Aligning Spatially Constrained Graphs

Vikram Ravindra, Huda Nassar, David F. Gleich, Ananth Grama

Abstract—We focus on the problem of aligning graphs that have a spatial basis. In such graphs, which we refer to as *rigid graphs*, nodes have preferred positions relative to their graph neighbors. Rigid graphs can be used to abstract objects in diverse applications such as large biomolecules, where edges corresponding to chemical bonds have preferred lengths, functional connectomes of the human brain, where edges corresponding to co-firing regions of the brain have preferred anatomical distances, and mobile device/ sensor communication logs, where edges corresponding to point-to-point communications across devices have distance constraints. Effective analysis of such graphs must account for edge lengths in addition to topological features. For instance, when identifying conserved patterns through graph alignment, it is important for matched edges to have correlated lengths, in addition to topological similarity.

In this paper, we formulate the problem of *rigid graph alignment* and present a method for solving it. Our formulation of rigid graph alignment simultaneously aligns the topology of the input graphs, as well as the geometric structure represented by the edge lengths, which is solved using a block coordinate descent technique. Using detailed experiments on real and synthetic datasets, we demonstrate a number of important desirable features of our method: (i) it significantly outperforms topological and structural aligners on a wide range of problems; (ii) it scales to problems in important real-world applications; and (iii) it has excellent stability properties, in view of noise and missing data in typical applications.

Index Terms—Graph Alignment, Rigid Body Alignment, Human Brain Connectomics

1 Introduction

Graph databases store data abstracted from diverse systems ranging from social networks to biomolecular interactions. In many such graphs, node positions are constrained in space relative to their topological neighbors. We refer to such graphs as *rigid graphs*. An intuitive example of a rigid graph is an abstraction of a molecule, with nodes corresponding to atoms, and edges to bonds. Bonds between specific atom pairs have well-defined lengths (e.g., a Hydrogen-Hydrogen bond has a preferred length of 74pm, a Carbon-Carbon bond, 154 pm, etc.). Bonds may be stretched or compressed, however, their potential energy increases in the process, and the molecule releases this energy in kinetic form to return to its native structure. Note that the precise interaction models (interaction potentials) are considerably more complex – however, this high-level description is illustrative of our concept.

As another example of rigid graphs, we consider common models for the human brain connectome. The brain connectome is constructed by correlating and thresholding the activity levels of different regions of the brain represented as time series. The resulting connectome is modeled as a rigid graph, with nodes corresponding to regions of the brain and edges to nerve bundles, which have preferred lengths in particular species. Abstractions of functional activity in the brain in the form of a connectome can be used for various purposes. It can be used to characterize subnetworks associated with specific activity or stimulus. It can also be used to understand changes in structure and function through progression of disease. In the event of traumatic brain injuries (and subsequent recovery), it can be used to understand loss of function and recovery.

There have been significant efforts aimed at formulating and solving a variety of analytics problems on graphs. Among these, solutions to computation of modularity [1], [2], centrality [3], [4], alignment [5], and clustering [6], [7] have received significant research attention. Solutions to these problems are typically topological in nature – their cost functions are based on the presence or absence of edges, and typically do not simultaneously consider the preferred relative positions of nodes. For example, graph clustering aims to identify densely connected subgraphs in a larger graph, alignment aims to identify conserved subgraphs with high edge overlap, and centrality aims to identify edges that are critical (e.g., carrying large numbers of shortest paths). When formulating these problems on rigid graphs, one must generalize associated cost functions to account for preferred edge lengths, and derive techniques for optimizing these cost functions.

Rigid graph alignment aims to find correspondences between nodes in two input graphs such that edge mappings induced by these correspondences are highly correlated in terms of their lengths. Conventional formulations of related problems focus on structural alignments, which identify structural transformations such as translation, rotation, and dilation, to maximize structural overlap. Topological graph alignment techniques aim to identify node correspondences with the objective of maximizing edge overlaps (without considering edge lengths). *Rigid graph alignment integrates both structural and topological alignment into a single objective function.* We formalize this objective function and present a novel method for optimizing it. We show that our solution outperforms state of the art structural and topological alignment techniques using both real and synthetic datasets

- *V. Ravindra, D. F. Gleich and A. Grama are with the Department of Computer Science, Purdue University, West Lafayette, IN, 47907. E-mail: ravindrv@purdue.edu*
- *H. Nassar is with Relational AI, this work was done while at Purdue University and Stanford University*

Manuscript received Month DD, YYYY; revised Month DD, YYYY .

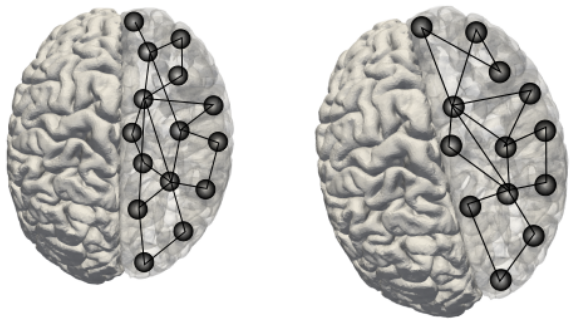


Fig. 1: Illustration of two brains with functional network shown in one hemisphere. The two networks are similar in terms of node placement, and pattern of edges, but are not structurally identical – the second brain is slightly larger and mis-oriented (in imaging), with respect to the first brain.

based on application metrics as well as network-based measures.

We motivate our rigid graph alignment problem formulation using an important application in the analyses of 3D brain images. It has been hypothesized that cognitive processes manifested in MRI images are unique to individuals [8]. Stated differently, it is possible to uniquely identify an individual using their resting state MRI from among a given set of MRI images. Such analyses can be performed by matching structural features, by registering to a common coordinate, or by extracting networks of functional activity and aligning these networks (Figure 1). Our rigid graph alignment model incorporates both anatomical (structural) and functional (connectomic) information by iteratively improving on the quality of network match using a (topological) network aligner, and quality of structural match using rigid body transformations (Figure 2). This ensures that we: (i) do not suppress signals unique to an individual; and (ii) obtain anatomically consistent alignments. Owing to structural and functional uniqueness of individuals, we hypothesize and validate that the quality of alignment between two networks of the same individual (taken across two distinct imaging sessions) should be higher than that between two different individuals, and that "rigid graph alignment" is a critical methodological basis for such alignments.

Summary of Contributions. The paper makes the following technical contributions: (i) it mathematically formulates the rigid graph alignment problem and puts the problem in the context of existing graph alignment and structural alignment problem formulations; (ii) it proposes a solution to the rigid graph alignment problem based on block coordinate descent; (iii) it comprehensively evaluates the performance of the proposed solution on real-world and synthetic graphs and demonstrates significant performance improvement over state-of-the-art aligners; (iv) it characterizes the robustness of the solution and the impact of various problem parameters through a comprehensive study on synthetic graphs and (v) demonstrates the feasibility of using Rigid Graph Alignment to register pairs of functional MRIs, without requiring normalization to a standard coordinate system. This paper significantly builds on the preliminary results presented by us [9].

2 Related Literature

Network Alignment Early formulations and applications of network alignment aimed to find matches in small chemical networks [10]. This was largely limited to exact graph matching (or subgraph isomorphism). Subsequent work by Zelinka [11], [12] is credited with the formalization of the idea of distances between pairs of graphs. These efforts primarily worked with graph isomorphisms as well. The first methods to allow for approximate matches used the Hungarian Method to find matches on weighted adjacency matrices [13]. Current scalable network alignment techniques are typically approximate – they allow for unmatched edges, multiple matches, graphs of different sizes, etc. Modern alignment techniques broadly fall in two categories – global and local. Global aligners optimize over the entire graph, and they reward decisions that benefit a global objective. IsoRank [5], [14] is a popular global network alignment method. Matches in IsoRank are driven by similarity of neighbourhoods of pairs of nodes in two graphs. There are many variants of IsoRank, such as the approximation by Kollias et al. [15] and adaptation of IsoRank to multiple graph alignment [16]. A linear relaxation of the QP IsoRank formulation is described by Klau et al. [17]. The second class of aligners are local aligners, such as AlignNemo [18], which find sub-graphs that match well. The optimization criteria rewards locally accurate matches, while not taking a global view of optimality. Using a local network alignment approach, Lässig et al. [19] successfully align gene regulation networks in *E. coli*. In addition to pair-wise alignment techniques, the problem of multiple network alignment has also been investigated. Methods for multiple network alignment include IsoRankN, along with other more recent techniques [20], [21], [22].

Network Alignment Algorithms. While run-time is a primary consideration for network aligners, the ability to align large graphs also requires memory efficiency. Methods such as GRAAL [23] and GHOST [24] are memory efficient, but require cubic run-time. Mohammadi et al. [25] show that a low-rank approximation of the similarity matrix in the IsoRank formulation can reduce memory requirement substantially. In EigenAlign, Feizi et al. [26] utilize the dominant eigen-vector of product-graph matrix to find a matching between the vertices of the graphs by solving a maximum-weight bipartite matching problem on a dense bipartite graph.

A natural approach to limit the search-space of potential matches involves the use of domain knowledge, or a prior that informs the aligner of possible matches that are meaningful in context to the application. Examples of these approaches include the previously mentioned IsoRank. A message passing algorithm *netalignbp* by Bayati et al. [27] has been shown to be efficient in matching large, sparse graphs. In this paper, we use graph alignment techniques that incorporate prior knowledge. While we will use *netalignbp* in our results, we note that our meta-algorithm is agnostic to the network aligner, as long as it allows for the incorporation of a prior.

