Matlab) is publically available for download at https://github.com/lichen11/LSGMcode.

### 5.1. Simulation results

Once the vertices of the two graphs are clustered, we can run the matching procedures in full parallel across the clusters. Our first experiment seeks to understand how available bijective matching algorithms perform (with respect to accuracy and speed), so that we can better understand how to appropriately set the maximum allowed cluster size. To this end, we run the following experiment. We consider two $\rho$-correlated SBM random graphs with the following parameters (where $J_n := \vec{1}_n\vec{1}_n^T \in \mathbb{R}^{n\times n}, I_n$ is the $n \times n$ identity matrix, and $\otimes$ denotes the Kronecker product): each of $\rho = 0.6$ and $0.9, D = I_2 \otimes .3J_{n/2} + .3J_n \in \mathbb{R}^{n\times n}, \vec{n} = [n/2, n/2]$, for each of $n = 100, 200, 300, 400$. We cluster the graphs into 2 clusters and run a variety of *bijective* GM algorithms on these clusters. We record both the performance of the algorithms in recovering the true alignment and the corresponding running time of each algorithm. Note we ran the matching procedures on the two clusters in parallel. The algorithms we ran include SGM [16], FAQ [34], the spectral matching algorithm of Umeyama [33], the PATH algorithm and the associated convex relaxation PATH CR, which is solved exactly using Frank–Wolfe methodology [18] [37], and the GLAG algorithm [15]. See Fig. 2 for the results.




Mean Running Time

| $n =$ | 100 | 200 | 300 | 400 |
|---|---|---|---|---|
| SGM | 0.14 | 0.78 | 2.13 | 4.49 |
| FAQ | 0.51 | 3.12 | 9.13 | 16.67 |
| Umeyama | 0.09 | 0.14 | 0.21 | 0.34 |
| PATH CR | 0.30 | 1.24 | 2.96 | 3.46 |
| PATH | 2.21 | 9.90 | 15.82 | 69.31 |
| GLAG | 8.53 | 33.83 | 109.48 | 261.72 |

Mean Running Time

| $n =$ | 100 | 200 | 300 | 400 |
|---|---|---|---|---|
| SGM | 0.16 | 0.64 | 1.76 | 2.91 |
| FAQ | 0.58 | 3.05 | 9.09 | 15.56 |
| Umeyama | 0.13 | 0.19 | 0.27 | 0.41 |
| PATH CR | 0.42 | 1.25 | 2.73 | 2.36 |
| PATH | 3.70 | 27.00 | 83.60 | 142.63 |
| GLAG | 8.11 | 32.41 | 108.22 | 235.34 |

**Fig. 2.** Mean accuracy (top) and mean runtime (bottom) for graph matching algorithms for $\rho = 0.9$ (left) and $\rho = 0.6$ (right). The parameters for the SBM graph are $D = I_2 \otimes .3J_{n/2} + .3J_n, \vec{n} = [n/2, n/2]$, for each of $n = 100, 200, 300, 400$ and $\vec{m} = [3, 3]$. For each value of $n$, we ran 100 Monte Carlo replicates. Note, the difference in scales for the left and right accuracy plots. We do not include the accuracy results for SGM for $\rho = 0.6$ because they are near 1 and obscure the ordering for the remaining vertices.

To run LSGM, we used $\vec{m} = [3, 3]$ seeds for $\rho = 0.9$ and $\vec{m} = [5, 5]$ seeds for $\rho = 0.6$, all seeds chosen uniformly at random from the two blocks. The seeds are always used in the embedding and clustering procedure, but SGM is the only algorithm to use seeded vertices when matching the clusters. It is not surprising that it achieves best performance. We expect similarly excellent results from the other matching algorithms once they are seeded.

In the $\rho = 0.9$ experiment, we note that, of the nonseeded matching algorithms, PATH and its associated convex relaxation achieve the best results. The PATH CR procedure scales very well in running time but performs progressively worse as $n$ increases. On the other hand, the PATH algorithm's running time scales poorly (as does that of the GLAG algorithm), needing significantly longer running time than SGM or PATH CR across all values of $n$. While PATH and PATH CR achieve similar results to SGM for $n = 100, 200, 300$, the significantly longer run time for PATH and the sharply decreased performance for PATH CR at $n = 400$ hinder these algorithms effectiveness as post-clustering matching procedures. Indeed, to employ these two procedures, we would need to severely restrict the maximum allowed size of our clusters to achieve a feasible running time and/or accurate matchings. We note that seeding GLAG, the PATH algorithm and PATH CR may yield significantly faster running times and less performance degradation as $n$ increases, as seeding FAQ yields both.

SGM is remarkably stable, achieving excellent matching performance across all $n$. This not only indicates that our clustering methodology is consistent across graphs, but points to the importance of using the seeds in the subsequent matching. Here the correlation is very high, and for smaller $n$, PATH and PATH CR perform on par with SGM, suggesting that seeds are less important when matching very similar graphs. We next explore the effect of decreased correlation.

We explore this in the $\rho = 0.6$ experiment, and again we note that SGM significantly outperforms all the nonseeded matching algorithms (with average accuracy $> 99\%$ for all $n$). This points to the consistency of our clustering procedure here. Note that we needed slightly more seeds to achieve this consistency with the lower correlation. Indeed, with three seeds from each cluster, the clustering was not consistent when $\rho = 0.6$, unlike in the $\rho = 0.9$ case.

