2022

Graph matching with applications to network analysis

BOSTON UNIVERSITY

GRADUATE SCHOOL OF ARTS AND SCIENCES

Dissertation

GRAPH MATCHING WITH APPLICATIONS TO NETWORK ANALYSIS

by

ZIHUAN QIAO

B.S., Renmin University of China, 2016
M.S., Boston University, 2018

Submitted in partial fulfillment of the

requirements for the degree of

Doctor of Philosophy

2022

Approved by

First Reader

Daniel Sussman, PhD
Assistant Professor of Mathematics and Statistics

Second Reader

Luis Carvalho, PhD
Associate Professor of Mathematics and Statistics

Third Reader

Masanao Yajima, PhD
Associate Professor of the Practice of Mathematics and Statistics

Fourth Reader

Judith Lok, PhD
Associate Professor of Mathematics and Statistics

Acknowledgments

Firstly, a huge thanks to my advisor, Daniel L. Sussman, for his constant support and guidance during my research journey. While steering through my research, Dan always encouraged me to explore new research projects for my own interests with extraordinary support. Dan also offered valuable opportunities in helping build academic connections and fostering my growth to become a more mature researcher. It has been a great pleasure and my honor to work with Dan.

I would also like to thank my friends and fellow graduate students in the Department of Mathematics and Statistics, particularly, Keer, Qiuyun, Huaiyu, Esther, Kelly, and Ben. I believe after many years when I look back, I will still clearly remember the days we had discussions on coursework and research, and those small talks in the basement offices around the corner.

Finally, many thanks to my boyfriend, Zhe Feng, and my dear parents, Yi Qiao and Xin Wang. Their unconditional love and support have been important for me along the way and especially in the past intense months.

Zihuan Qiao

GRAPH MATCHING WITH APPLICATIONS TO NETWORK ANALYSIS

ZIHUAN QIAO

Boston University, Graduate School of Arts and Sciences, 2022

Major Professor: Daniel Sussman, PhD
Assistant Professor of Mathematics and Statistics

ABSTRACT

The graph matching (GM) problem seeks to find an alignment between the vertex sets of two graphs and has applications in social network analysis, bioinformatics, and pattern recognition. In this dissertation, we propose to (a) develop the `iGraphMatch` `R` package for common GM algorithms and steps for analysis of GM, (b) develop the mutual nearest neighbor algorithm for matching node pairs with high precision in polynomial time, and (c) analyze a user de-anonymization problem based on their Venmo transaction networks along with various information associated with users and their transactions. The `iGraphMatch` package enables seamless matching of generalized graphs with versatile options for the form of input graphs and the specification of available prior information, as well as evaluation of matching performance, and visualization. For the methodology component, we empirically demonstrate the effectiveness of the mutual nearest neighbor algorithm for finding a high precision sub-match. In addition, we develop a mathematical framework for

the analysis of the proposed method and provide conditions for achieving high matching precision under a correlated random graph model. Lastly, for the challenging GM problem on Venmo transaction networks, we introduce a similarity-based learning method for integrating multiple features into a single similarity score that optimizes the expected number of correctly matched nodes.

Contents

List of Tables

List of Figures

List of Abbreviations

ACM	.....	Association for Computing Machinery
CE	.....	Common Edges
CNE	.....	Common Non-Edges
EC	.....	Edge Correctness
EE	.....	Extra Edges
FW	.....	Frank-Wolfe
GM	.....	Graph Matching
ICML	.....	International Conference on Machine Learning
IEEE	.....	Institute of Electrical and Electronics Engineers
IMA	.....	Institute of Mathematics and its Applications
KNN	.....	K-Nearest Neighbors
LAP	.....	Linear Assignment Problem
LCCS	.....	Largest Common Connected Subgraph
MAP@3	.....	Mean Average Precision @ 3
ME	.....	Missing Edges
MNN	.....	Mutual Nearest Neighbor
MP	.....	Matching Pairs
NCA	.....	Neighborhood Components Analysis
NCA4GM	.....	NCA for Graph Matching
PA	.....	Procrustes Analysis
QAP	.....	Quadratic Assignment Problem
QDA	.....	Quadratic Discriminant Analysis
$\mathbb{R}^2$	.....	the Real plane
RDPG	.....	Random Dot Product Graphs
RL	.....	Record Linkage
SIAM	.....	Society for Industrial and Applied Mathematics
VLDB	.....	Very Large Data Bases
w.l.o.g.	.....	Without Loss of Generality
$\mathbb{Z}$	.....	the Set of Integers

Chapter 1

Introduction

The graph matching (GM) problem seeks to find an alignment between the vertex sets of multiple graphs that best preserves common structure across graphs. This is often posed as minimizing edge disagreements of two graphs over all alignments. Formally, given A and B, two adjacency matrices corresponding to two graphs $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$, the goal is to find

$$\underset{P \in \Pi}{\operatorname{argmin}} \|A - PBP^\top\|_F^2$$

where Π is the set of all permutation matrices. More generally, the problem of discovering some true latent alignment between two networks can often be regarded as variations on the above problem by adjusting the objective function for the setting.

GM has wide applications in diverse fields, such as pattern recognition (Caelli and Kosinov, 2004; Conte et al., 2004), machine learning (Liu and Qiao, 2012; Cour et al., 2006), bioinformatics (Uetz et al., 2000; Ito et al., 2001), neuroscience (Chen et al., 2016), social network analysis (Narayanan and Shmatikov, 2009), and knowledge graph queries (Hu et al., 2018). For example, in the field of pattern recognition, local features of images are for-

mulated as graph nodes and edges represent proximity of two features and graph matching can be applied to find a correspondence between nodes in order to align two images (Berg et al., 2005). Another illustrative application example is from the field of bioinformatics. At least one-third of the functional annotations of the typical proteome's proteins remain unknown. In the formulated graphs, each node represents a protein and edges indicate the functional between them. Graph matching can be used to determine protein function by aligning to another protein graph with revealed functions (Nabieva et al., 2005). In social network analysis, graph matching can be used for de-anonymization of a network and enables methods for joint graph inference that rely on aligned networks (Narayanan and Shmatikov, 2009).

The well-known graph isomorphism problem is a special case of GM problem when there exists a bijection between the nodes of two graphs which exactly preserves the edge structure. In terms of computational complexity, GM is equivalent to the NP-hard quadratic assignment problem, which is considered a very challenging problem where few theoretical guarantees exist, even in special cases (Finke et al., 1987). For certain problems where the graphs are nearly isomorphic, polynomial-time algorithms do exist (Aflalo et al., 2015a; Umeyama, 1988) but these methods frequently break down for more challenging instances.