Heimann et al. [28] introduce a greedy approach to identify node matchings by aligning their latent feature representations. In the first step, graph attributes such as

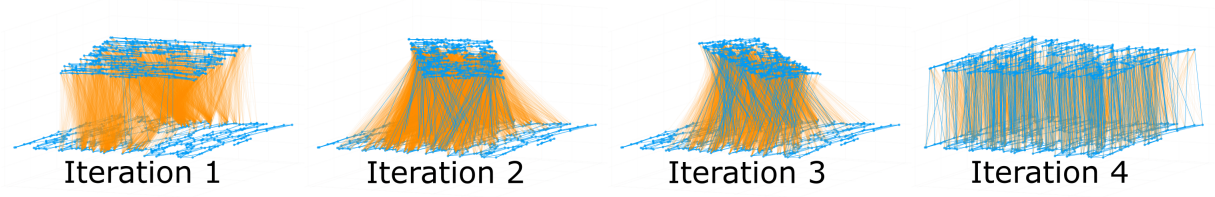


Fig. 2: **A demonstrative example showing iterative improvement in alignment by Rigid Graph Alignment.** The two blue planes are the two graphs that we aim to align. The vertical lines represent to predicted node correspondences – orange lines represent wrong predictions and blue lines represent correct predictions. As the top graph is rotated to structurally align better with the bottom graph, quality of network alignment also increases significantly.

degree and neighbors are used to establish the identity of nodes. Following this, a similarity matrix is computed for pairs of nodes across the two graphs on the basis of the previously established attributes. Then, nodes are matched in a greedy fashion. **In a method named CONE-Align, Chen et al. [29] embed nodes of both graphs using a GNN. Then, they align the embeddings using Wasserstein Procrustes. Finally, they use euclidean distances to match pairs of nodes.**

The problem of alignment of social networks with location data was explored by Riederer et al. [30]. While these methods use node and/ or edge attributes to construct similarity measures between nodes, they are not well suited for the applications that motivate our approach. We explicitly leverage positions of nodes as rich sources of information, and their geometric transformations to improve the quality of alignment.

Network Alignment in Databases. The problem of graph alignment is also commonly applied to databases. Melnik et al. [31] introduce a meta algorithm to match different data structures, or *models*. These models are abstracted as graphs, and network alignment algorithms are used to find matching nodes. In this application, network alignment is used to manipulate and maintain schemas and match results. However, human intervention is required to validate the matches. In [32], the problem of alignment between versions of evolving RDF databases is explored. In particular, the paper proposes a method to align nodes of two consecutive versions of an RDF database using a bisimulation-based approach. The approach is flexible to accommodate nodes without labels (called "blank" nodes). In another network alignment algorithm for databases, Zhang et al. [33] define constraints for *consistency* and align nodes accordingly.

Network Alignment in Imaging. Graph matching algorithms have also been applied to image alignment. Wiskott et al. [34] propose a face recognition algorithm, wherein graph representations of faces are obtained by using a Gabor Wavelet transform. Following this, a graph matching algorithm is used to correctly match faces. A number of variants of such *elastic graph alignments* have since been proposed. Zhou et al. [35] propose a method to align landmarks on images of faces. The first step in this method uses affine transformation to learn shape constraints, which are represented as graphs. This is followed by network alignment to find similarities across images. Our approach differs from these techniques in that we do not assume the presence of a fixed set of landmarks. Indeed, a small number of useful landmarks would simplify the problem to requiring only one iteration of our algorithm. We work with noisy graphs of different sizes, which requires iterative procedures to continuously improve structural and graph alignments. Furthermore, none of these prior methods have been shown to scale to the size of graphs we target in our work.

Structural Alignment The formulation of the graph matching problem by [13] can be viewed as a variant of the Procrustes problem, namely the two-sided Procrustes Problem [36], where approximate graph matching is made possible in cases where vertices of the two graphs can be grouped into parcels or clusters. The problem of structural alignment itself is often solved using the *Iterative Closest Point* (ICP) method of Besl et al. [37]. At a high level, the first step of ICP selects source-points from both bodies. ICP then matches the points and computes weights corresponding to the matches appropriately. Points without a close match are discarded, and an error minimizing transformation is computed between the pair of point-clouds. In every iteration, the transformation aims to improve the agreement in the "shape" of the two point-clouds. Our approach to Rigid Graph Alignment loosely follows the same formulation. In our case, the transformation is a rigid body transformation using Kabsch's algorithm. However, the important distinction is that ICP does not view the two sets of source-points as graphs, and therefore cannot incorporate any connectivity information.

3 Problem Formulation

We motivate our problem of rigid graph alignment by first discussing existing formulations of graph and structural alignment. We then contrast our problem with these formulations, and formalize the rigid graph alignment problem.

Formulation of the Network Alignment Problem. The literature on network alignment is vast, with a large number of formulations and associated methods. We focus on the quadratic programming formulation of the network alignment objective [27]. Given two graphs $A = G(V_A, E_A)$ and $B = G(V_B, E_B)$, the goal of network alignment is to find a mapping between V_A and V_B. We assume here that $|V_A| = |V_B| = n$, though this is not required in practice since we can always add *empty nodes* with *no edges* to the graph without changing the objective functions. We denote $\mathbf{L} \in \mathbb{R}^{n \times n}$ to be the matrix that encodes prior knowledge on likelihood of alignment. If $\mathbf{L}_{ij} > 0$, prior knowledge indicates that node i in V_A potentially aligns with node j

in V_B. The prior typically comes from domain knowledge, and pushes the solution towards likely mappings, while simplifying the problem of network alignment.

The aim of the alignment problem is to find a matching $\mathcal{M}$ from V_A to V_B using only the prior matrix $\mathbf{L}$ and adjacency matrices of the two graphs. We define $\mathbf{X} \in \mathbb{R}^{n \times n}$ to be a matrix of the same dimensions as L to encode matching $\mathcal{M}$:

$$\mathbf{X}_{ij} = \begin{cases} 1 & \text{if } i \in V_A \text{ is matched with } j \in V_B \text{ under } \mathcal{M} \\ 0 & \text{otherwise} \end{cases}$$

Under such a mapping, we say that an edge $(u, v) \in E_A$ *overlaps* with an edge $(u', v') \in E_B$ if u matches u' and v matches v' under the match $\mathcal{M}$.

Using this formulation, the goal of graph alignment is to construct matching $\mathcal{M}$ that maximizes a linear combination of edge *overlap* and matching *weight*. We can search over matchings by looking for $\mathbf{X}$,

$$\mathbf{X}_{ij} \in [0,1], \quad \forall i \in [|V_A|], \quad \textstyle\sum_j \mathbf{X}_{ij} \leq 1, \tag{1}$$

$$\forall j \in [|V_B|], \quad \textstyle\sum_i \mathbf{X}_{ij} \leq 1 \tag{2}$$

then

$$\text{overlap of matching } \mathcal{M} = A \bullet \mathbf{X} B \mathbf{X}^T = \textstyle\sum_{ij} A_{ij} (\mathbf{X} B \mathbf{X}^T)_{ij} \tag{3}$$

Here, we use the adjacency matrices A and B for the graphs and $\bullet$ to represent a matrix inner product ($A \bullet B = \sum_{ij} A_{ij} B_{ij}$. The overall objective is given by:

$$\begin{aligned} \max_{\mathbf{X}} \quad & \alpha \mathbf{L} \bullet \mathbf{X} + \beta A \bullet \mathbf{X} B \mathbf{X}^T \\ \text{s.t.} \quad & \textstyle\sum_i \mathbf{X}_{ij} \leq 1 \quad \forall j = 1 \ldots |V_B|, \\ & \textstyle\sum_j \mathbf{X}_{ij} \leq 1 \quad \forall i = 1 \ldots |V_A|, \quad \mathbf{X}_{ij} \in \{0,1\} \end{aligned} \tag{4}$$

The first term of the objective, i.e., $\mathbf{L} \bullet \mathbf{X}$ incentivizes the non-zeros of $\mathbf{X}$ (the matches) to overlap with the positive entries of $\mathbf{L}$, thereby driving the solution towards a matching consistent with prior information. Parameters, α, β are non-negative constants that we can use as parameters to balance the relative importance of edge overlap and matching weights.

Formulation of structural alignment. The problem of structural alignment between two bodies aims to find a transformation that increases the similarity in their "shapes". In our work, we focus on rigid body transformations between two sets of points, wherein the relative positions of the points are preserved after the transformation. The more general version of the problem, popularly known as the Orthogonal Procrustes Problem was first solved by Schönemann [38]. It aims to find an *orthogonal* transformation that reduces the distance between two matrices in the Frobenius norm. Formally, for any two matrices Y and Z, the problem minimizes:

$$\begin{aligned} \min_{\mathbf{\Omega}} \quad & ||Y - Z\mathbf{\Omega}||_F^2 \\ \text{s.t.} \quad & \mathbf{\Omega}^T \mathbf{\Omega} = \mathbf{I} \end{aligned} \tag{5}$$

In our setting, the matrices Y and Z correspond to the physical positions of the graph nodes (or node *coordinates*) in d dimensions; $C_A, C_B \in \mathbb{R}^{n \times d}$. Our objective is to construct the rigid body transformation between the sets of matched nodes of the two graphs. In other words, we aim to find rotation matrix $\hat{\mathbf{R}} \in \mathbb{R}^{d \times d}$ and translation vector $\hat{t} \in \mathbb{R}^{1 \times d}$ such that:

$$C_A = C_B \hat{\mathbf{R}} + \mathbb{1}\hat{t} \tag{6}$$

where, $\mathbf{R}$ is an orthogonal matrix (i.e. $\hat{\mathbf{R}}^T \hat{\mathbf{R}} = I$) and $\mathbb{1}$ is the vector of ones of length n. As per convention, we pad C_A and C_B by a vector of ones to create the "homogeneous" coordinate system (for convenience in notation, we do not explicitly show this detail). This allows us to combine rotation matrix and the translation vector into one matrix $\mathbf{\Omega}$ as:

$$\mathbf{\Omega} = \begin{bmatrix} \hat{\mathbf{R}} & \mathbf{0} \\ \hat{t} & 1 \end{bmatrix} \tag{7}$$

We can now recast the problem as:

$$\min_{\mathbf{\Omega}} \quad ||C_A - C_B \mathbf{\Omega}||_F^2 \tag{8}$$

The first solution to this problem was proposed by Kabsch [39]. We use a related SVD-based method due to Sabata et al. [40], which is empirically stable, as shown by Eggert et al. [41]. First, we center both sets of coordinates.

$$\mu_A = \frac{1}{n} \sum_{i=1}^{n} C_{A_i} \qquad \overline{C}_A = C_A - \mathbb{1}\mu_A \tag{9}$$

$$\mu_B = \frac{1}{n} \sum_{i=1}^{n} C_{B_i} \qquad \overline{C}_B = C_B - \mathbb{1}\mu_B \tag{10}$$

Here, C_{A_i} and C_{B_i} refer to the coordinates of the i-th nodes of graphs A and B. We have, $\mu_A, \mu_B \in \mathbb{R}^{1 \times d}$. Define $\mathbf{H} = \overline{C}_A^T \overline{C}_B$, then the estimated rotation matrix $\hat{\mathbf{R}}$ is given by:

$$\hat{\mathbf{R}} = \mathbf{V}\mathbf{U}^T, \tag{11}$$

where $\mathbf{V}$ and $\mathbf{U}$ are orthonormal matrices that are obtained from SVD of $\mathbf{H}$.