### 5.2. Robustness to misspecified k

How sensitive is the performance of our algorithm to mis-specifying $k$? We claim that as long as the clusters are consistently estimated, the procedure is relatively insensitive to mis-estimating $k$. Following this reasoning, if our clustering step allows clusters that are larger than $\max_i n_i$, then we would expect our clusters to be consistent and our performance would not degrade significantly. However, if our clustering step does not allow cluster larger than $\max_i n_i$, then we would not expect our clusters to be consistent and our performance would degrade significantly.

To this end, we consider the following experiment. We consider $\rho \in \{0.6, 0.9\}$-correlated SBM's, with 10 blocks each of size $n_i = 100$, and interblock edge probability 0.6 and across block edge probability 0.3. We run 20 MC replicates of divide-and-conquer graph matching with 20 seeds and with the maximum allowed cluster size equal to 100, 200, 300, 400, 500. We summarize results in Fig. 3. Note that we have included the "Oracle" matcher, which gives the maximum number of vertices possibly matched correctly given the clustering.

From the Fig. 3, we see that the performance of SGM again is significantly better than all the other GM algorithms considered, and is also resilient to allowing larger clusters in the $k$-means procedure. This is echoed in the experiment for $\rho = 0.9$, where we see that SGM nearly achieves oracle accuracy across all maximum cluster sizes. We also explore the sensitivity of the LSGM's runtime to the maximum allowed cluster size. Utilizing 12 cores, the average runtimes of the LSGM algorithm (using SGM for matching and $\rho = 0.6$) are $(10.2831, 24.0464, 41.2820, 61.8609, 86.1164)$ seconds for max cluster size equal to $(100, 200, 300, 400, 500)$; indeed, SGM has runtime $O(n^3)$ and is the slowest step of our divide-and-conquer procedure, so we expect to see the runtime increase if the matching subroutines are between bigger graphs. Larger clusters may be more consistent and therefore may lead to better matching performance, but this is achieved at the expense of increased runtime.

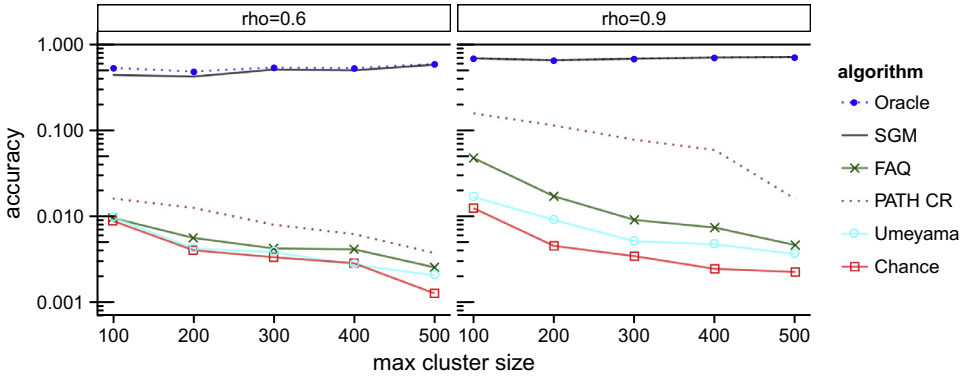



**Fig. 3.** Mean matching accuracy (on the log scale) versus maximum allowed cluster size for graph matching algorithms for $\rho = 0.6$ and $\rho = 0.9, D = I_{10} \otimes .3J_{100} + .3J_{100}, \vec{n} = 100 * \vec{1}$, and 20 seeds randomly selected from the 1000 vertices. For each combination of parameters, we run 20 MC replicates. Note that the oracle and SGM matching overlap heavily. Due to scalability issues, GLAG and PATH were not run in this experiment.

### 5.3. LSGM vs. SGM: the price of embedding

While each of PATH, PATH CR, FAQ and GLAG perform significantly better than chance, again PATH and GLAG scale poorly in running time. The PATH CR algorithm and FAQ scale well in running time but have their matching performance decrease significantly as $n$ increases. PATH and GLAG also see this performance degradation in $n$. In addition, all the algorithms (except SGM) perform significantly worse than the $\rho = 0.9$ case. As real data is, at best, weakly correlated, this points to the primacy of seeding in matching real data graphs. Due to the decreased performance and poor scalability of the nonseeded matching algorithms as $n$ increases, we will henceforth focus our attention on using SGM to match the clusters. Again, we expect the best performing unseeded algorithms (PATH, PATH CR and GLAG) will achieve excellent performance when seeded, though we do not pursue this modification here.

Our two step approach first embeds and clusters the two graphs and then matches them accordingly. Theoretically, we can embed and cluster and then match the graphs perfectly, but we next explore how much accuracy is practically lost because of the embedding step. When $n$ is small (e.g. $\leqslant 1500$) and the SGM algorithm of [16] can be feasibly run without first clustering, the SGM algorithm will outperform LSGM in general, even in the SBM setting. Indeed SGM utilizes the across cluster connectivity structure in the matching task, information which LSGM does not utilize when matching across clusters. It is also clear that SGM is utilizing more of the information contained in the seeding than LSGM. If the latent positions generating the SBMs are separated enough (as at assumption *i.* of Theorem 4.1) and $n$ is large enough for the clustering to be consistent across the graphs, then we will illustrate that LSGM performs excellently. However, even in the case of perfect clustering, LSGM still needs (modestly) more seeds than SGM to achieve comparable performance. We illustrate this in Fig. 4. We match across two $\rho = 0.7$-correlated SBMs with $K = 3$ blocks, $\vec{n} = (200, 200, 200)$, with block–block adjacency probabilities dictated by the matrix

$$\begin{pmatrix} 0.6 & 0.3 & 0.2 \\ 0.3 & 0.7 & 0.3 \\ 0.2 & 0.3 & 0.7 \end{pmatrix},$$

and seed values ranging from $s = 3, 4, 5, 6, 7$ drawn uniformly from the 600 vertices. The dashed curve plots the fraction correctly matched by the SGM algorithm across the various $s$, with error bars $\pm 1 s.e.$ Analogously, the solid curve plots the fraction correctly matched by the LSGM algorithm across the various $s$, with error bars again $\pm 1 s.e.$ Note that with only 4 seeds, SGM perfectly matches across the graphs, though LSGM requires 7 seeds for comparable performance.