In this dissertation, we study the GM problem from methodological, theoretical, applied perspectives. A key focus is on the incorporation of information known about the alignment, nodes, and edges to improve GM performance. In particular, we present the detailed functionality of `iGraphMatch`, an R package that we implemented which serves as a practical tool for the use

of prevalent graph matching methodologies. We propose the fast Mutual Nearest Neighbor (MNN) algorithm for generating a partial alignment between graphs utilizing seeded neighborhood information and we provide empirical and theoretical evidence for its high precision in identifying a partial match. We also investigate the problem of building a single similarity score by integrating multiple information associated with nodes and edges through a proposed similarity-based learning method adapted for the GM downstream task.

As a practical complement to the rapid growth of GM algorithms developed for finding approximations to the original GM problem, the `iGraphMatch` R package implements several prevalent GM algorithms. These algorithms utilize either the spectral embedding of vertices (Umeyama, 1988), or relaxations of the objective function (Zaslavskiy et al., 2009; Lyzinski et al., 2016), or apply ideas from percolation theory (Yartseva and Grossglauser, 2013; Kazemi et al., 2015). The `iGraphMatch` package provides versatile options of working with graphs in the form of matrices, igraph objects or lists of either, and matching graphs under a generalized setting: weighted, directed, graphs of a different order, and multilayer graphs.

In addition to the topological structure of graphs, many algorithms incorporate prior information in the form of seeds, or similarities, or both to alleviate the computational challenge of the GM problem. The `iGraphMatch` package enables seeds and similarities for all the implemented algorithms. Seeds, or anchors, refer to partial knowledge of the alignment of two graphs. In practice, seeds can be users with the same name and location across different social networks or pairs of genes with the same DNA sequences. Some

algorithms like the percolation algorithm (Yartseva and Grossglauser, 2013; Kazemi et al., 2015) which matches two graphs by propagating matching information to neighboring pairs require seeds to kick off. Mossel and Xu (2020) develop GM algorithms that utilize large seeded neighboring statistics to nominate plausible matches. All algorithms improve substantially by incorporating seeds and can achieve accurate matching in polynomial time (Lyzinski et al., 2014).

The proposed MNN algorithm is a seeded GM method that introduces a procedure for generating a high-precision partial mapping which evolves over iterations. We observe a dramatic improvement of matching precision over other seeded GM methods on several real data sets. We also investigate this high precision from a theoretical perspective. The high quality matches can be leveraged to initialize subsequent GM analysis closer to the truth. The method stops matching automatically when there is no sufficient evidence supporting additional matches and the process is free of hyperparameters. Another principal advantage of the proposed method is its computational efficiency and scalability, that it has a computational complexity of $O(n^2)$ for each iteration and could be easily parallelizable.

Similarity scores are another commonly used form of prior information which measures the similarity between pairs of nodes across the graphs. In the bioinformatics area, the BLAST similarity score is an example of a similarity score that plays an important role in aligning two protein-protein interaction networks (Singh et al., 2008). Similarity scores are usually generated from nodal covariates that are observed in both networks (Kelley et al., 2004; Belongie et al., 2002). When there are multiple attributes associated with nodes

and edges, it is not trivial to generate a similarity score that integrates all the information and optimizes the objective function for the GM task. The last component of this dissertation develops methods for learning similarity scores in order to solve a Venmo user de-anonymization problem based on user transaction graphs from two different months. Various information associated with users and their transactions, such as transaction topics, locations and dates are used to form similarities. We propose to learn two linear transformations on the attribute vector spaces of two graphs separately to accommodate for the difference in user behavior across graphs by optimizing for the expected number of correctly aligned users on the transformed space.

The rest of the dissertation is organized as follows. Chapter 2 details the methods and functionalities implemented in the `iGraphMatch` R package with examples on real datasets illustrating how to conduct graph matching analysis with the package step by step. We discuss the trade-off between efficiency and effectiveness among the GM algorithms and provide insights into the preference of different methods under different cases. Then in chapter 3, we introduce the Mutual Nearest Neighbor algorithm followed by empirical results demonstrating its dramatic improvement in terms of matching precision and running time. We investigate theoreticall why MNN tends to generate a set of matched nodes with higher precision compared to other methods. The following chapter 4 states the Venmo users' de-anonymization problem and reduces it to a similarity-based learning problem. We present the proposed Neighborhood Component Analysis method adapted for the GM task, and demonstrate its performance on the simulated data and apply it to the Venmo transaction graph matching problem. Chapter 5 concludes the dissertation

with discussions on future work.

Chapter 2

`iGraphMatch`: an R package for the analysis of graph matching

In this chapter, we provide a background review of the methods and functionalities implemented in the `iGraphMatch` package, detailed usage guidelines through real data examples followed by a discussion on the trade-off between efficiency and effectiveness of different GM methods.

Although there exist some open source software and packages containing graph matching functionality, `iGraphMatch` package provides a centralized repository for common graph matching methodologies with flexibility, tools for developing graph matching problem methodology, as well as metrics for evaluating and tools for visualizing matching performance. Among the alternative GM packages, the most relevant ones include the `igraph` (Csardi and Nepusz, 2006) package which focuses on descriptive network analysis and graph visualization based on igraph objects and provides a single graph matching algorithm, the `GraphM` (Zaslavskiy et al., 2009) package which implements several GM algorithms proposed between 1999 and 2009 in C, and the `Corbi` (Huang et al., 2013) R package which is particularly designed for studies in bioinformatics and `SpecMatch` (Mateus et al., 2008) which only involves implementations of spectral embedding based GM algorithms and written in

C/C++. None of these packages provide the breadth of state-of-the-art tools, flexibility, and ease-of-use provided by the `iGraphMatch` package.

2.1 Graph matching background

In this section, we give background on graph matching and related problems followed by descriptions of the principal algorithms implemented in `iGraphMatch`. For simplicity, we state all the algorithms in the context of matching undirected, unweighted graphs with the same cardinality. All algorithms can also be directly applied to directed and weighted graphs. In the second subsection, we discuss the techniques for matching graphs with a different number of vertices along with other extensions. To conclude the section, we introduce the statistical models for correlated networks and discuss measures for the goodness of matching.