The optimal translation $\hat{t}$ is given by:

$$\hat{t} = \mu_A - \mu_B \hat{\mathbf{R}} \tag{12}$$

When $det(\hat{\mathbf{R}}) = +1$, this process works well. However, if the graph is planar or with large amounts of noise, $det(\hat{\mathbf{R}})$ can become -1. In this case, the rotation matrix $\hat{\mathbf{R}}$ becomes:

$$\hat{\mathbf{R}} = \mathbf{V} \begin{bmatrix} 1 & & \\ & 1 & \\ & & \det(\mathbf{V}\mathbf{U}^T) \end{bmatrix} \mathbf{U}^T$$

We can estimate scaling between the two bodies represented by the graphs using more generalized formulations of the Procrustes Problem. Instead, we make a simplifying assumptions that both coordinate systems are drawn from the same physical units. This assumption also allows us to implicitly bypass issues that arise from nodes being placed in different coordinate systems.

3.1 Rigid Graph Alignment.

We now define our problem of rigid graph alignment. Given two graphs $A = G(V_A, E_A, C_A)$ and $B = G(V_B, E_B, C_B)$, our goal is to find a matching $\mathcal{M}$ between the vertices V_A and V_B, such that the relative positions of the nodes (implicitly coded by the co-ordinates $C_A, C_B \in \mathbb{R}^{n\times d}$) is preserved. Combining the objective functions of network alignment and rigid body alignment, as described in Equations 4 and 8, our objective function for *rigid graph alignment* can be written as:

$$\begin{aligned}\mathbf{F} = \max_{\mathbf{X},\,\mathbf{\Omega}} \quad & \alpha\mathbf{L}\bullet\mathbf{X} + \beta A\bullet\mathbf{X}B\mathbf{X}^T - \gamma||C_A - \mathbf{X}C_B\mathbf{\Omega}||_F^2 \\ \text{s.t.} \quad & \textstyle\sum_i \mathbf{X}_{ij} \le 1 \quad \forall j = 1\ldots|V_B|, \\ & \textstyle\sum_j \mathbf{X}_{ij} \le 1 \quad \forall i = 1\ldots|V_A|, \quad \mathbf{X}_{ij}\in\{0,1\}\end{aligned} \tag{13}$$

As before, $\mathbf{L}\bullet\mathbf{X}$ quantifies the consistency between the prior $\mathbf{L}$ and the mapping of vertices across the two graphs $\mathbf{X}$. The second term, $A\bullet\mathbf{X}B\mathbf{X}^T$, denotes the number of edges that are aligned between the two graphs, after the nodes of B are permuted by the matching $\mathbf{X}$. The final term, $||C_A - \mathbf{X}C_B\mathbf{\Omega}||_F^2$ incorporates structural alignment. In this formulation, $\mathbf{X}C_B$ denotes the conformably permuted set of coordinates of graph B after alignment.

Weights α, β and γ are parameters for the user to adjust the relative importance of the prior, graph matching, and structural alignment. With $\alpha = 1, \beta = 0, \gamma = 0$, the problem reduces to the maximum weight matching problem, where the aligner is driven solely by prior knowledge and not connectivity information. When $\alpha = 0, \beta = 1, \gamma = 0$, it is the solution to the problem of maximizing edge overlap, and when $\alpha = 0, \beta = 0, \gamma = 1$, the problem is simply rigid body transformation.

As discussed before, the residual error in the third term of Equation 13 can be minimized using the generalized Procrustes method, once we have the set of matching vertices. This is provided by the network aligner. For ease of analysis, we view $\mathbf{X}$ as a permutation matrix (which makes $\mathbf{X}$ orthogonal, or $\mathbf{X}^T\mathbf{X} = \mathbf{I}$). Then, the error term corresponding to structural alignment can be expressed as:

$$\begin{aligned}&\|C_A - \mathbf{X}C_B\mathbf{\Omega}\|_F^2 \qquad (14)\\ &= \text{tr}[(C_A - \mathbf{X}C_B\mathbf{\Omega})^T(C_A - \mathbf{X}C_B\mathbf{\Omega})] \\ &= C_A\bullet C_A + \mathbf{X}C_B\mathbf{\Omega}\bullet\mathbf{X}C_B\mathbf{\Omega} - 2C_A\bullet\mathbf{X}C_B\mathbf{\Omega} \\ &= C_A\bullet C_A + C_B\mathbf{\Omega}\bullet C_B\mathbf{\Omega} - 2C_A\mathbf{\Omega}^TC_B^T\bullet\mathbf{X} \qquad (15)\end{aligned}$$

For a given transformation matrix $\mathbf{\Omega}$, using Equation 15, we can rewrite the objective function as

$$\begin{aligned}\mathbf{F} = \max_{\mathbf{X},\,\mathbf{\Omega}} \quad & \overbrace{\alpha\mathbf{L}\bullet\mathbf{X} + \beta A\bullet\mathbf{X}B\mathbf{X}^T}^{\text{Network Alignment}} + \gamma\underbrace{C_A\mathbf{\Omega}^TC_B^T\bullet\mathbf{X}}_{\text{Structural Alignment}} \\ \text{s.t.} \quad & \textstyle\sum_i \mathbf{X}_{ij} = 1 \quad \forall j = 1\ldots|V_B|, \quad \mathbf{\Omega}^T\mathbf{\Omega} = I, \\ & \textstyle\sum_j \mathbf{X}_{ij} = 1 \quad \forall i = 1\ldots|V_A|, \quad \mathbf{X}_{ij}\in\{0,1\}\end{aligned} \tag{16}$$

This re-statement of our objective function suggests that the search for the optimal pair of $\mathbf{X}$ and $\mathbf{\Omega}$ would maximize network alignment and structural alignment, while conforming to prior information. Furthermore, the structural alignment term rewards similarity between the two ordered sets of coordinates. Premised on the key observation that the two objectives depend on, and reinforce each other, we propose an algorithm that alternately optimizes the two terms.

$$\mathbf{X}^{(t)} = \underset{\mathbf{X}}{\text{argmax}}\,\mathbf{F}(\mathbf{X}, \mathbf{\Omega}^{(t)}) \tag{17}$$

$$\mathbf{\Omega}^{(t+1)} = \underset{\mathbf{\Omega}}{\text{argmax}}\,\mathbf{F}(\mathbf{X}^{(t)}, \mathbf{\Omega}) \tag{18}$$

This formulation motivates our rigid graph matching algorithm.

3.2 Rigid Graph Matching Algorithm

We split the task of rigid graph matching into two stages wherein we alternately optimize for network and structural alignment.

With $\mathbf{\Omega}$ fixed, then the problem reduces to the following network alignment problem:

$$\begin{aligned}\mathbf{F} = \max_{\mathbf{X}} \quad & \overbrace{(\alpha\mathbf{L} + \gamma C_A\mathbf{\Omega}^TC_B^T)\bullet\mathbf{X} + \beta A\bullet\mathbf{X}B\mathbf{X}^T}^{\text{Network Alignment}} \\ \text{s.t.} \quad & \textstyle\sum_i \mathbf{X}_{ij} = 1 \quad \forall j = 1\ldots|V_B|,, \\ & \textstyle\sum_j \mathbf{X}_{ij} = 1 \quad \forall i = 1\ldots|V_A|, \quad \mathbf{X}_{ij}\in\{0,1\}.\end{aligned} \tag{19}$$

Here, the matrix $\mathbf{P} = \alpha\mathbf{L} + \gamma C_A\mathbf{\Omega}^TC_B^T$ serves as the *prior* information on how likely pairs of nodes are to align. However, $C_A\mathbf{\Omega}^TC_B^T$ is a dense matrix; this adversely affects the runtime of network aligners. We avoid this by limiting the number of non-zeros using suitable distance or size of neighborhood constraints, and directly use the intuition that *close nodes under the transformation* $\mathbf{\Omega}$ will be more likely to align. Towards that end, let $\tilde{C}_B = C_B\mathbf{\Omega}^{(t)}$ be the current transformation of the coordinates. Then, we consider the following choices for $\mathbf{P}$:

First,

$$\mathbf{P}_{i,j} = \begin{cases}\gamma + \alpha\mathbf{L}_{i,j} & \left\|C_{A_i} - \tilde{C}_{B_j}\right\|_2^2 \le \varepsilon \\ \alpha\mathbf{L}_{i,j} & \text{otherwise}\end{cases} \tag{20}$$

where ε denotes a small neighborhood. Alternatively, we can set the ij-th entry of $\mathbf{L}$ to be inversely proportional to the distance between the i-th node of A and the j-th node of B in a co-registered space.