Given a consistent clustering of the graphs, LSGM needs modestly more seeds to perform as well as the full SGM. In contrast, if the clustering is not consistent, LSGM cannot hope to match the clusters exactly. However, while SGM can only match graphs of order $\approx 1000$, LSGM can be used to match much larger graphs. We demonstrate this in the following experiment, where we match two large SBM graphs. In Fig. 5, we plot the average accuracy of LSGM in matching the unseeded vertices in 25 MC simulations across two $K = 900$ block, $\vec{n} = 30 \cdot \vec{1}, d = 10$ dimensional, $\rho$-correlated SBM's with $s$ seeds drawn uniformly at random from the 27,000 vertices. The $K$ latent positions $X$ are sampled uniformly from the $d$-dimensional simplex, and we utilize the $k$-means clustering algorithm ($k$ an estimate of $K$) in Step 5 of Algorithm 2. Note how few seeds are needed to ensure good performance for even modestly correlated graphs. For example, we correctly match 78.75% of the

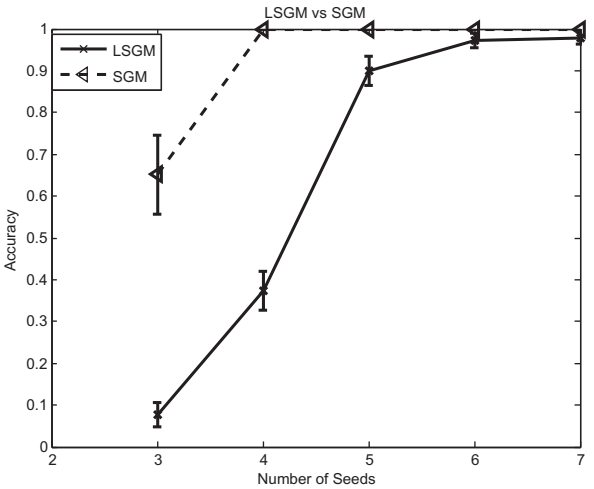



**Fig. 4.** The fraction of the unseeded vertices correctly matched across SBMs with $K = 3$ blocks, block–block connectivity as specified in the text, $\vec{n} = (200, 200, 200), \rho = .7$, and $s = 3, 4, 5, 6, 7$ seeds randomly assigned to one of the three blocks. The dashed curve plots the fraction of unseeded vertices correctly matched by the SGM algorithm across the various $s$, with error bars $\pm 1 s.e.$ The solid curve plots the fraction of unseeded vertices correctly matched by the LSGM algorithm across the various $s$, with error bars again $\pm 1 s.e.$ Here SGM is the algorithm of [16] run without clustering.

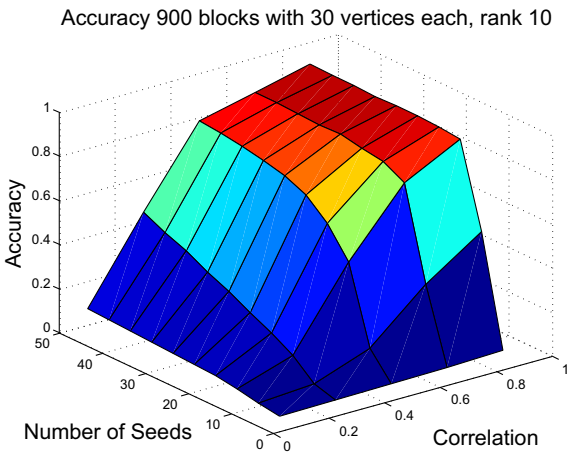



**Fig. 5.** Fraction of unseeded vertices correctly matched across two $K = 900$ block, $\vec{n} = 30 \cdot \vec{1}, d = 10$ dimensional $\rho$-correlated SBM's with $s$ seeds drawn uniformly at random from the 27,000 vertices. Note that for each combination of $s$ and $\rho$, we ran 25 MC simulates. All standard deviations are <.03 except with 10 seeds where the s.d. is $0.0694, 0.2325, 0.1958$ for $\rho = 0.5, 0.7, 0.9$.

unseeded vertices correctly with only 50 seeds and $\rho = 0.5$. This again reflects the consistency of our clustering procedure, and the applicability of our procedure in matching real data graphs, which are (at best) modestly correlated and have (at best) a modest number of seeds.

We do not assume knowledge of the true $K$ in the above procedure, instead estimating an appropriate $k$ from the data. The figure shows that the matching is robust to this estimation. We also do not assume knowledge of the true $d$, and here we used the automated spectral procedure of [39] to estimate the embedding dimension $d$. The model is relatively low rank, and for higher rank SBM's we see slower algorithmic performance in general.