For the remainder of this paper we use the following notations. Let $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$ denote two graphs with n vertices. Let A and B be their corresponding binary symmetric adjacency matrices. In the setting of seeded graph matching, suppose without loss of generality, the first s pairs of nodes are seeds for simplicity. In `iGraphMatch`, much more flexible seed specifications are possible. Accordingly, let A and B be partitioned as:

$$A = \begin{bmatrix} A_{11} & A_{21}^\top \\ A_{21} & A_{22} \end{bmatrix} \text{ and } B = \begin{bmatrix} B_{11} & B_{21}^\top \\ B_{21} & B_{22} \end{bmatrix} \tag{2.1}$$

where $A_{11}, B_{11} \in \{0,1\}^{s \times s}$ denote seed-to-seed adjacencies, $A_{21}, B_{21} \in \{0, 1\}^{(n-s)\times s}$ denote nonseed-to-seed adjacencies and $A_{22}, B_{22} \in \{0,1\}^{(n-s)\times(n-s)}$ denote nonseed-to-nonseed adjacencies. Let S be an n-by-n real-valued matrix

of similarity scores. Let Π be the set of all permutation matrices and $\mathcal{D}$ be the set of all doubly stochastic matrices.

Assignment problems Matching or assignment problems are core problems in combinatorial optimization and appear in numerous fields (Burkard et al., 2012). As we illustrate in Equation 2.4, a general version of the graph matching problem is equivalent to the quadratic assignment problem (QAP). Similarly, QAP is related to the linear assignment problem (LAP) which also plays a role in GM. The LAP asks how to assign n items (eg. workers or nodes in G_1) to n other items (eg. tasks or nodes in G_2) with minimum cost. Let C denote an $n \times n$ cost matrix, where C_{ij} denotes the cost of matching i to j, then the LAP is to find

$$\underset{P \in \Pi}{\operatorname{argmin}} \quad \operatorname{trace}(C^\top P) \tag{2.2}$$

LAP is solvable in $O(n^3)$ time and there are numerous exact and approximate methods for both general (Jonker and Volgenant, 2005; Kuhn, 1955) and special cases, such as sparse cost matrices (Volgenant, 1996).

The statement of QAP resembles LAP, except that the cost function is expressed as a quadratic function. Given two n-by-n matrices A and B which can represent flows between facilities and the distance between locations respectively, or the adjacency matrices of two unaligned graphs, the objective function for QAP is:

$$\underset{P \in \Pi}{\operatorname{argmax}} \operatorname{trace}(APBP^\top). \tag{2.3}$$

This problem is NP-hard (Finke et al., 1987) leading to a core challenge for any graph matching approach. As will be illustrated in the rest of the section, some

matching algorithms reduce the graph matching problem to solving a LAP. For these algorithms, we include similarity scores S by adding an additional term trace$(S^\top P)$ to the reduced objective function.

2.1.1 Graph matching algorithms

In the `iGraphMatch` package, we implement three types of prevalent GM algorithms. The first group uses relaxations of the objective function, including convex, concave, and indefinite relaxations. The second group consists of algorithms that apply ideas from percolation theory, where matching information is spread from an initial set of matched nodes. The last group is based on the spectral embedding of vertices.

Relaxation-based algorithms

These approaches relax the constraint that P is a permutation matrix to require only that P is doubly stochastic, optimizing over $\mathcal{D}$, the convex hull of Π. When P is a permutation matrix

$$\|A - PBP^\top\|_F^2 = \|AP - PB\|_F^2 = \|A\|_F^2 + \|B\|_F^2 - 2 \cdot \text{trace} APBP^\top. \quad (2.4)$$

However, these equalities do not hold for all $P \in \mathcal{D}$, leading to different relaxations.

The second term of Equation 2.4 is a convex function of permutation matrices and optimizing it over $P \in \mathcal{D}$ gives the convex relaxation, where the gradient at P to the convex relaxed objective function is $-4APB + 2A^\top AP + 2PBB^\top$. The last equality in Equation 2.4 shows that minimizing edge disagreements is equivalent to maximizing the number of edge agree-

ments, trace$APBP^\top$, a QAP. Optimizing the indefinite function over $\mathcal{D}$ gives the indefinite relaxation with gradient $-2APB$ (Lyzinski et al., 2016).

Table 2.1: Summary of relaxation methods.

Relaxation	Objective function	Domain	Optimization guarantee	Optimum form
None	$\|A - PBP^T\|_F^2$	Π		Π
Indefinite	$trBDAD^T$	$\mathcal{D}$	Local	$\mathcal{D}$ (often Π)
Convex	$\|AD - DB\|_F^2$	$\mathcal{D}$	Global	$\mathcal{D}$
Concave	$-tr(\Delta D) -$ $2tr(L_1^T D L_2 D^T)$	$\mathcal{D}$	Local	Π

Generally, the convex relaxation leads to a solution that is not guaranteed to be near the solution to the original GM. However, Aflalo et al. (2015a) introduced the class of "friendly" graphs based on the spectral properties of the adjacency matrices to characterize the applicability of the convex relaxation. Matching two friendly graphs by using the convex relaxation is guaranteed to find the exact solution to the GM problem. Unfortunately, this class is quite limiting and does not hold for most statistical models or real-world examples.

Another relaxation is the concave relaxation used in the PATH algorithm (Zaslavskiy et al., 2009). The concave relaxation uses the Laplacian matrix defined as $L = D - A$, where D is the diagonal degree matrix with diagonal entries $D_{ii} = \sum_{i=1}^{N} A_{ij}$. Assume L_i and D_i, $i = 1, 2$, are the Laplacian matrices and degree matrices for G_1 and G_2 respectively, then we can rewrite the

objective function as

$$\begin{aligned}\|A - PBP^\top\|_F^2 &= \|AP - PB\|_F^2 \\ &= \|(D_1P - PD_2) - (L_1P - PL_2)\|_F^2 \\ &= -\text{trace}(\Delta P) + \text{trace}(L_1^2) + \text{trace}(L_2^2) - 2\text{trace}(L_1^\top P L_2 P^\top),\end{aligned} \tag{2.5}$$

where the matrix $\Delta_{ij} = (D_{2_{jj}} - D_{1_{ii}})^2$. Dropping the terms not dependent on P in Equation 2.5, we obtain the concave function $-\text{trace}(\Delta P) - 2\text{trace}(L_1^\top P L_2 P^\top)$ on $\mathcal{D}$.

A summary of the different relaxations is provided in Table 2.1. Relaxing the discrete problem to a continuous problem breaks the equivalence to the original formulation of the edge disagreement and enables employing algorithms based on gradient descent.