$$\mathbf{P}_{i,j} = \begin{cases}\alpha\mathbf{L}_{i,j} + \gamma\text{exp}(-\left\|C_{A_i} - \tilde{C}_{B_j}\right\|_2^2) & \left\|C_{A_i} - \tilde{C}_{B_j}\right\|_2^2 \le \varepsilon \\ \alpha\mathbf{L}_{i,j} & \text{otherwise}\end{cases} \tag{21}$$

Yet another option is to match node i of graph A with one of its k-nearest neighbours. The distance, d_i^k, of i and the k-th closest neighbor is given by:

$$\mathbf{P}_{i,j} = \begin{cases}\alpha\mathbf{L}_{i,j} + \gamma\text{exp}(-\left\|C_{A_i} - \tilde{C}_{B_j}\right\|_2^2) & \left\|C_{A_i} - \tilde{C}_{B_j}\right\|_2^2 \le d_i^k \\ \alpha\mathbf{L}_{i,j} & \text{otherwise}\end{cases} \tag{22}$$

To perform this first step, we devise a routine – *get_prior* to implement these choices (a user would select among them); alternatively, one could provide their own procedure for this purpose. The second step requires a structural

aligner, which updates $\mathbf{\Omega}$ based on the current estimate of $\mathbf{X}$. This is:

$$\mathbf{F} = \max_{\mathbf{\Omega}} \overbrace{\alpha \mathbf{L} \bullet \mathbf{X} + \beta A \bullet \mathbf{X} B \mathbf{X}^T}^{\text{Constant in } \mathbf{\Omega}} + \gamma \underbrace{C_A \mathbf{\Omega}^T C_B^T \bullet \mathbf{X}}_{\text{Structural Alignment}} \quad \text{s.t.} \quad \mathbf{\Omega}^T \mathbf{\Omega} = I \tag{23}$$

This is again equivalent to a standard structural alignment problem. Thus, we use an SVD-based method due to Sabata et al. [40], as discussed in Section 3, to align C_A to $\mathbf{X}^{(t)} C_B$ (i.e., we permute C_B based on $\mathbf{X}$ and then seek the transformation). Stated alternately, in the second step, we align the spatial positions of nodes after re-ordering them according to the current estimate of matches computed by the network aligner. This requires an initial alignment to begin, which we discuss at the end of the section.

To summarize, in step (i), we maximize the network alignment term $\alpha \mathbf{L} \bullet \mathbf{X} + \beta A \bullet \mathbf{X} B \mathbf{X}^T$ with the term $\gamma ||C_A - \mathbf{X} C_B \mathbf{\Omega}||_F^2$ approximately fixed, whereas in step (ii), we maximize $-\gamma ||C_A - \mathbf{X} C_B \mathbf{\Omega}||_F^2$ with the network alignment term fixed.

The resulting *rigid graph alignment* procedure is given in Algorithm 1. For this algorithm, we assume that there is no *a-priori* matrix $\mathbf{L}$ given, so the only input to the network alignment scheme is based on the construction from (22) in the get_prior routine.

Algorithm 1 Rigid Graph Alignment

1: **Input:** Graphs $A(V_A, E_A)$ and $B(V_B, E_B)$, Coordinates C_A and C_B, α, β, γ
2: **Output:** Aligned graphs A and B
3: **repeat**
4: $\mathbf{P} =$ get_prior(C_A, C_B) (see (22))
5: $\mathbf{X} =$ align$(A, B, \mathbf{P})$
6: $B = \mathbf{X} B \mathbf{X}^T$
7: $C_B = \mathbf{X} C_B$
8: $\mathbf{\Omega} =$ transform_coordinates(C_A, C_B) (note that C_B is updated below).
9: $C_B = C_B \mathbf{\Omega}$
10: **until** converged

This process is continued until convergence is achieved. The metric for deciding convergence depends on the application itself. Natural measures include edge/ node overlap, or residual error in the structural alignment. In both synthetic (Section 4.1) and real-world (Section 4.3) rigid graphs, we reach convergence (in terms of edge overlap) within few iterations. The strength of the meta-algorithm proposed here is that it works with any graph aligner that admits a prior, and with any strutural aligner. In our experiments, we present results using *netalignmbp* [27] as the graph aligner and an SVD-based structural aligner [40].

Bootstrap Procedure. In the first iteration, we have no prior knowledge on correspondence between nodes, nor do we have an estimate of the transformation required to map the physical position of nodes. To overcome this, we devise a bootstrapping procedure in which we populate the prior matrix on the basis of similarity of distance profiles. Specifically, we compute the pairwise distance between all pairs of nodes in each of the two graphs. Then, we draw histograms of distances for each of the nodes. The initial estimate of potential matches is on the basis of similarity of these distance profiles. We quantify the similarity using Pearson Correlation. We enforce sparsity by thresholding the correlations, so as to retain only significant entries. Pearson Correlation, and therefore the distance profiles that we construct, are invariant to rotations, translations and scaling.

3.3 An iterative illustration

In Figure 2, we show an illustrative example of the working of Rigid Graph Alignment on a synthetic graph. The figure shows two graphs, represented by the two blue planes. The vertical lines show node correspondence, as predicted by the network aligner. The orange lines represent pairs of nodes wrongly matched, and blue lines represent pairs of nodes correctly matched. An iterative improvement in quality of network alignment is observed, as evidenced by the increase in blue vertical lines. At the same time, the top graph is progressively rotated to conform with the lower graph, thereby increasing structural match.

4 Experimental Evaluation

We present results from two sets of experiments – the first set of experiments is performed on synthetic datasets. These experiments are used to assess the stability of our proposed method, comparison to state of the art techniques, and demonstrating runtime characteristics and impact of various parameters. The second set of experiments is performed on a real dataset from the Human Connectome Project. These experiments demonstrate the superiority of our method on this important problem, while validating a number of application hypotheses.

4.1 Experiments on Synthetic Datasets

We test the performance of our formulation and method on synthetically generated structural adaptations of preferential attachment and Erdős-Rényi ($G(n,p)$) graphs to assess the impact of various problem parameters.

Generation of Synthetic Graphs. The process of generating synthetic rigid graphs involves two steps – (i) spatial location of nodes; and (ii) assignment of edges between them. To locate nodes, we first create an evenly spaced d-dimensional grid (we show results for $d = 3$), with g (= 100) points along each dimension. A grid point is assigned a network node on the basis of the outcome of an independent, (biased) coin toss. If we don't have the required number of nodes after sampling all grid points, we randomly sample from unassigned grid points. This process is continued until we have the required number of nodes. Note that not all grid points have network nodes assigned to them.

To assign edges, we consider two strategies: Preferential Attachment [42] and Erdős-Rényi ($G(n,p)$) [43]. In the Preferential Attachment or the Barabási-Albert model, we start with an initial graph consisting only of node n_0. Then, an incoming node attaches itself to a subset of existing nodes with probability proportional to their degrees, i.e., $p_i = k_i / \sum_j k_j$, where p_i is the probability of attaching

to node i, which has degree k_i. In the $G(n,p)$ model, an edge exists between any pair of nodes i and j with a probability p, independent of other edges. **We note that our generation approach is similar to the generation process for random-geometric graphs [44], [45]. To generate large graphs, distributed approaches such as the one proposed by Funke et al. [46] can be used.**

Perturbation Schemes. For network alignment, we generate pairs of networks. The first graph is generated in accordance with the aforementioned procedure. The second network is created by adding noise both with respect to the spatial location of the nodes, as well connectivity profiles of the graphs. To add spatial noise, the coordinates are rotated along each axis by $\theta°$. Then, each coordinate is translated by a random vector $t \in \mathbb{R}^d$. Following this, the coordinates of each node are independently perturbed by $C_n \in \mathbb{R}^{n\times d}$. Finally, edges are deleted independently with a probability p_d and added with a probability p_a. This process yields two networks with parametrizable differences, thereby allowing us to characterize the behaviour of both network alignment and structural alignment.

We quantify performance in terms of correctly identified edges and node overlap (or node correctness) – the fraction of correctly paired nodes, which corresponds to the precision for our problem. Note that these measures are feasible for synthetic experiments, where the correct solution is known a-priori.

4.1.1 Impact of perturbation to the adjacency matrix

In this experiment, we add and delete edges, while holding the relative position of nodes. We allow the entire graph to be rotated along all axes. We set $p_d = p_a$ for these experiments. We repeat the experiment for $n = \{500, 1000\}$, $d = 3$, $\theta = \{1°, 5°, 30°, 60°, 90°, 180°\}$ for both preferential attachment and $G(n,p)$ graphs.

Figures 3 and 4 show the results of the experiment for $n = 1000$, $d = 3$, $\theta = 60°$ for preferential attachment graphs. The performance of both rigid graph alignment and regular network alignment with respect to edge overlaps is comparable for low noise; however, rigid graph alignment is robust to higher levels of noise (Figure 3). In fact, edge overlap in rigid graph alignment closely follows the pattern of true edge alignment. The drop in edge overlap for regular graph alignment corresponds to a drop in node alignment, as seen in Figure 4. The node overlap of rigid graph alignment is, on average, $97.67 \pm 1.21\%$, whereas the same for regular network alignment is $65.85 \pm 3.90\%$, across different rotations. In the absence of node noise, the spatial relationships between nodes are maintained, thereby leaving edge lengths of overlapped edges unchanged. This means that a correct rigid body transformation leaves the two graphs structurally identical, which explains the high values of node overlap when rigidity of edges is considered.

The results of the same experiment on $G(n,p)$ graphs are presented in Figures S1 and S2. Similar patterns of behavior are observed in the case of rigid graph alignment. However, the absence of patterns in edges in $G(n,p)$, coupled with the fact that regular graph aligners do not leverage structural data, causes the edge and node overlap to be substantially lower.