### 5.4. Scalability

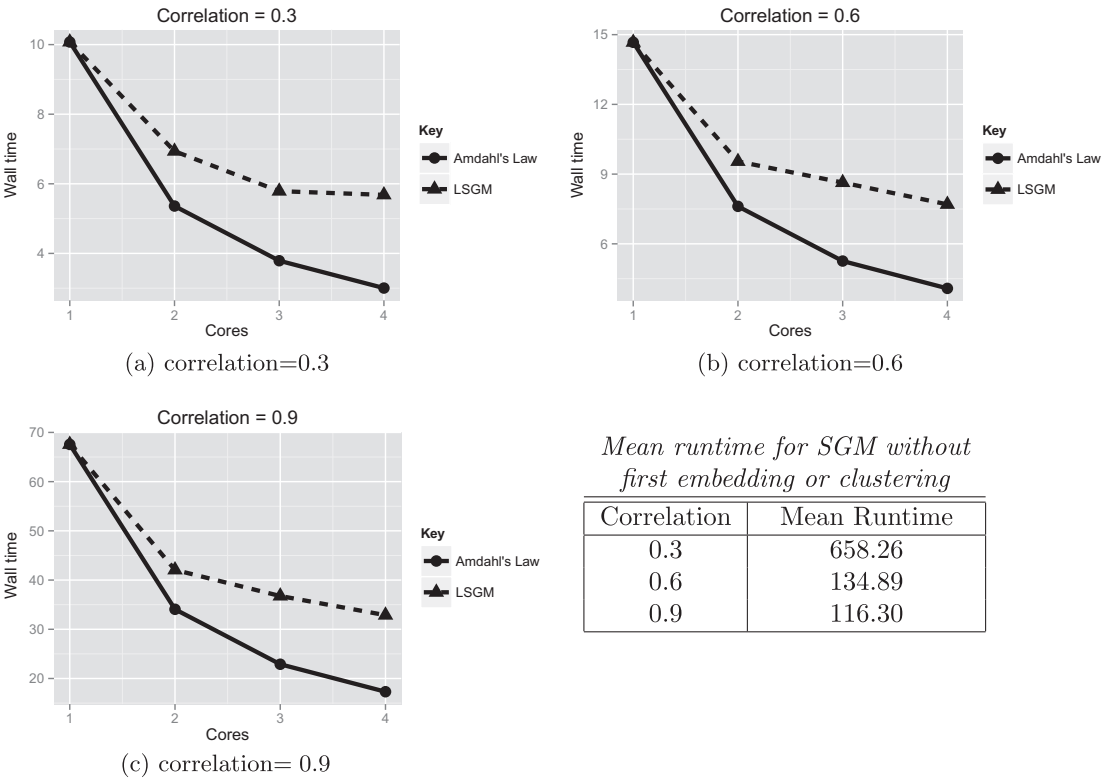
Our divide and conquer algorithm essentially is composed of four steps: embed, Procrustes, cluster, match. The final matching step lends itself to parallelization, and insomuch as the embedding, Procrustes and clustering are computationally less expensive than the subsequent matching step, we expect our algorithm to scale well. Note that we observed this scaling previously in Section 5.2 as well, where we saw that on a 1600 vertex simulated graph our parallelization procedure was able to achieve an 8x improvement in speed at minimal accuracy degradation by increasing the number of clusters and hence the number of cores that were used.

To explore this further, we run our algorithm on three pairs of SBMs with varying $\rho = 0.3, 0.6, 0.9$. Each SBM has 8 blocks (with intrablock connection probability 0.6 and interblock connection probability 0.3) each of size 200, and we run our LSGM procedure with 20 seeds utilizing 1-to-4 cores and, in all cases, clustering the graphs into 8 clusters. We plot the resulting algorithmic wall times in Fig. 6 (run on a Genuine Intel laptop: model name: Intel(R) Xeon(R) CPU E31,290 @ 3.60 GHz with 4 processors). We note that with lower correlation, matching is the most costly step in our procedure as expected, while in the high correlation setting ($\rho = 0.9$), the matching steps are relatively inexpensive. In all cases, we see roughly a $2\times$ speedup in our procedure when utilizing 4 cores. We lastly note that matching these graphs using SGM with $\rho = 0.9$ without first embedding and clustering the graphs has average runtime $\approx 116$ s ($\approx 134$ s when $\rho = 0.6$ and $\approx 658$ s when $\rho = 0.3$), compared with $\approx 5$ s ($\approx 7$ s when $\rho = 0.6$ and $\approx 32$ s when $\rho = 0.3$) with our divide-and-conquer procedure using 4 cores. See Fig. 6 for detail.

For each of the three correlation levels and for each of 1–4 cores, we also calculated the average runtime of each step of our algorithm: embedding, Procrustes, clustering and matching (see the Table 1 for details). We see that matching is the most time intensive aspect of the procedure (especially in the low correlation setting), and that parallelizing the other components of our algorithm would yield incremental runtime improvements when compared to parallelizing the matching step. While parallelizing the other components of our algorithm has been the subject of independent research, the gains in implementing these parallelization strategies are incremental in this setting, and therefore we do not pursue them here.

### 5.5. Connectomes

We next demonstrate the effectiveness of the LSGM algorithm in a practical real data setting. In this data set, for each of 21 subjects, we have two brain connectome graphs. For each subject, the vertices in the connectome graphs correspond to voxels in the $64 \times 64 \times 64$ voxel diffusion tensor MRI brain mask. Edges between vertices are present if there exists at least one neural fiber bundle connecting the voxel regions corresponding to the two vertices. The largest connected component