Frank Wolfe methodology Lyzinski et al. (2016) introduced an algorithm for the relaxed graph matching problem, with each iteration computable in polynomial time, that can find local optima for the relaxations above. The Frank-Wolfe (FW) (Frank and Wolfe, 1956) methodology is an iterative gradient ascent approach composed of two steps. The first step finds an ascent direction that maximizes the gradient ascent. In this case the ascent direction is a permutation matrix which is a vertex of the polytope of doubly stochastic matrices. For the convex, indefinite, and concave relaxations, this corresponds to a LAP with the gradient as the cost function. The second step performs a line search along the ascent direction to optimize the relaxed objective function. As the objectives are all quadratic, this line search simply

requires optimizing a single-variable quadratic function along a line segment. After the iterative algorithm converges, the final step of the procedure is to project the doubly stochastic matrix back to the set of permutation matrices, which is also a LAP.

The various relaxed forms can all serve as the objective function $f(\cdot)$ in the FW Methodology, but in all cases a matrix $D^0 \in \mathcal{D}$ must be chosen to initialize the procedure. For the convex relaxation, the FW methodology is guaranteed to converge to the global optimum regardless of D^0. On the other hand, the FW algorithm for the indefinite relaxation is not guaranteed to find a global optimum so the initialization is critical.

In many instances, the optimal solution to the convex relaxation lies in the interior of $\mathcal{D}$. This can lead to inaccurate solutions after the last projection step. The local optima for the indefinite relaxation are often at extreme points of $\mathcal{D}$, meaning the final projection often does nothing.

The default initialization for the indefinite problem is at the barycenter matrix, $D^0 = \frac{1}{n}\mathbb{1}\mathbb{1}^\top$, but many other initialization procedures can be used. These include randomized initializations, initializations based on similarity matrices, and initializing the indefinite relaxation at the interior point solution of the convex relaxation (Aflalo et al., 2015b). When prior information regarding a partial correspondence is known to be noisy, rather than incorporating this information as seeds, one can incorporate it as "soft" seeds which are used to generate the initialization (Fang et al., 2018).

When prior information is available in the form of seeds, the seeded graph matching problem (Lyzinski et al., 2014) works on the objective function (2.4) with the permutation matrix $P^{n\times n}$ substituted by $I_s \oplus P^{(n-s)\times(n-s)}$, the direct

Algorithm 1 Frank-Wolfe Methodology

Input : A, B, doubly stochastic matrix D^0, tolerance ϵ
Output: permutation matrix P
Set $i = 0$;
while $\|D^i - D^{i-1}\|_F^2 \geq \epsilon$ **do**
  $P^i = \text{argmin}_{P \in \Pi} \text{trace} \nabla f(D^i)^\top P$;
  $D^{i+1} = \text{argmin}_{D \in \mathcal{D}} f(D)$ over line segment from D^i to P^i;
  $i = i + 1$;
end
Project D^i to the nearest P by maximizing $\text{trace}(P^\top D)$;

sum of an $s \times s$ identity matrix and an $(n-s) \times (n-s)$ permutation matrix. Employing the indefinite relaxed objective function incorporating seeds, we formulate the problem as finding

$$\hat{P} = \underset{P \in \mathcal{D}}{\text{argmax}}\, 2 \cdot \text{trace} P^\top A_{21} B_{21}^\top + \text{trace} A_{22} P B_{22} P^\top$$

where the gradient to the objective function is

$$\nabla f(P) = 2 \cdot A_{21} B_{21}^\top + 2 \cdot A_{22} P B_{22}. \tag{2.6}$$

In total, this uses the information between seeded nodes and nonseeded nodes and the nonseed-to-nonseed information. Applying seeded graph matching to the convex relaxation and concave relaxation closely resembles the case of indefinite relaxation.

PATH algorithm Zaslavskiy et al. (2009) introduced a convex-concave programming approach to approximately solve the graph matching problem. The concave relaxation has the same solution as the original graph matching problem. The PATH algorithm finds a local optimum to the concave relaxation

by considering convex combinations of the convex relaxation $F_0(P)$ and the concave relaxation $F_1(P)$ denoted by $F_\lambda = (1-\lambda)F_0 + \lambda F_1$. Starting from the solution to the convex relaxation ($\lambda = 0$) the algorithm iteratively performs gradient ascent using the FW methodology at F_λ, increasing λ after each iteration, until $\lambda = 1$.

Percolation-based algorithms

Under the FW methodology, all the nodes admit a correspondence but the (relaxed) matching correspondence evolves through iterations. On the other hand, percolation approaches start with a set of seeds, adding one new match at each iteration. The new matches are fixed and hence not updated in future iterations.

Each iteration expands the set of matched nodes by propagating the current matching information to neighbors. The guiding intuition is that more matched neighbors are an indicator of a more plausible match, an intuition analogous to the gradient ascent approaches above. We will present two algorithms in this category where the ExpandWhenStuck algorithm is an extension to the Percolation algorithm.

There are some distinctions about the inputs and outputs of percolation methods compared to the above relaxation methods.

Percolation Algorithm Yartseva and Grossglauser (2013) provide a simple and fast approach to solve the graph matching problem by starting with a handful of seeds and propagating to the rest of the graphs. At each iteration, the matching information up to the current iteration is encoded in a subper-

mutation matrix P where $P_{ij} = 1$ if i is matched to j, and 0 otherwise. The PERCOLATION algorithm searches for the most promising new match among the unmatched pairs through the mark matrix, $M = APB$, which is the gradient of the indefinite relaxation when extended to sub-doubly stochastic matrices. When similarity scores are available, they are added to the mark matrix to combine topological structure and similarity scores.

Adopting analogous partitions on the adjacency matrices as in Eq. (2.1), we let A_{21}, B_{21} denote sub-matrix corresponding to potential adjacencies between unmatched and matched nodes. Since all the candidates of matched pairs are permanently removed from consideration, we need only consider $M' = A_{21}B_{21}^{\top}$, the sub-matrices of M corresponding to the unmatched nodes in both graphs. As a result, the PERCOLATION algorithm only uses matched-to-unmatched information to generate new matches.

Moreover, the mark matrix M can also be interpreted as encoding the number of matched neighboring pairs for each pair of nodes $i \in V_1$, $j \in V_2$. Suppose $u, u' \in V_1$, $v, v' \in V_2$, $[u, u'] \in E_1$ and $[v, v'] \in E_2$, then (u', v') is a neighboring pair of (u, v). In each iteration, while there remain unmatched nodes with more than r matched neighboring pairs, the percolation algorithm matches the pair of nodes with the highest score M_{uv}, and adds one mark to all the neighboring pairs of (u, v). Note that the algorithm may stop before all nodes are matched, leading to the return of a partial match.

There is only one tuning parameter in the PERCOLATION algorithm, the threshold r which controls a tradeoff between quantity of matches and quality of matches. With a small threshold, the algorithm has a larger chance of matching wrong pairs. If r is larger, then the algorithm might stop before

matching many pairs (Kazemi et al., 2015).