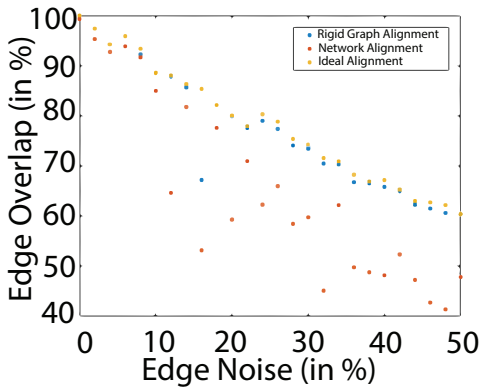


Fig. 3: **Change in edge overlap with increase in edge noise in a preferential attachment network, while node noise is fixed to zero.** It can be seen that the overlap in rigid graph alignment algorithm closely follows the true edge overlap, whereas edge overlap in regular network alignment methods is considerably lower at higher noise levels.

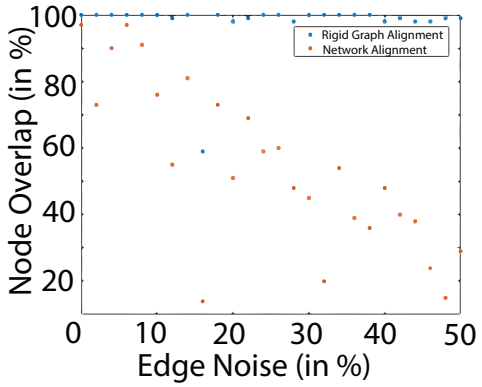


Fig. 4: **Behavior of node overlap with increase in edge noise in a preferential attachment network, while node noise is fixed to zero.** This graph shows that edge noise has no impact on the correct identification of nodes for rigid graph alignment. However, for regular network alignment, node overlap decreases linearly with increasing edge noise

4.1.2 Impact of perturbation by node movement

In this experiment, we perturb the physical position of nodes, while fixing the edge noise to zero, i.e., $p_a = p_d = 0$. The nodes are perturbed along each coordinate by independently sampled, scaled random matrix C_n. The scaling is done so as to represent the perturbation as a proportion of the physical size of the network. The experiment was conducted for $n = \{500, 1000\}$, $d = 3$, $\theta = \{1°, 5°, 30°, 60°, 90°, 180°\}$ for both preferential attachment and $G(n,p)$ graphs.

The results for $n = 1000$, $d = 3$, $\theta = 60°$ in preferential attachment graphs are shown in Figures 5 and 6. In rigid graph alignment, the node overlap is $80.35 \pm 1.21\%$, whereas the same for regular network alignment is $35.85 \pm 3.90\%$. The decrease in node overlap for high node noise is due to the fact that large perturbations to individual nodes decrease the efficacy of the correction made by rigid body transformations. However, the transformations still make the graphs structurally similar, which explains the signifi-

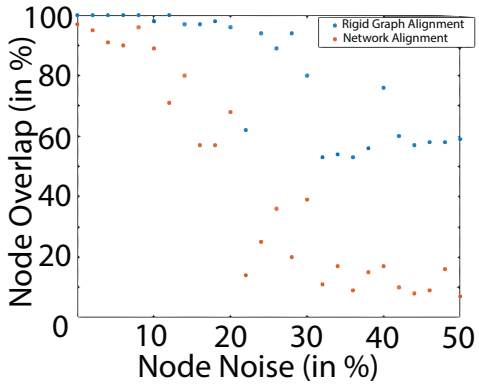


Fig. 5: **Relationship between node overlap and node noise with zero edge noise in a preferential attachment network.** Large perturbation to positions of nodes lead to inaccurate priors, which explains the decrease in node overlap. However, rigid body alignment increases similarity between the networks, which explains the difference in node overlaps between the two methods.

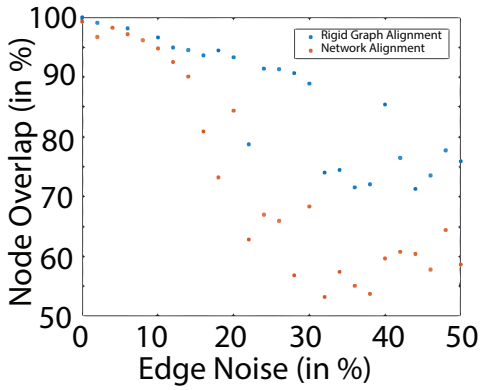


Fig. 6: **Relationship between edge overlap and node noise with zero edge noise in a preferential attachment network.** The drop in node overlap corresponds to the drop in edge overlap.

cant difference in node overlap between the two methods.

Figures S3 and S4 show results for the same experiment on $G(n,p)$ graphs. These figures show that an absence of patterns in edges of a $G(n,p)$ graph, coupled with poor priors due to perturbations in node positions make both algorithms more sensitive to noise.

4.1.3 Impact of perturbation of both nodes and edges

The effect of perturbing both nodes and edges on edge overlaps in preferential attachment networks is shown in Figures 7b and 7a. In rigid graph alignment (Figure 7b), edge overlaps are more stable for moderate amounts of both edge and node noise. Edge overlap values track true edge overlap. Edge overlap in the case of regular network alignment, however shows a sharp decline for small amounts of noise. Note that the scales on the two figures are made identical to facilitate comparison.

The effect on node overlaps in the same setting is shown in Figures 8b and 8a. Node alignment in rigid graph alignment is determined primarily by the amount of noise in the location of the nodes. However, regular alignment is far more sensitive to either noise.

4.1.4 Choice of graph aligners

We compare the performance of the previously used belief propagation algorithm (Figures 7b and 8b) with the method due to Klau et al. [17] in Figure S5 and IsoRank [14] in Figure S6 on preferential attachment networks to assess the impact of choice of graph aligners. As before, we vary edge and node noise and record the edge and node overlap in each case.

We see that the algorithm due to Klau et al. [17] is comparable with belief propagation in terms of edge overlap, however it does slightly worse in terms of node overlap when the edge noise is high ($> 20\%$). Similarly, IsoRank performs quite well in terms of edge overlap, however, node overlap degrades sharply with edge noise. Based on these observations, we conclude that our meta-algorithm can indeed be used with a wide variety readily available network aligners.

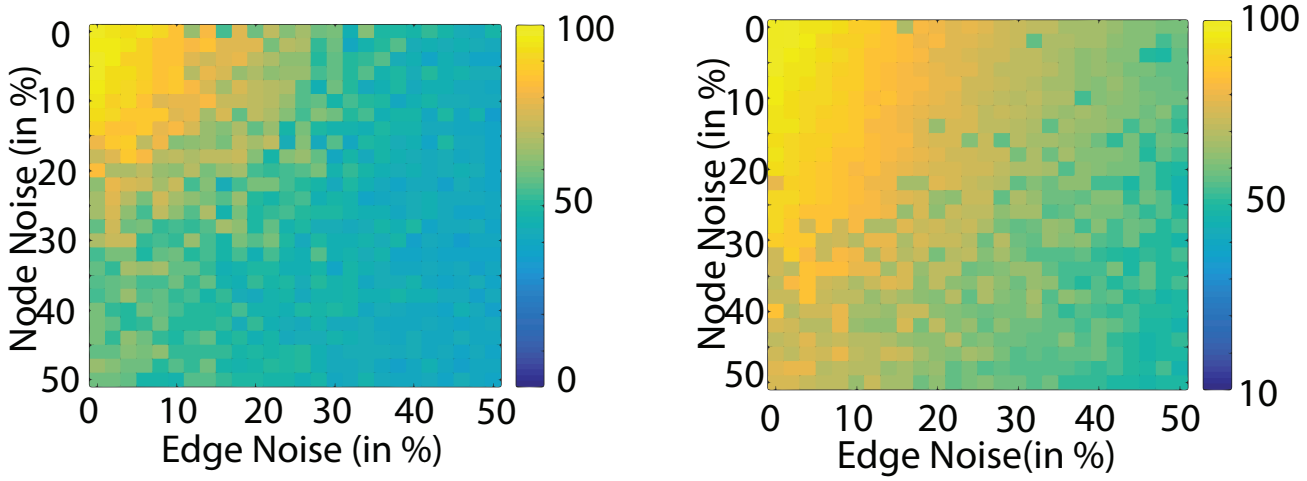
In our experiments, we have chosen different network aligners that admit a prior. This is motivated by our formulation which suggests that improved structural alignment leads to improved network alignment (and vice-versa). However, many network alignment methods such as ConeAlign [29] do not require such prior information. In Figures S7 and S8, we show that the performance of ConeAlign, as measured by edge and node overlaps is comparable to other network alignment algorithms run in isolation (i.e., outside of Rigid Graph Alignment). Since ConeAlign does not use spatial constraints to compute alignments, the edge and node overlaps are much lower than Rigid Graph Alignment, especially as more noise is introduced into the graphs. We note that a fair comparison of Rigid Graph Alignment with methods that do not admit prior information (such as ConeAlign) would be possible only if structural information is somehow encoded into its input.

4.2 Runtime Considerations

Our rigid graph alignment method is an iterative process, where a network aligner is called at every iteration. As such, it can be used in conjunction with any aligner that accepts a prior. Hence, the runtime depends primarily on the base alignment technique and is linear in the number of alternating steps. Our choice of the network alignment algorithm – *netalignmbp* [27] takes $\mathcal{O}(nnz(A \otimes B) + |E_L| + matching)$, where E_L is the number of edges in the prior L. Here, $\mathcal{O}(matching)$ is the complexity of bipartite matching (Khan et al. [47]), which depends on the matching algorithm used. In typical experiments, as we have shown, a small number of alternating steps yield significant improvement in solution quality.