*Mean runtime for SGM without first embedding or clustering*

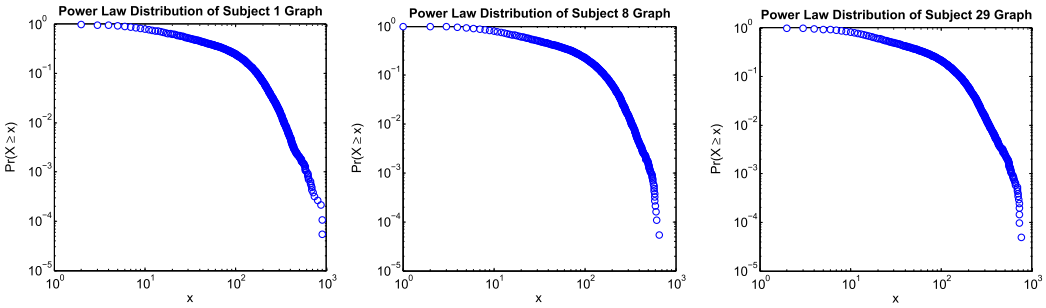
| Correlation | Mean Runtime |
|---|---|
| 0.3 | 658.26 |
| 0.6 | 134.89 |
| 0.9 | 116.30 |

**Fig. 6.** Runtimes when running LSGM and SGM (without first embedding and clustering the vertices) using 1, 2, 3, 4 cores to match two SBM random graphs with 8 blocks each of size 200 (with intrablock connection probability 0.6 and interblock connection probability 0.3). Note, SGM ran on a single core only. For each experiment and each combination or $\rho$ and core number, we run 200 MC replicates, and we ran 20 MC replicates for the SGM experiment. For the full LSGM procedure, we then plot the achieved runtime against the theoretical maximum speedup possible when parallelizing as predicted by Amdahl's law.

**Table 1**
The table show mean runtimes when running LSGM using 1, 2, 3, 4 cores to match two SBM random graphs with 8 blocks each of size 200 (with intrablock connection probability 0.6 and interblock connection probability 0.3). For each experiment and each combination or $\rho$and core number, we run 200 MC replicates. The table shows how the runtime breaks down into the four steps of the algorithm: embedding, procrusties, clustering, and matching.

| *Runtime in seconds* Cores | Embed | Procrustes | Cluster | Match |
|---|---|---|---|---|
| | $\rho = 0.3$ | | | |
| 1 | 0.53 | $0.89 \times 10^{-3}$ | $0.17 \times 10^{-2}$ | 67 |
| 2 | 0.54 | $0.73 \times 10^{-3}$ | $0.18 \times 10^{-2}$ | 41 |
| 3 | 0.54 | $0.72 \times 10^{-3}$ | $0.18 \times 10^{-2}$ | 36 |
| 4 | 0.54 | $0.72 \times 10^{-3}$ | $0.15 \times 10^{-2}$ | 32 |
| | $\rho = 0.6$ | | | |
| 1 | 0.53 | $0.89 \times 10^{-3}$ | $0.19 \times 10^{-2}$ | 14 |
| 2 | 0.53 | $0.73 \times 10^{-3}$ | $0.18 \times 10^{-2}$ | 8.9 |
| 3 | 0.54 | $0.72 \times 10^{-3}$ | $0.19 \times 10^{-2}$ | 8.0 |
| 4 | 0.54 | $0.73 \times 10^{-3}$ | $0.19 \times 10^{-2}$ | 7.1 |
| | $\rho = 0.9$ | | | |
| 1 | 0.53 | $1.06 \times 10^{-3}$ | $1.06 \times 10^{-2}$ | 9.4 |
| 2 | 0.53 | $0.74 \times 10^{-3}$ | $0.20 \times 10^{-2}$ | 6.3 |
| 3 | 0.54 | $0.73 \times 10^{-3}$ | $0.21 \times 10^{-2}$ | 5.2 |
| 4 | 0.54 | $0.72 \times 10^{-3}$ | $0.20 \times 10^{-2}$ | 5.1 |

(LCC) in these connectomes ranges from 20,000 to 30,000 vertices. For more detail on the creation of these graphs and their utility in the neuroscience literature, see [20,21] and the references contained therein. All the data can be found at http://openconnecto.me/graphs (note that we have spatially down-sampled each data point by a factor of four in each dimension).




(a) Graph 1 (b) Graph 8 (c) Graph 29

**Fig. 7.** Degree distribution for connectomes 1, 8 and 29. The degrees are plotted on a log–log scale and are strong evidence of a heavy-tailed degree distribution.

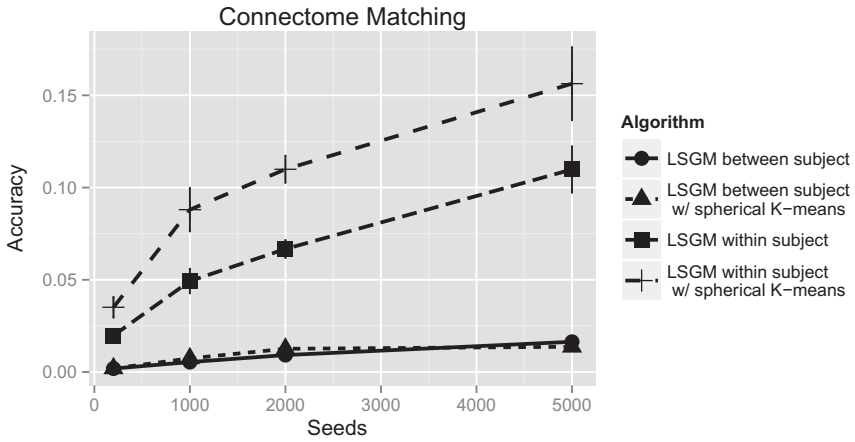



**Fig. 8.** The fraction of the unseeded vertices correctly matched for graphs 8 and 29 (within-subject) and for graphs 1 and 8 (across-subject). For the 8–29 pair, $n = 20,541, d = 30$. For the 1–8 pair, $n = 18,694, d = 30$, we cluster using $k$-means and $sk$-means, reclustering any clusters of size $\geqslant 800$. We plot the fraction of the vertices correctly matched in each of the four experiments for number of seeds $s = 200, 1000, 2000,$ and 5000. Here we ran 5 MC simulates and the error bars are $\pm 2s.e.$