The PERCOLATION algorithm can be generalized to matching weighted graphs by making an adjustment to how we measure the matching information from the neighbors. Since we prefer to match edges with smaller weight differences and higher absolute weights, we propose to adopt the following update formula for the score associated with each pair of nodes $(i, j), i \in V_1, j \in V_2$:

$$M_{ij} = M_{ij} + \sum_{u \in N(i)} \sum_{v \in N(j)} 1 - \frac{|w_{iu} - w_{jv}|}{\max(|w_{iu}|, |w_{jv}|)}.$$

Thus, the score contributed by each neighboring pair of (i, j) is a number in $[0, 1]$.

Algorithm 2 Percolation Algorithm

Input : A, B, s pairs of seeds, threshold r
Output: (sub-)permutation matrix P
Initialize the sub-permutation matrix P incorporating seeds;
Calculate the mark matrix $M = APB$;
Set the rows and columns of M corresponding to seeds to minus infinity;
while $max(M) \geq r$ **do**
    $P_{ij} \leftarrow 1$, where $[i, j]$ is the index of $max(M)$;
    $M \leftarrow APB$;
    Set i^{th} row and j^{th} column of M to minus infinity;
end

ExpandWhenStuck Algorithm Kazemi et al. (2015) extends the PERCOLATION algorithm to a version that can operate with a smaller number of seeds. Without enough seeds, when there are no more unmatched pairs with a score higher or equal to the threshold r, the PERCOLATION algorithm would stop even if there are still unmatched pairs. EXPANDWHENSTUCK uses all pairs of nodes with at least one matched neighboring pair, $M_{ij} \geq 1$, as new

seeds to restart the matching process by adding one mark to all of the new seeds' neighboring pairs, without updating the matched set. If the updated mark matrix consists of new pairs with marks greater or equal to r, then the percolation algorithm continues, leading to larger matched sets.

Spectral-based algorithm

Another class of graph matching algorithms uses the spectral properties of adjacency matrices.

IsoRank Algorithm Singh et al. (2008) propose the IsoRank algorithm that uses neighboring topology and similarity scores and exploits spectral properties of the solution. The IsoRank algorithm is also based on the relaxation-based algorithms by encoding the topological structure of two graphs in ADB, which is again proportional to the gradient of the indefinite relaxation. However, the representations of each term of ADB are slightly different. A and B are the column-wise normalized adjacency matrices and D is not necessarily a doubly stochastic matrix yet D_{ij} still indicates how promising it is to match $i \in V_1$ to $j \in V_2$.

Similar to the idea of Percolation algorithm, the intuition is that the impact of a pair of matched nodes is evenly distributed to all of their neighbors to propagate plausible matches. This is achieved by solving the eigenvalue problem

$$\text{vec}(D) = (A \otimes B)\text{vec}(D), \tag{2.7}$$

where $\text{vec}(D)$ denotes the vectorization of matrix D, and the right hand side is

equivalent to ADB. To combine network-topological structure and similarity scores in the objective function, the normalized similarity score E is added to the right hand side of Equation 2.7, where $E = S/\|S\|_1$, and $\|\cdot\|_1$ denotes the L1 norm.

Note that when similarity score is not available as prior information, we can also construct a doubly stochastic similarity score matrix from seeds by taking $I_{s\times s} \oplus \frac{1}{n-s}\mathbb{1}\mathbb{1}^\top_{(n-s)\times(n-s)}$. To solve the eigenvalue problem in Equation 2.7, we resort to the power method. Finally, the global alignment is generated by a greedy algorithm or using the algorithms for solving the linear assignment problem (LAP).

Algorithm 3 IsoRank Algorithm

Input : A, B, similarity scores S
Output: permutation matrix P
Column-wise normalize the adjacency matrices A and B;
Normalize similarity scores: $E = S/|S|_1$;
Initialize $D^0 = E$ and tolerance ϵ;
while $|D^i - D^{i-1}| \geq \epsilon$ **do**
    Calculate $D^{i+1} = AD^iB + E$;
    Normalize $D^{i+1} = D^{i+1}/|D^{i+1}|$;
    $i = i + 1$;
end
Extract node mapping from $\hat{D}$ using a greedy method or by solving a LAP;

Umeyama algorithm Umeyama (1988) is a spectral approach to find approximate solutions to the graph matching problem. Assuming eigendecompositions of adjacency matrices A and B as $A = U_A\Lambda_A U_A^\top$ and $B = U_B\Lambda_B U_B^\top$, let $|U_A|$ and $|U_B|$ be matrices which takes absolute values of each element of U_A and U_B. Such modification to the eigenvector matrices guarantees the uniqueness of eigenvector selection. The global mapping is obtained by minimizing

the differences between matched rows of U_A and U_B:

$$\hat{P} = \underset{P \in \Pi}{\operatorname{argmin}} \| |U_A| - P|U_B| \|_F = \underset{P \in \Pi}{\operatorname{argmax}} \operatorname{trace}(|U_B| |U_A|^\top P)$$

The UMEYAMA algorithm can be generalized to matching directed graphs by eigendecomposing the Hermitian matrices E_A and E_B derived from the asymmetric adjacency matrices of the directed graphs. The Hermitian matrix for the adjacency matrix A is defined as $E_A = A_S + iA_N$, where $A_S = (A + A^\top)/2$ is a symmetric matrix, $A_N = (A - A^\top)/2$ is a skew-symmetric matrix and i is the imaginary unit. Similarly, we can define the Hermitian matrix for B. Assume the eigendecompositions of E_A and E_B as follows:

$$E_A = W_A \Gamma_A W_A^*, \quad E_B = W_B \Gamma_B W_B^*$$

and we aim at searching for:

$$\hat{P} = \underset{P \in \Pi}{\operatorname{argmax}} \operatorname{trace}(|W_B| |W_A|^\top P)$$

Note that the UMEYAMA algorithm works on the condition that two graphs are isomorphic or nearly isomorphic.

Algorithm 4 Umeyama Algorithm

Input : A, B

Output: permutation matrix P

Compute the eigendecompositions of A and B: $A = U_A \Lambda_A U_A^\top$, $B = U_B \Lambda_B U_B^\top$;

Solve the LAP: $P = \operatorname{argmax}_{P \in \Pi} \operatorname{trace}(|U_B| |U_A|^\top P)$;

2.1.2 Auxiliary graph matching tools

Centering technique

Instead of encoding the non-adjacencies by zeros in the adjacency matrices, the centering technique (Sussman et al., 2020) assigns negative values to such edges. The first approach is encoding non-adjacent node-pairs as -1 with centered adjacency matrices $\tilde{A} = 2A - \mathbf{J}$ and $\tilde{B} = 2B - \mathbf{J}$, where $\mathbf{J}$ is a matrix of all ones. An alternative approach relies on modeling assumptions where the pair of graphs are correlated but do not share a global structure. We match $\tilde{A} = A - \Lambda_A$ and $\tilde{B} = B - \Lambda_B$, where Λ is an n-by-n matrix with ij-th entry denoting an estimated marginal probability of an edge. In general, Λ is unknown but there are methods in the literature to estimate Λ.