4.3 Analyzing the Human Functional Connectome

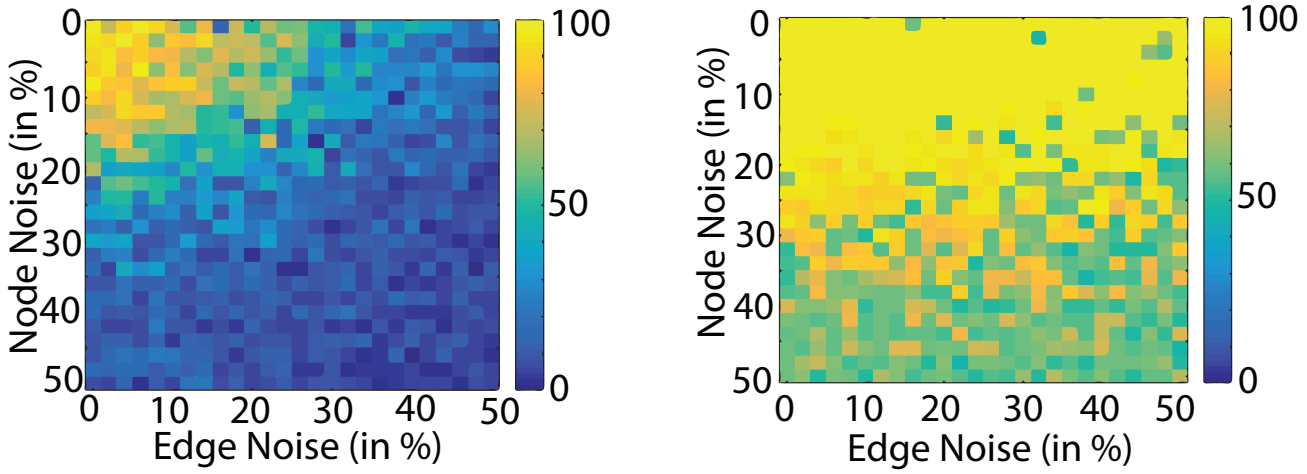
We present experimental results on networks drawn from functional human connectomes. Specifically, we apply the rigid graph alignment algorithm to distinguish between individuals – the "identifiability" problem in functional connectomes. Given a dataset of multiple functional MRIs



(a) Network alignment alone

(b) Our rigid graph alignment

Fig. 7: Heatmap showing edge overlaps for various degrees of edge and node perturbations in a preferential attachment network with: (a) regular network alignment; and (b) rigid graph alignment. In low noise regimes, the edge overlap of a regular network aligner matches the rigid graph aligner. However, at moderate noise levels, the number of edges recovered falls sharply. However, in rigid graph alignment, we are able to recover most edges. Even with higher edge noise, edge overlap closely matches the true edge overlap.



(a) Network alignment alone

(b) Our rigid graph alignment

Fig. 8: Heatmap showing node overlaps for various degrees of edge and node perturbations in preferential attachment network with: (a) regular network alignment; and (b) rigid graph alignment. In regular network alignment, node overlap is affected by increasing values of either or both noise models. However, rigid graph alignment is observed to be much more robust.

gathered from a population of individuals, we seek to find images belonging to the same individual. We do this by aligning pairs of functional networks. The intuition behind these experiments is that networks belonging to the same individual, across imaging sessions and acquired using different instruments are more similar to each other than to networks drawn from different individuals. We demonstrate that our method achieves significant improvement in terms of alignment quality over state-of-the-art network alignment techniques that do not take rigidity of graph into account. This translates to better accuracy in correctly identifying networks (and therefore, images) belonging to the same individual.

4.3.1 *Dataset*

The dataset we work with is part of the Young Adult Database of the Human Connectome Project, as described in VanEssen et al. [48]. The dataset contains medical images of the brain collected from over 1100 subjects aged between 21 and 35. The images were acquired across multiple-sites and include structural and functional modalities. They include structural, diffusion and functional MRI, Magnetoencephalography (MEG), and Electroencephalography (EEG) in both resting state and seven tasks.

The resting state functional MRI were collected in two sessions. Each session lasted for 30 minutes. The voxels were isotropic, with side $2mm$. The images were acquired once every 720 ms. The detailed acquisition protocol is presented in Smith et al. [49]. In our experiments, we used the resting state images from 20 subjects.

4.3.2 Preprocessing Steps

Functional MRIs can be thought of as a collection of time-series data. Each time-series records neuronal activity of a specific voxel in the brain, and hence has an associated 3D coordinate. Raw functional MRI data is typically noisy, and hence requires an effective pre-processing stage. A common, unavoidable source of error is due to head-motion of subjects. We correct for this error by aligning all (volumetric) images to the image in the first time slice using FSL's Linear Image Registration Tool (MCFLIRT) [50]. We follow this by skull-stripping using FSL's Brain Extraction Tool (BET) [51]. To create reasonably sized images, we re-sample all voxels to $4mm$. This allows us to run network alignment algorithms in reasonable time. We mask out all non-brain voxels, along with voxels with low variance. We vectorize the remaining relevant voxels to create a voxel $\times$ time matrix. We then use a bandpass filter with upper and lower limits set to 0.08Hz and 0.001Hz, as it has been observed that low-frequency fluctuations are effective in capturing spontaneous firings of resting-state functional MRIs. We also perform global signal regression, but do not observe significant difference in the final results if this is removed.

We compute Pearson correlation between all pairs of voxels to create a voxel $\times$ voxel similarity matrix. We sparsify this correlation matrix to create the adjacency matrices, by thresholding such that we retain only the top 5% of the (non-diagonal) entries. Each edge of this network reflects coherence in firing.

The key difference between our pre-processing pipeline and other pre-processing pipelines is that we do not register the images to a standard coordinate system (such as MNI) – if we did this, we would know the identity of each voxel, thereby making the job of the network aligner trivial. Instead, we work on un-registered, non-standarized images. We will use rigid graph alignment to register the functional networks derived from the images, using graph properties while respecting the rigidity constraints imposed by physical layout of the voxels. This has the advantage that individual specific fine-grained artifacts are not smoothed over by the registration process – thus enabling more resolved processing.

4.3.3 Rigid Graph Alignment Yields Higher Edge Overlap

First, we show that Rigid Graph Alignment substantially improves edge-overlap, which is an intuitive and commonly used network alignment metric. Recall that the edge overlap of two graphs A and B under a matching $\mathbf{X}$ is defined as $A \bullet \mathbf{X}B\mathbf{X}^T$. Since our final goal is to distinguish individuals, we define intra-subject alignment as the process of aligning pairs of networks belonging to the same individual (across sessions) and inter-subject alignment as the process of aligning pairs of networks across subjects. We run intra-subject and inter-subject alignments using Rigid Graph Alignment for a randomly selected subset of 20 subjects. We populate the prior on the basis of spatial proximity of voxels by limiting possible matches of a node in one graph to a fraction (10%) of the closest nodes in the other graph. We give equal weights to the prior, network aligner and structural aligner (i.e, $\alpha = \beta = \gamma = 1$). The convergence criterion is defined as 0.1% change in edge-overlap between consecutive iterations. We found that the percentage of edges that overlapped after only network alignment was $20.18 \pm 4.2\%$. Note that this corresponds to the performance of a network aligner that does not take rigidity of edge-lengths into account. On the other hand, *Rigid Graph Alignment* yielded edge overlap of $53.05 \pm 12.5\%$ on convergence. The improvement in performance is due to an iterative improvement in network alignment quality, driven by successive improvement of quality in structural alignment. The improvement in intra-subject alignment can clearly be seen in Figure 9. We consistently observed slower improvement in the first few iterations, owing largely to poorly (or ill-) informed priors. However, subsequent iterations rapidly improved the quality of alignment, due to more accurate priors.

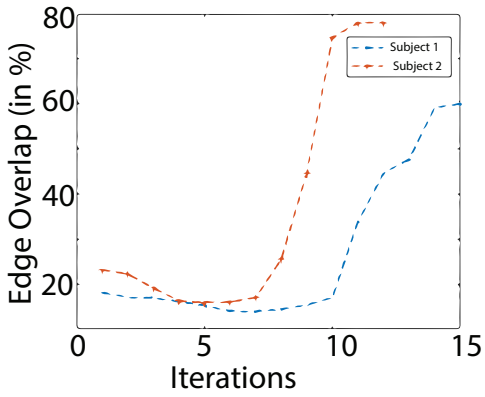


Fig. 9: **Rigid Graph Alignment increases edge-overlap when aligning two functional networks belonging to the same subject.** In this figure, we show the improvement in edge-overlap in two typical intra-subject alignments. The first iteration corresponds to the edge-overlap that is obtained by using a regular network aligner, whereas the final iteration corresponds to the edge-overlap due to rigid graph alignment. Observe that it takes few (typically less than 20) iterations for the algorithm to converge.

To show the statistical significance and robustness of our method, we perform the following test: we transform the coordinate-system of one of the brains with a random rigid-body transformation, which is to say that we randomly orient one of the brains for 100 random trials. In each case, the edge overlap by rigid graph alignment is higher than other aligners (netalignmbp, netalignmr [17], [27], IsoRank [5]).

4.3.4 Residual Error in Structural Transformation as a Metric for Network Alignment

In the previous section, we showed that rigid graph alignment improves the quality of alignment between a pair of networks. However, our goal is to identify pairs of networks belonging to the same individual from a dataset

of such networks. Our intuition is that networks drawn from the same individual must have more in common with each other than with networks drawn from other subjects. However, in terms of edge-overlap there is not a significant difference between intra-subject and inter-subject alignment with network alignment alone($20.73 \pm 4.45\%$ and $19.62 \pm 4.28\%$). The high overlap in the two histograms in Figure 10a illustrates this problem. Furthermore, after we run rigid graph alignment, the problem persists as the edge overlap of both inter-subject and intra-subject alignments improve significantly, as seen in Figure 10b. This suggests that edge-overlap is not the best metric for this application. Instead, we find that the residual error in structural alignment (Equation 8) is a better indicator of identity, as seen in Figure 10c. For intra-subject alignments, the *residual error*, normalized by the number of vertices was found to be $1.52 \pm 0.01\%$, whereas for inter-subject alignments, it was $1.99 \pm 0.025\%$. This demonstrates significantly higher distinguishability from the rigidity metric in rigid graph alignment. However, regular aligners use spatial constraints only once in the prior. In Netalignmbp, we see that residual error in case of intra-subject alignment is $9.4 \pm 1.7\%$ and is $10.14 \pm 1.5\%$. Similar results were also observed in IsoRank, with residual errors in intra- and inter-subject alignments being $9.81 \pm 2.4\%$ and $10.44 \pm 3.3\%$. In both cases, we see that the residual errors are higher, without clear separation intra- and inter-subject distributions.