While our theory is proven in the setting of SBM random graphs, this example shows the applicability of our method in matching graphs with heavy-tailed degree distribution. Indeed, when we plot on a log–log scale the degree sequence of two of the connectomes to be matched below, we see all three connectomes have a heavy-tailed degree distribution rather than the flat degree distribution we would expect from the SBM; see Fig. 7 for detail. While our algorithm uncovers significant signal when matching across these connectomes, it will be useful to explore modifications to our approach for accommodating heavy-tailed degree graphs and power-law graphs. We strongly suspect that there is significant signal in the degree distribution, with higher degree vertices being easier to correctly match that lower degree vertices, and we are presently working to theoretically verify and empirically explore the algorithmic impact of these heavy-tailed degrees.

In [20], the authors collapsed the larger graphs into smaller, more manageable graphs (with vertex count $< 1000$) and matched across these smaller graphs. For any two subjects, they were able to correctly match a significantly higher percentage of the vertices for the two pairs of within–subject graphs than for the four pairs of across–subject graphs. We obtain analogous results by running the LSGM algorithm to match across the larger, less downsampled, graphs. The graphs are created such that the true alignment for any two graphs matches vertices comprised of the same voxels in the $64^3$ voxel brain mask.

In Fig. 8, we highlight our results for a single pair of subjects, and note that analogous results held across the data set. In this example, the LCC of graphs 8 and 29 are of size 21,891 and 22,307 respectively, and the LCC of graph 1 is size 22,734. We match across the intersection of the LCC's for graphs 8 and 29 (same subject, results plotted in Fig. 8) and for graphs 1 and 8 (different subjects, results plotted in Fig. 8). From the SCREE plot, we estimate the optimal embedding dimension to be $d = 30$ in both cases and we cluster using $k$-means, and as noted in Section 3.3, we recluster any overly large clusters—here reclustering any clusters of size $\geqslant 800$—and hence we initially set $k = \lceil n/800 \rceil$. It is clear from Fig. 8 that LSGM correctly matches a significantly larger proportion of vertices for the within–subject connectomes than the across–subject

connectomes. As these connectomes are too large to feasibly run SGM (or any of the bijective matching procedures other than U—which performed very poorly here), we cannot compare the performance of LSGM to the other bijective approaches here.

On the 2x Intel(R) Xeon(R) CPU E5-2660 0 @ 2.20 GHz machine using 12 cores, we display the average runtime (wall time) when matching across connectomes for the four steps of our algorithm in Table 2. Although the projection step takes longer to run than matching in some of the examples, this is an artifact of the full parallelization of the matching step; indeed, the matching step would be computationally unwieldy without parallelizing. We also note that slower matching corresponds to better algorithmic performance. With this in mind, we expect greater improvement from implementing our algorithm on more specialized computational hardware (and parallelizing the SVD calculation for very large graphs) and from employing hot restarts when the algorithm terminates quickly. We emphasize that even very large graphs can be reasonably matched with a simple computing cluster.

It is worth noting that in this example (and across the entire data set), more seeds corresponded to a significantly better matched ratio for both the within-subject and across-subject pairs of graphs. However, for the larger values of $s$ ($s = 1000, 2000, 5000$), we are unable to run the SGM subroutines utilizing the full seeding. Instead, we used the active seed selection algorithm of Section 3.3 to pick an "optimal", computationally feasible set of seeds to use in matching across each cluster. In all cases, our algorithm performs significantly better than chance (chance here being $1/(n-s) = [5.35e-5, 5.65e-5, 5.99e-5, 7.3e-5]$ for the 1–8 pair and $1/(n-s) = [4.87e-5, 5.12e-5, 5.39e-5, 6.43e-5]$ for the 8–29 pair for $s = [0, 1000, 2000, 5000]$).

We also explore the potential for increased performance in LSGM by utilizing different clustering procedures. The brain graphs are very sparse, and there is precedent in the literature that first projecting the latent positions onto the sphere and then clustering the graphs via $k$-means results in better clustering performance in the presence of graph sparsity [28]. We call this variant of $k$-means the *spherical k-means* (*sk*-means) algorithm, and we see that replacing standard $k$-means with *sk*-means significantly increases the performance of the LSGM algorithm. This result reinforces the idea that, in practice, the clustering procedure should be chosen to leverage the signal present in the data.

Our results reconfirm that variability in the estimated connectivity is greater between subjects than within subjects. The estimated connectivity varies due to both noise in the collection of raw scan data and the use of a suite of pre-processing tools used to clean, register and analyze the raw data. As a result, large scale graph matching can serve as another tool to assess the reliability of these methods. Furthermore, this suggests that when registering two scans from the same subject, jointly using geometric properties and connectivity will improve registration accuracy.

We lastly note that within cluster matching using SGM also significantly outperforms the other graph matching algorithms (applied post embedding and clustering) when matching across brain graphs; see Table 3 for the matching accuracy and standard error (over 20 Monte Carlo replicates) for matching the 8–29 within-subject pair in the divide-and-conquer paradigm using $s = 2500$ seeded vertices and embedding the graphs into $d = 30$. We did not run PATH and GLAG here due to scalability concerns. We lastly note that running even the fastest of these algorithms, Umeyama's spectral matching procedure, without first performing the embedding and clustering is prohibitively slow. Indeed, here Umeyama's algorithm has a runtime in excess of 50 h and using over 30 GB of RAM, reinforcing the necessity of the divide-and-conquer step (note that in the *graphm* package Umeyama was downloaded from, the large graph example has 1500 vertices).