Matching centered graphs changes the rewards for matching edges, non-edges, and the penalties for mismatches. Adapting the centering technique for the problem at hand can be used to find specific types of correspondences. This can also be combined with constructing multilayer networks out of single layer networks to match according to multiple criteria (Li and Sussman, 2019; Fan et al., 2020). The centering technique can be applied to any of the implemented graph matching algorithm. It is especially useful when padding graphs with differing numbers of vertices to distinguish isolated vertices from padded vertices.

Padding graphs of different orders

Until this section, we have been considering matching two graphs whose vertex sets are of the same cardinality. However, matching graphs with different orders are commonly seen in real-world problems.

Suppose $A \in \{0,1\}^{n \times n}$ and $B \in \{0,1\}^{n_c \times n_c}$ with $n_c < n$. One can then pad the smaller graph with extra vertices to match the order of the larger graph, $\tilde{B} = B \oplus \mathbf{0}_{n-n_c}$ and match A and $\tilde{B}$. Every implemented graph matching algorithm in the `iGraphMatch` package automatically handles input graphs with a different number of vertices using sparse padding with minimal memory impact.

Since the isolated vertices and the padded vertices share the same topological structure, it can be useful to center the original graphs first then pad the smaller graph in the same manner. This approach serves to differentiate between isolated vertices the padded ones. It's theoretically verified that in the correlated Erdős-Rényi graph model, the centered padding scheme is guaranteed to find the true correspondence between the nodes of G_1 and the induced subgraph of G_2 under mild conditions even if $|V_1| \ll |V_2|$, but the true alignment is not guaranteed without centering (Sussman et al., 2020).

Exploiting sparse and low-rank structure

Many real-world graphs, especially large graphs, are often very sparse with $o(n^2)$ and often $\theta(n)$ edges. This can increase the difficulty of the graph matching problem due to the fact that there are fewer potential edges to match, but sparse graphs also come with computational advantages. We rely on `igraph` and `Matrix` for efficient storage of these structures as well as the efficient implementation of various matrix operations. We also use the LAPMOD algorithm for sparse LAP problems (Volgenant, 1996) (see below).

Similarly, a low-rank structure appears in some of the procedures including starting at the rank-1 barycenter matrix and the different centering schemes.

Since low-rank matrices are generally not sparse and visa-versa we implemented the splr S4 class, standing for sparse plus low-rank matrices. This class inherits from the Matrix class and includes slots for an $n \times n$ sparse matrix x and $n \times d$ dense matrices a and b, to represent matrices of the form x + tcrossprod(a, b). This class implements efficient methods for matrix multiplication and other operations that exploit the sparse and low-rank structure of the matrices. Specifically, these methods often require only $O(\|x\|_0) + O(nd)$ storage as opposed to $O(n^2)$ required for densely stored matrices, and enjoy analogous computational advantages. While users can also use these matrices explicitly, most use of them is automatic within functions such as init_start and center_graph and the matrices can largely be used interchangeably with other matrices.

LAP methods

Multiple graph matching methods include solving an LAP and so we have included multiple methods for solving LAPs into the package. Specifically we implement the Jonker-Volgenant algorithm (Jonker and Volgenant, 2005) for dense cost matrices and the LAPMOD algorithm (Volgenant, 1996) for sparse cost matrices. Both algorithms are implemented in C to provide improved performance. The LAPMOD approach is typically advantageous when the number of non-zero entries is less than 50We also depend on the `clue` package for the solve_LSAP function which implements the Hungarian algorithm (Papadimitriou and Steiglitz, 1998) for solving an LAP. Each of these methods can be used independently of a specific graph matching method using the do_lap function.

Multi-layered graph matching

Frequently, networks edges may have categorical attributes and from these categories, we can construct multilayer graphs (Kivelä et al., 2014), where each layer in the networks contains edges from specific categories. For matching two multilayer graphs, the standard graph matching problem can be extended as $\sum_{l=1}^{m} \|A^{(l)} - PB^{(l)}P^\top\|_F^2$ where m denotes the number of categories and $A^{(l)}, B^{(l)}$ are the adjacency matrices for the lth layers in each graph. Note, we assume that the layers are aligned, so that layer l corresponds to the same edge-types in both multi-layer networks.

For an `igraph` object, the function split_igraph can be used to convert a single object with categorical edge attributes into a list with each element only containing the edges with a specific attribute value. The implemented algorithms can seamlessly match multi-layer graphs, which are encoded as a list of either igraph objects or matrix-like objects. We also implemented a matrixlist S4 class that implements many standard matrix operations so that algorithms can be easily extended to work with multilayer networks.

2.1.3 Correlated random graph models

The correlated Erdős-Rényi model (Lyzinski et al., 2016) is essential in the theoretical study of graph matching algorithms. In a single Erdős-Rényi graph, each edge is present in the graph independently with probability p. The correlated Erdős-Rényi model provides a joint distribution for a pair of graphs, where each graph is marginally distributed as an Erdős-Rényi graph and corresponding edge-pairs are correlated.

Definition 1 (Correlated Erdős-Rényi Model)**.** *Suppose* $p \in (0, 1)$ *and* $\rho \in$

$[\max\{\frac{-p}{1-p}, \frac{p-1}{p}\}, 1]$, *a pair of adjacency matrices* $(A, B) \sim CorrER(pJ, \rho J)$ *if:*

- *For each* $1 \leq u \leq v \leq n, A_{uv}$ *are independent with* $A_{uv} \sim Bernoulli(p)$.
- *For each* $1 \leq u \leq v \leq n, B_{uv}$ *are independent with* $B_{uv} \sim Bernoulli(p)$.
- A_{uv} *and* $B_{u'v'}$ *are independent unless* $u = u', v = v'$ *in which case the Pearson correlation* $\text{corr}(A_{uv}, B_{uv}) = \rho$.

To sample a pair of correlated Erdős-Rényi graphs with edge probability p, and Pearson correlation ρ, we first sample three independent Erdős-Rényi graphs G_1, Z_0 and Z_1 with edge probabilities p, $p(1-\rho)$ and $p + \rho(1-p)$ respectively. Let $G_2 = (Z_1 \cap G_1) \bigcup (Z_0 \cap G_1^c)$.