5 Concluding Remarks

This paper formulates a novel and important problem of aligning a class of graphs called rigid graphs. The problem integrates topological and structural alignments into a single framework, and presents a method for computing rigid graph alignments. The method is versatile, in that it admits a range of topological and structural alignment techniques into a block coordinate descent framework. The paper presents a detailed experimental study demonstrating significant performance improvements from rigid graph alignments over prior methods. In the context of human brain connectomes, it demonstrates significant improvements in distinguishability of connectomes and identification of individual signatures in brains.

Our results open significant new avenues of research in the area of rigid graph aligners, both in terms of alternate formulations, and associated methods. For instance, we note variants of the traditional orthogonal Procrustes method, such as the weighted orthogonal Procrustes problem [52], [53] which allows a user to assign higher rewards for accurately transforming an important subset of the points. We can use these in conjunction with a network aligner that returns confidence measures for each matching. This approach fits within our framework of rigid graph alignment for specific classes of applications, as it ties the structural aligner more tightly with the network aligner.

Availability of Data and Materials

The data from the Human Connectome Project is made available by the HCP Consortium in http://www.humanconnectomeproject.org/. The MATLAB implementation of the alignment algorithms are available in https://www.cs.purdue.edu/homes/dgleich/codes/netalign.

Funding

The authors are supported by the National Science Foundation grants IIS-1546488, CCF-1909528, IIS-2007481, Center for Science of Information STC, CCF-0939370, as well as NASA and the Sloan Foundation.

References

[1] M. E. J. Newman, "Modularity and community structure in networks," vol. 103, no. 23, p. 8577, 2006. [Online]. Available: http://www.pnas.org/content/103/23/8577.abstract

[2] D. Godwin, R. L. Barry, and R. Marois, "Breakdown of the brain's functional network modularity with awareness," *Proceedings of the National Academy of Sciences*, vol. 112, no. 12, p. 3799, 2015. [Online]. Available: http://www.pnas.org/content/112/12/3799.abstract

[3] S. P. Borgatti, "Centrality and network flow," *Social networks*, vol. 27, no. 1, pp. 55–71, 2005.

[4] C. F. Negre, U. N. Morzan, H. P. Hendrickson, R. Pal, G. P. Lisi, J. P. Loria, I. Rivalta, J. Ho, and V. S. Batista, "Eigenvector centrality for characterization of protein allosteric pathways," *Proceedings of the National Academy of Sciences*, vol. 115, no. 52, pp. E12 201–E12 208, 2018.

[5] R. Singh, J. Xu, and B. Berger, "Pairwise global alignment of protein interaction networks by matching neighborhood topology," in *Proceedings of the 11th Annual International Conference on Research in Computational Molecular Biology*, ser. RECOMB'07. Berlin, Heidelberg: Springer-Verlag, 2007, pp. 16–31.

[6] H. Yin, A. R. Benson, J. Leskovec, and D. F. Gleich, "Local higher-order graph clustering," in *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2017, pp. 555–564.

[7] S. Aibar, C. B. González-Blas, T. Moerman, H. Imrichova, G. Hulselmans, F. Rambow, J.-C. Marine, P. Geurts, J. Aerts, J. van den Oord *et al.*, "Scenic: single-cell regulatory network inference and clustering," *Nature methods*, vol. 14, no. 11, pp. 1083–1086, 2017.

[8] E. S. Finn, X. Shen, D. Scheinost, M. D. Rosenberg, J. Huang, M. M. Chun, X. Papademetris, and R. T. Constable, "Functional connectome fingerprinting: identifying individuals using patterns of brain connectivity," *Nature Neuroscience*, vol. 18, no. 11, 2015.

[9] V. Ravindra, H. Nassar, D. F. Gleich, and A. Grama, "Rigid graph alignment," in *Complex Networks and Their Applications VIII*, H. Cherifi, S. Gaito, J. F. Mendes, E. Moro, and L. M. Rocha, Eds. Cham: Springer International Publishing, 2020, pp. 621–632.

[10] E. H. Sussenguth, "A graph-theoretic algorithm for matching chemical structures." *Journal of Chemical Documentation*, vol. 5, no. 1, pp. 36–43, 1965.

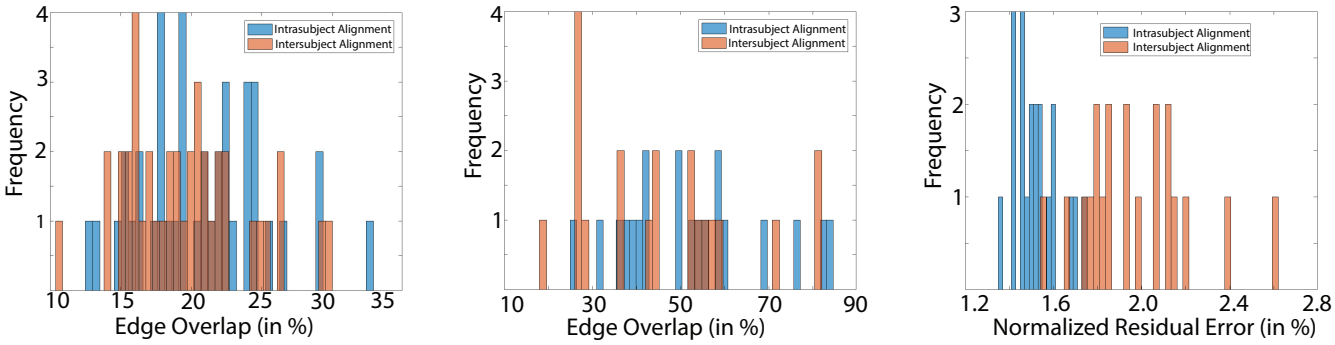
[11] B. Zelinka, "On a certain distance between isomorphism classes of graphs," *Časopis pro pěstování matematiky*, vol. 100, no. 4, pp. 371–373, 1975.

[12] ——, "Distances between graphs (extended abstract)," in *Fourth Czechoslovakian Symposium on Combinatorics, Graphs and Complexity*, ser. Annals of Discrete Mathematics. Elsevier, 1992, vol. 51, pp. 355 – 361.

[13] S. Umeyama, "An eigendecomposition approach to weighted graph matching problems," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 10, no. 5, pp. 695–703, Sep. 1988.

[14] R. Singh, J. Xu, and B. Berger, "Pairwise global alignment of protein interaction networks by matching neighborhood topology," in *Research in Computational Molecular Biology*, T. Speed and H. Huang, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2007, pp. 16–31.

[15] G. Kollias, S. Mohammadi, and A. Grama, "Network similarity decomposition (nsd): A fast and scalable approach to network alignment," *IEEE Trans. on Knowl. and Data Eng.*, vol. 24, no. 12, pp. 2232–2243, Dec. 2012. [Online]. Available: http://dx.doi.org/10.1109/TKDE.2011.174



(a) (b) (c)

Fig. 10: Histograms showing distributions for quality of intra- and intersubject alignments. (a) Edge-overlap metric after only network alignment. The poor edge overlap suggests that regular network alignment alone is ineffective for the task of identifiability in functional connectomes. (b) Edge overlap, after rigid graph alignment. While Rigid Graph Alignment increases the number of overlapping edges, it does not separate the two distributions. The high intersection suggests edge-overlap is an ill-suited metric for the task of identifiability in functional connectomes. (c) Residual error in structural alignment for inter-subject and intra-subject alignments. Here, the values are much better separated than edge overlap, suggesting that this metric is better suited for brain fingerprints than edge overlap

[16] C.-S. Liao, K. Lu, M. Baym, R. Singh, and B. Berger, "Isorankn: spectral methods for global alignment of multiple protein networks," *Bioinformatics*, vol. 25, no. 12, pp. i253–i258, 2009.

[17] G. W. Klau, "A new graph-based method for pairwise global network alignment," *BMC Bioinformatics*, vol. 10, no. 1, p. S59, Jan 2009. [Online]. Available: https://doi.org/10.1186/1471-2105-10-S1-S59

[18] G. Ciriello, M. Mina, P. H. Guzzi, M. Cannataro, and C. Guerra, "Alignnemo: A local network alignment method to integrate homology and topology," *PLOS ONE*, vol. 7, no. 6, pp. 1–14, 06 2012.

[19] J. Berg and M. Lässig, "Local graph alignment and motif search in biological networks," *Proceedings of the National Academy of Sciences*, vol. 101, no. 41, pp. 14 689–14 694, 2004. [Online]. Available: https://www.pnas.org/content/101/41/14689

[20] M. Heimann, W. Lee, S. Pan, K.-Y. Chen, and D. Koutra, "Hashalign: Hash-based alignment of multiple graphs," in *Advances in Knowledge Discovery and Data Mining*, D. Phung, V. S. Tseng, G. I. Webb, B. Ho, M. Ganji, and L. Rashidi, Eds. Cham: Springer International Publishing, 2018, pp. 726–739.