## 6. Discussion

Many graph inference tasks rely on being able to efficiently match across graphs. State-of-the-art bijective approximate graph matching algorithms have computational complexity $O(n^3)$–rendering them infeasible (without significant computational resources) for very large graphs. We present the fully parallelizable LSGM approximate graph matching algorithm which, under some mild conditions, has computational complexity $O(n^2 d)$–a marked improvement over $O(n^3)$. We demonstrate, via simulated data examples and a real data example, the effectiveness of our LSGM algorithm in performing seeded graph matching across large graphs, which heretofore were unassailable using existing bijective matching techniques. In

**Table 2**
Runtime for LSGM on the connectome graphs. For each of the four steps of our procedure and each combination of seeds and connectomes, we display the average wall time measured in seconds. The clustering step is the traditional $k$-means, not the *sk*-means modification. Again, matching is the most time intensive step. It is interesting to note that a longer matching runtime corresponds to better algorithmic performance. Note that the graphs are projection into $\mathbb{R}^{30}$.

| Subjects | Seeds | Projection | Procrustes | Clustering | Matching |
|---|---|---|---|---|---|
| *Runtime in seconds for connectome experiment using k-means clustering* | | | | | |
| 01-08 | 200 | 562.82 | 1.57 | 13.49 | 473.46 |
| 01-08 | 1000 | 754.84 | 2.44 | 15.79 | 883.58 |
| 01-08 | 2000 | 862.43 | 2.82 | 15.14 | 1495.26 |
| 01-08 | 5000 | 981.86 | 3.60 | 13.55 | 2698.32 |
| 08-29 | 200 | 777.11 | 1.94 | 18.16 | 569.97 |
| 08-29 | 1000 | 987.08 | 2.80 | 18.47 | 1019.73 |
| 08-29 | 2000 | 1096.84 | 3.26 | 19.29 | 1592.76 |
| 08-29 | 5000 | 890.89 | 2.90 | 15.19 | 2902.19 |

**Table 3**
The matching accuracy and standard error for matching the 8–29 within-subject pair across matching algorithms in the divide-and-conquer paradigm. The max cluster size is set to 800, $s = 2500$, the graphs are embedded into $d = 30$, and the number of Monte Carlo replicates is 20. Note that SGM greatly outperforms the other algorithms.

| | SGM | FAQ | Umeyama | PATH CR |
|---|---|---|---|---|
| *Mean matching accuracy and standard error* | | | | |
| Mean accuracy | 0.0773 | 0.0064 | 0.0034 | 0.0091 |
| Standard Error | 1.72e-03 | 4.25e-04 | 1.12e-04 | 2.91e-04 |

addition, we theoretically justify our divide-and-conquer procedure in the SBM regime by proving that the procedure perfectly matches correlated SBM random graphs under some mild assumptions.

Our algorithm allows for flexibility in the choice of clustering and matching procedure. We focused on $k$-means clustering here due to its ease of implementation and theoretical tractability, but the clustering procedure can (and should!) be chosen to leverage the signal present in the data. The variety of matching procedures implemented point to the need for seeding the rest of the bijective graph matching procedures.

When using the seeds to match, we need to intelligently choose as many seeds as is feasible in the subsequent matching task. We present a procedure for dynamically selecting seeded vertices. Our procedure also provides a heuristic for defining "good" seeded vertices, and we are working on extending this heuristic towards the task of active learning of seeded vertices.

## Acknowledgments

This work is partially supported by a National Security Science and Engineering Faculty Fellowship (NSSEFF), Johns Hopkins University Human Language Technology Center of Excellence (JHU HLT COE), and the XDATA program of the Defense Advanced Research Projects Agency (DARPA) administered through Air Force Research Laboratory contract FA8750-12-2-0303. We also would like to thank William Roncal Gray, R. Jacob Vogelstein and Disa Mhembere for their help with the connectome data and thoughtful discussions and suggestions.