Yartseva and Grossglauser (2013) provide an alternative formulation for the correlated Erdős-Rényi model $CorrER(n, p', s')$. First, one samples a single random Erdős-Rényi graph G with edge probability p'. Conditioned on G, each edge in G is present independently in G_1, G_2 with probability s'. These two parameterizations are related to each other by the relationship $s' = p+\rho(1-p)$ and $p' = p/(p+\rho(1-p))$. The original parameterization is slightly more general because it allows for the possibility of negative correlation.

In addition to homogeneous correlated Erdős-Rényi random graphs, we also implement heterogeneous generalizations of this model. The stochastic block model (Holland et al., 1983) and the random dot product graphs (RDPG) model (Young and Scheinerman, 2007) can both be regarded as extensions of the Erdős-Rényi model. The stochastic block model is useful to represent the community structure of graphs by dividing the graph into K groups. Each node is assigned to a group and the probability of edges is determined by the group memberships of the vertex pair. For the RDPG model, each vertex is

assigned a latent position in $\mathbb{R}^d$ and edge probabilities are given by the inner product between the latent positions of the vertex pair.

For both of these models, we can consider correlated graph-pairs where marginally they arise from one of these models and again corresponding edge pairs are correlated.

2.1.4 Measures for goodness of matching

The ability to assess the quality of the match when ground truth is unavailable is critical for the usage of the matching approaches. There are various topological criteria that can be applied to measure the quality of matching results. At the graph level, the most frequently used structural measures include matching pairs (MP), edge correctness (EC), and the size of the largest common connected subgraph (LCCS) (Kuchaiev and Przulj, 2011). MP counts the number of correctly matched pairs of nodes, thus can only be used when the true alignment is available. Global counts of common edges (CE) and common non-edges (CNE) can be defined as

$$CE = \frac{1}{2}\sum_{i,j} \mathbb{1}\{A_{ij} = PBP^{\top}_{ij} = 1\}$$
$$CNE = \frac{1}{2}\sum_{i,j} \mathbb{1}\{A_{ij} = PBP^{\top}_{ij} = 0\},$$

along with error counts such extra edges (EE) and missing edges (ME),

$$EE = \frac{1}{2}\sum_{i,j} \mathbb{1}\{A_{ij} = 0 = 1 - PBP^{\top}_{ij}\}$$
$$ME = \frac{1}{2}\sum_{i,j} \mathbb{1}\{A_{ij} = 1 = 1 - PBP^{\top}_{ij}\},$$

EC measures the percentage of correctly aligned edges, that is the fraction

CE/$|E_1|$. The LCCS denotes the largest subset of aligned vertices such that the corresponding induced subgraphs of each graph are connected. Matches with a larger LCCS are often preferable to those with many isolated components.

Another group of criteria measures the goodness of matching at the vertex level. Informally, we aim at testing the hypotheses

$$H_0^{(v)} : \text{the vertex } v \text{ is not matched correctly by } P^*,$$
$$H_a^{(v)} : \text{the vertex } v \text{ is matched correctly by } P^*$$

for each vertex v.

The goal is to address if the permutation matrix found by graph matching algorithm is significantly different from the one sampled from uniformly distributed permutation matrices (Lyzinski and Sussman, 2020). Unfortunately, vertex-level matching criteria have only received limited attention in the literature, however, we include two test statistics to measure fit. The row difference statistic is the L_1-norm of the difference between A and $P^*BP^{*\top}$, namely

$$T_d(v, P^*) := \|A_{v\cdot} - (P^*BP^{*\top})_{v\cdot}\|_1.$$

Intuitively, a correctly matched vertex v should induce a smaller $T_d(v, P^*)$, which for unweighted graphs corresponds to the number of edge disagreements induced by matching v. Alternatively, the row correlation statistic is defined as

$$T_c(v, P^*) := 1 - corr(A_{v\cdot}, (P^*BP^{*\top})_{v\cdot}).$$

We expect the empirical correlation between the neighborhoods of v in A and $P^*BP^{*\top}$ to be larger for a correctly matched vertex.

We employ permutation testing ideas to the raw statistics as a normalization across vertices. Let us take the row difference statistic for example. The guiding intuition is that if v is correctly matched, the number of errors induced by P^* across the neighborhood of v in A and B (i.e., $T_d(v, P^*)$) should be significantly smaller than the number of errors induced by a randomly chosen permutation P (i.e., $T_d(v, P)$).

With this in mind, let $\mathbb{E}_P$ and Var_P denote the expectation and variance of the raw statistic with P uniformly sampled over all permutation matrices. The normalization is then given by

$$T_p(v, P^*) := \frac{T(v, P^*) - \mathbb{E}_P T(v, P)}{\sqrt{Var_P T(v, P)}}$$

where $T(v, P)$ can be either of the two test statistics we introduced earlier.

In addition to measuring match quality, these vertex-wise statistics can also serve as a tool to find which vertices have no valid match in another network, i.e., the vertex entity is present in one network but not the other.

2.2 Usage examples

In this section, we demonstrate graph matching analysis using `iGraphMatch` via examples on real datasets, including communication networks, neuronal networks, and transportation networks. Table 2.2 presents brief overviews of the first two datasets. Note that the number of edges doesn't consider weights for weighted graphs, and for directed graphs, an edge from node i to node j and another edge from j to i will be counted as two edges. Tables 2.3 and 2.7 summarize the edge correspondence between two graphs under the true

alignment including the number of common edges, missing edges, and extra edges in two graphs.

Table 2.2: Overview of the Enron and *C.Elegans* graphs.

Dataset	# Nodes	# Edges	Correlation	Weighted	Directed
Enron	184	488 / 482	0.85	No	Yes
C.Elegans	279	2194 / 1031	0.10	Yes	Yes

In the first Enron email network example, we demonstrate the usage of Frank-Wolfe methodology and how to improve matching performance by using the centering technique and incorporating adaptive seeds. In the second example using *C.Elegans* synapses networks, we illustrate how to use soft matching for a challenging graph matching task using Frank-Wolfe methodology, PATH algorithm and IsoRank algorithm. Finally, we include an example of matching two multi-layer graphs with similarity scores on the Britain transportation networks.

Table 2.3: Edge summary under the true alignments of the Enron and *C.Elegans* graphs.