[21] V. Vijayan and T. Milenkovic, "Multiple network alignment via multimagna++," *IEEE/ACM Trans. Comput. Biol. Bioinformatics*, vol. 15, no. 5, pp. 1669–1682, Sep. 2018. [Online]. Available: https://doi.org/10.1109/TCBB.2017.2740381

[22] H. Nassar, G. Kollias, A. Grama, and D. F. Gleich, "Low rank methods for multiple network alignment," 2018.

[23] O. Kuchaiev, T. Milenković, V. Memišević, W. Hayes, and N. Pržulj, "Topological network alignment uncovers biological function and phylogeny," *Journal of The Royal Society Interface*, 2010. [Online]. Available: http://rsif.royalsocietypublishing.org/content/early/2010/03/24/rsif.2010.0063

[24] R. Patro and C. Kingsford, "Global network alignment using multiscale spectral signatures," *Bioinformatics*, vol. 28, no. 23, pp. 3105–3114, 2012.

[25] S. Mohammadi, D. F. Gleich, T. G. Kolda, and A. Grama, "Triangular alignment tame: A tensor-based approach for higher-order network alignment," *IEEE/ACM Trans. Comput. Biol. Bioinformatics*, vol. 14, no. 6, pp. 1446–1458, Nov. 2017. [Online]. Available: https://doi.org/10.1109/TCBB.2016.2595583

[26] S. Feizi, G. Quon, M. R. Mendoza, M. Médard, M. Kellis, and A. Jadbabaie, "Spectral alignment of networks," *CoRR*, vol. abs/1602.04181, 2016. [Online]. Available: http://arxiv.org/abs/1602.04181

[27] M. Bayati, D. F. Gleich, A. Saberi, and Y. Wang, "Message-passing algorithms for sparse network alignment," *ACM Trans. Knowl. Discov. Data*, vol. 7, no. 1, pp. 3:1–3:31, Mar. 2013. [Online]. Available: http://doi.acm.org/10.1145/2435209.2435212

[28] M. Heimann, H. Shen, T. Safavi, and D. Koutra, "Regal," *Proceedings of the 27th ACM International Conference on Information and Knowledge Management - CIKM '18*, 2018. [Online]. Available: http://dx.doi.org/10.1145/3269206.3271788

[29] X. Chen, M. Heimann, F. Vahedian, and D. Koutra, "Cone-align: Consistent network alignment with proximity-preserving node embedding," in *Proceedings of the 29th ACM International Conference on Information & Knowledge Management*, 2020, pp. 1985–1988.

[30] C. Riederer, Y. Kim, A. Chaintreau, N. Korula, and S. Lattanzi, "Linking users across domains with location data: Theory and validation," in *Proceedings of the 25th International Conference on World Wide Web*, ser. WWW '16. Republic and Canton of Geneva, Switzerland: International World Wide Web Conferences Steering Committee, 2016, pp. 707–719. [Online]. Available: https://doi.org/10.1145/2872427.2883002

[31] S. Melnik, H. Garcia-Molina, and E. Rahm, "Similarity flooding: a versatile graph matching algorithm and its application to schema matching," in *Proceedings 18th International Conference on Data Engineering*. IEEE, 2002, pp. 117–128.

[32] P. Buneman and S. Staworko, "Rdf graph alignment with bisimulation," in *International Conference on Very Large Databases (VLDB)*, ser. Proceedings of the VLDB Endowment, vol. 9, 2016, pp. 1149–1160.

[33] S. Zhang and H. Tong, "Final: Fast attributed network alignment," in *Proceedings of the 22nd ACM SIGKDD International Conference on knowledge discovery and data mining*, ser. KDD '16. ACM, 2016, pp. 1345–1354.

[34] L. Wiskott, N. Krüger, N. Kuiger, and C. Von Der Malsburg, "Face recognition by elastic bunch graph matching," *IEEE Transactions on pattern analysis and machine intelligence*, vol. 19, no. 7, pp. 775–779, 1997.

[35] F. Zhou, J. Brandt, and Z. Lin, "Exemplar-based graph matching for robust facial landmark localization," in *Proceedings of the IEEE International Conference on Computer Vision*, 2013, pp. 1025–1032.

[36] B. R. Conroy and P. J. Ramadge, "The grouped two-sided orthogonal procrustes problem," in *2011 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, May 2011, pp. 3688–3691.

[37] P. Besl and H. McKay, "A method for registration of 3-d shapes. ieee trans pattern anal mach intell," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 14, pp. 239–256, 03 1992.

[38] P. H. Schönemann, "A generalized solution of the orthogonal procrustes problem," *Psychometrika*, vol. 31, no. 1, pp. 1–10, Mar 1966.

[39] W. Kabsch, "A solution for the best rotation to relate two sets of vectors," *Acta Crystallographica Section A*, vol. 32, no. 5, pp. 922–923, Sep 1976.

[40] B. Sabata and J. Aggarwal, "Estimation of motion from a pair of range images: A review," *CVGIP: Image Understanding*, vol. 54, no. 3, pp. 309 – 324, 1991. [Online]. Available: http://www.sciencedirect.com/science/article/pii/104996609190032K

[41] D. Eggert, A. Lorusso, and R. Fisher, "Estimating 3-d rigid body transformations: a comparison of four major algorithms," *Machine Vision and Applications*, vol. 9, no. 5, pp. 272–290, Mar 1997.
[42] A.-L. Barabási and R. Albert, "Emergence of scaling in random networks," *Science*, vol. 286, no. 5439, pp. 509–512, 1999. [Online]. Available: http://science.sciencemag.org/content/286/5439/509
[43] E. N. Gilbert, "Random graphs," *Ann. Math. Statist.*, vol. 30, no. 4, pp. 1141–1144, 12 1959. [Online]. Available: https://doi.org/10.1214/aoms/1177706098
[44] J. Dall and M. Christensen, "Random geometric graphs," *Phys. Rev. E*, vol. 66, p. 016121, Jul 2002. [Online]. Available: https://link.aps.org/doi/10.1103/PhysRevE.66.016121
[45] R. B. Ellis, J. L. Martin, and C. Yan, "Random geometric graph diameter in the unit ball," *Algorithmica*, vol. 47, no. 4, pp. 421–438, 2007.
[46] D. Funke, S. Lamm, U. Meyer, M. Penschuck, P. Sanders, C. Schulz, D. Strash, and M. von Looz, "Communication-free massively distributed graph generation," *Journal of Parallel and Distributed Computing*, vol. 131, pp. 200–217, 2019.
[47] A. M. Khan, D. F. Gleich, A. Pothen, and M. Halappanavar, "A multithreaded algorithm for network alignment via approximate matching," in *Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis*, ser. SC '12. Los Alamitos, CA, USA: IEEE Computer Society Press, 2012, pp. 64:1–64:11.
[48] D. C. V. Essen, S. M. Smith, D. M. Barch, T. E. Behrens, E. Yacoub, and K. Ugurbil, "The wu-minn human connectome project: An overview," *NeuroImage*, vol. 80, pp. 62 – 79, 2013.
[49] S. M. Smith, C. F. Beckmann, J. Andersson, E. J. Auerbach, J. Bijsterbosch, G. Douaud, E. Duff, D. A. Feinberg, L. Griffanti, M. P. Harms, M. Kelly, T. Laumann, K. L. Miller, S. Moeller, S. Petersen, J. Power, G. Salimi-Khorshidi, A. Z. Snyder, A. T. Vu, M. W. Woolrich, J. Xu, E. Yacoub, K. Uğurbil, D. C. V. Essen, and M. F. Glasser, "Resting-state fMRI in the human connectome project," *NeuroImage*, vol. 80, pp. 144 – 168, 2013.
[50] M. Jenkinson, P. Bannister, M. Brady, and S. Smith, "Improved optimization for the robust and accurate linear registration and motion correction of brain images," *NeuroImage*, vol. 17, no. 2, pp. 825 – 841, 2002.
[51] S. M. Smith, "Fast robust automated brain extraction," *Human Brain Mapping*, vol. 17, no. 3, pp. 143 – 155, November 2002.
[52] R. Lissitz, P. Schönemann, and J. Lingoes, "A solution to the weighted procrustes problem in which the transformation is in agreement with the loss function," *Psychometrika*, vol. 41, no. 4, pp. 547–550, 1976.
[53] A. Mooijaart and J. Commandeur, "A general solution of the weighted orthonormal procrustes problem," *Psychometrika*, vol. 55, no. 4, pp. 657–663, 1990.

Vikram Ravindra received his B.E. from Visvesvaraya Technological University, M.Sc from TU Munich and his Ph.D. from Purdue University. Starting Fall '22, he will join the University of Cincinnati as an Assistant Professor in the Department of Computer Science. His research interests are biomedical image analysis, computational biology and rehabilitation robotics.

Huda Nassar received her B.S. and B.A. in Computer Science and Mathematics from the American University of Beirut, and received her Ph.D. in Computer Science from Purdue University. Huda is senior computer scientist at RelationalAI, a startup company focused on building a relational knowledge graph management system, and her focus is on graph algorithms and query optimization.

David F. Gleich received a B.Sc. degree from Harvey Mudd College, Claremont, CA, USA and the Ph.D. degree from Stanford University, Stanford, CA, USA. He is the Jyoti and Aditya Mathur Associate Professor of Computer Science at Purdue University, West Lafayette, IN, USA. His research is on matrix computations, network and graph algorithms, and parallel and distributed computing. Prof. Gleich has been awarded an NSF CAREER award, a Sloan Research Fellowship, and a SIAM Outstanding Paper Prize.

Ananth Grama got his B. Engg. from the Indian Institute of Technology at Roorkee (1989), M.S. from Wayne State University (1990), and Ph.D. from the University of Minnesota (1996). He is the Samuel Conte Professor of Computer Science at Purdue University. His research interests are parallel and distributed computing, large scale data analytics, and applications. He is the recipient of the NSF CAREER award, a Distinguished Alumnus of the University of Minnesota, and a Fellow of the American Association for Advancement of Sciences.