## References

[1] A. Armiti, M. Gertz, Efficient geometric graph matching using vertex embedding, in: Proceedings of the 21st ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems, ACM, 2013, pp. 224–233.
[2] J. Baglama, L. Reichel, Augmented implicitly restarted Lanczos bidiagonalization methods, SIAM J. Sci. Comput. 27 (1) (2005) 19–42.
[3] B. Bahmani, B. Moseley, A. Vattani, R. Kumar, S. Vassilvitskii, Scalable k-means++, Proc. VLDB Endowment 5 (7) (2012) 622–633.
[4] A.C. Berg, T.L. Berg, J. Malik, Shape matching and object recognition using low distortion correspondences, IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2005. CVPR 2005, vol. 1, IEEE, 2005, pp. 26–33.
[5] M.W. Berry, D. Mezher, B. Philippe, A. Sameh, Parallel algorithms for the singular value decomposition, Statistics Textbooks and Monographs 184 (2006) 117.
[6] M. Brand, Fast low-rank modifications of the thin singular value decomposition, Linear Algebra Appl. 415 (1) (2006) 20–30.
[7] T. Caelli, S. Kosinov, An eigenspace projection clustering method for inexact graph matching, IEEE Trans. Pattern Anal. Mach. Intell. 26 (4) (2004) 515–519.
[8] S. Chatterjee, Matrix estimation by universal singular value thresholding, 2012. Available from arXiv:1212.1247.
[9] M. Cho, J. Lee, K.M. Lee, Feature correspondence and deformable object matching via agglomerative correspondence clustering, in: IEEE 12th International Conference on Computer Vision, 2009, IEEE, 2009, pp. 1280–1287.
[10] M. Cho, K.M. Lee, Progressive graph matching: making a move of graphs via probabilistic voting, in: IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2012, IEEE, 2012, pp. 398–405.
[11] D. Conte, P. Foggia, C. Sansone, M. Vento, Thirty years of graph matching in pattern recognition, Int. J. Pattern Recogn. Artif. Intell. 18 (03) (2004) 265–298.
[12] D.L. Davies, D.W. Bouldin, A cluster separation measure, IEEE Trans. Pattern Anal. Mach. Intell. 2 (1979) 224–227.
[13] R.O. Duda, P.E. Hart, D.G. Stork, Pattern Classification, John Wiley & Sons, 2012.
[14] E. Elhamifar, R. Vidal, Sparse subspace clustering: Algorithm, theory, and applications, 2012.
[15] M. Fiori, P. Sprechmann, J. Vogelstein, P. Mus, G. Sapiro, Robust multimodal graph matching: sparse coding meets graph matching, in: Neural Information Processing Systems (NIPS) spotlight, 2013.
[16] D. Fishkind, S. Adali, C. Priebe, Seeded graph matching, 2012. Available from arXiv:1209.0367.
[17] D.E. Fishkind, D. Sussman, M. Tang, J.T. Vogelstein, C.E. Priebe, Consistent adjacency-spectral partitioning for the stochastic block model when the model parameters are unknown, SIAM J. Matrix Anal. Appl. 31 (1) (2013) 23–39.
[18] M. Frank, P. Wolfe, An algorithm for quadratic programming, Nav. Res. Logist. Quart. 3 (1–2) (1956) 95–110.
[19] M.R. Garey, D.S. Johnson, Computers and Intractability, Freeman, New York, 1979.
[20] W.R. Gray, J.A. Bogovic, J.T. Vogelstein, B.A. Landman, J.L. Prince, R.J. Vogelstein, Magnetic resonance connectome automated pipeline: an overview, IEEE Pulse 3 (2) (2012) 42–48.
[21] W.R. Gray et al., Migraine: Mri graph reliability analysis and inference for connectomics, in: GlobalSIP, 2013.
[22] K. Haris, S.N. Efstratiadis, N. Maglaveras, C. Pappas, J. Gourassas, G. Louridas, Model-based morphological segmentation and labeling of coronary angiograms, IEEE Trans. Med. Imaging 18 (10) (1999) 1003–1015.
[23] J.A. Hartigan, M.A. Wong, Algorithm as 136: a k-means clustering algorithm, J. R. Stat. Soc. Ser. C (Appl. Stat.) 28 (1) (1979) 100–108.
[24] P.W. Holland, K.B. Laskey, S. Leinhardt, Stochastic blockmodels: first steps, Social Netw. 5 (2) (1983) 109–137.
[25] A.M. Khan, D.F. Gleich, A. Pothen, M. Halappanavar, A multithreaded algorithm for network alignment via approximate matching, in: International Conference for High Performance Computing, Networking, Storage and Analysis (SC), 2012, IEEE, 2012, pp. 1–11.
[26] D. Knossow, A. Sharma, D. Mateus, R. Horaud, Inexact matching of large and sparse graphs using laplacian eigenvectors, in: Graph-Based Representations in Pattern Recognition, Springer, 2009, pp. 144–153.
[27] V. Lyzinski, D.E. Fishkind, C.E. Priebe, Seeded graph matching for correlated Erdos–Renyi graphs, J. Mach. Learn. Res. 15 (2014).

[28] V. Lyzinski, D. Sussman, M. Tang, A. Athreya, C.E. Priebe, Perfect clustering for stochastic block model graphs via adjacency spectral embedding, 2013. arXiv preprint.
[29] K. Rohe, S. Chatterjee, B. Yu, Spectral clustering and the high-dimensional stochastic blockmodel, Ann. Statist. 39 (4) (2011) 1878–1915.
[30] S. Sahni, T. Gonzalez, P-complete approximation problems, J. ACM 23 (3) (1976) 555–565.
[31] D. Sussman, M. Tang, D.E. Fishkind, C.E. Priebe, A consistent adjacency spectral embedding for stochastic blockmodel graphs, J. Am. Stat. Assoc. 107 (499) (2012) 1119–1128.
[32] D. Sussman, M. Tang, C. Priebe, Consistent latent position estimation and vertex classification for random dot product graphs, 2013.
[33] S. Umeyama, An eigendecomposition approach to weighted graph matching problems, IEEE Trans. Pattern Anal. Mach. Intell. 10 (5) (1988) 695–703.
[34] J. Vogelstein, J. Conroy, V. Lyzinski, L. Podrazik, S. Kratzer, E. Harley, D. Fishkind, R. Vogelstein, C. Priebe, Large (brain) graph matching via fast approximate quadratic programming, Dec. 2011. Available from arXiv:1112.5507.
[35] Y.J. Wang, G.Y. Wong, Stochastic blockmodels for directed graphs, J. Am. Stat. Assoc. 82 (397) (1987) 8–19.
[36] M. Zaslavskiy, F. Bach, J.-P. Vert, Global alignment of protein–protein interaction networks by graph matching methods, Bioinformatics 25 (12) (2009) i259–1267.
[37] M. Zaslavskiy, F. Bach, J.-P. Vert, A path following algorithm for the graph matching problem, IEEE Trans. Pattern Anal. Machine Intell. 31 (12) (2009) 2227–2242.
[38] F. Zhou, F. De la Torre, Factorized graph matching, in: IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2012, IEEE, 2012, pp. 127–134.
[39] M. Zhu, A. Ghodsi, Automatic dimensionality selection from the scree plot via the use of profile likelihood, Comput. Stat. Data Anal. 51 (2) (2006) 918–930.