Dataset	Common edges	Missing edges	Extra edges
Enron	412	76	70
C.Elegans	116	981	400

2.2.1 Example: Enron Email Network Data

The Enron email network data was originally made public by the Federal Energy Commission during the investigation into the Enron Corporation (Leskovec et al., 2009). Each node of the Enron network represents an email address and if there is at least one email sent from one address to another address, a directed edge exists between the corresponding nodes.

The `iGraphMatch` package includes the Enron email network data in the form of a pair of igraph objects derived from the original data where each graph represents one week of emails between 184 email addresses. The two networks are unweighted and directed with edge densities around 0.01 in each graph and the empirical correlation between two graphs is 0.85. Figure 2·1 visualizes the aligned full networks using the visualization tool in the package with vertices sorted by a community detection algorithm (Clauset et al., 2004) and degree. Grey, blue, and red pixels indicate common edges, missing edges, and extra edges respectively.

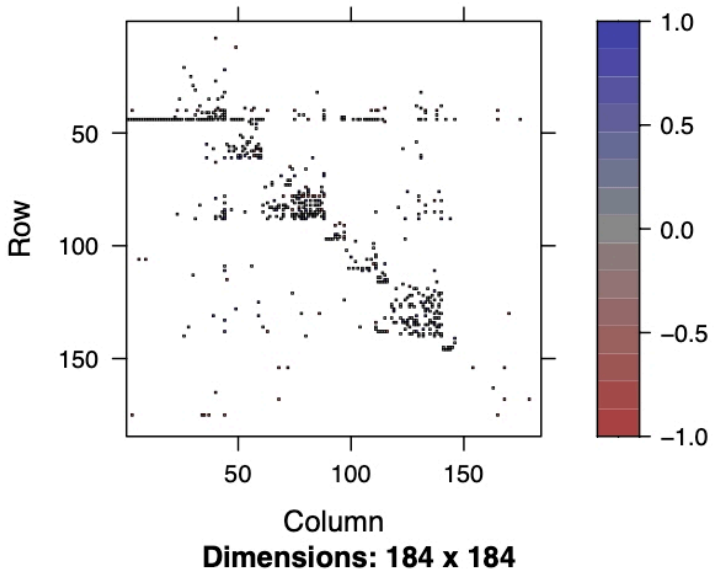


Figure 2·1: Asymmetric adjacency matrices of aligned Enron Corporation communication networks.

Note that 37 and 32 out of the total 184 nodes are isolated from the other nodes in two graphs respectively, indicating the corresponding employees haven't sent or received emails from other employees. This adds difficulty

to matching since it's impossible to distinguish the isolated nodes based on topological structure alone. In this example, we focus on the largest connected component of each graph. The sizes of the largest connected components of the two graphs are 146 and 151, which are different.

Matching largest connected components with the centering technique

Let's assume the Enron email communication network from the second week is anonymous, and we aim at finding an alignment between the email addresses from the first network and the second one to de-anonymize the latter. Additionally, we want to find the email addresses that are active in both months.

Suppose no prior information on partial alignment is available in this example. We match the two largest connected components using the FW algorithm with indefinite relaxation since seeds and similarity scores are not mandatory for this method. We assign equal likelihood to all the possible matches by initializing at the barycenter. Since two graphs are of different sizes, the gm function automatically pads the smaller graph with extra 0's. We can try to improve performance by centering the larger graph by assigning -1 to non-edges, so that we penalize edges that are missing in the larger graph but present in the smaller one.

Table 2.4: Summary of matching with the centering technique.

Centering	Common edges	Missing edges	Extra edges	Objective function
No	353	134	128	16
Yes	399	88	82	13

Assuming that there is no information on the true correspondence, in this

example, we can evaluate the matching result based on statistics matched nodes, matched edges and the graph matching objective function. From table 2.4, without any seeds or similarity scores, around 72% of edges are correctly matched using the FW algorithm on the raw Enron networks. Additionally, we would prefer matching Enron networks with the application of the centering scheme, since we get more matched common edges, as well as fewer missing edges and extra edges.

Matching with adaptive seeds

Since seeded graph matching enhances the graph matching performance substantially (Lyzinski et al., 2014), it may be useful to use some of these best matches as seeds to improve matching results. In the `iGraphMatch` package, we implement the best_matches function to measure and rank the vertex-wise matching performance. Here, we use adaptive seeds, taking the ns best matches and using them as additional prior information in a second run of the matching algorithm.

Firstly, let's apply the hard seeding approach, where ns best matches are used as seeds in a second run of the matching algorithm. Table 2.5 displays edge statistics and objective function values for different number of adaptive seeds used. The second column in the table shows the matching precision of the adaptive seeds based on ground truth. Incorporating adaptive seeds and repeating the FW matching procedure on centered graphs further improve the matching results, compared with the case without any adaptive seeds when $ns = 0$. The first 40 pairs of matched nodes ranked by best_matches function are all correctly matched, and this is also when matching is improved the most

by the hard seeding approach.

Table 2.5: Summary of matching with adaptive seeds.

ns	Precision	Hard seeding			Soft seeding		
		Common	Missing	Extra	Common	Missing	Extra
0	NaN	394	93	87	394	93	87
20	1.00	402	85	79	**410**	**77**	**71**
40	1.00	410	77	71	410	77	71
60	0.97	400	87	81	400	87	81
80	0.92	402	85	79	409	78	72

However, as the number of adaptive seeds increases, the precision of adaptive seeds decreases. Note that if they are treated as hard seeds, incorrect matches will remain in the matched set and might cause a cascade of errors. An alternative way is to treat the top-ranked matches as soft seeds embedded in the start matrix to handle the uncertainty. In this way, adaptive seeds not only provide prior information but also evolve over iterations. Table 2.5 shows that the soft seeding approach always outperforms or performs as good as the hard seeding approach regardless of the number of adaptive seeds being used.

Core vertices detection

The function best_matches can also be used to detect core vertices. Suppose the ground truth is known and that the first 145 vertices are core vertices. The mean precision of detecting core vertices and junk vertices using best_matches function is displayed in Figure 2·2. A lower rank is a stronger indicator of a core vertex and a higher rank is a stronger indicator of a junk vertex. Let $r_i^C, 1 \leq i \leq n_c$ and $r_j^J, 1 \leq j \leq n_j$ denote the ranks associated with each core vertex and each junk vertex. The figure shows the precision of identifying core vertices at each low rank r, i.e., $\frac{1}{r}\sum_{i=1}^{n_c} \mathbb{1}_{r_i^C \leq r}$, and the precision of identifying junk vertices at each high rank r, i.e., $\frac{1}{r}\sum_{j=1}^{n_j} \mathbb{1}_{r_j^J \geq n_c+n_j-r}$, which are separated

by the vertical lines.

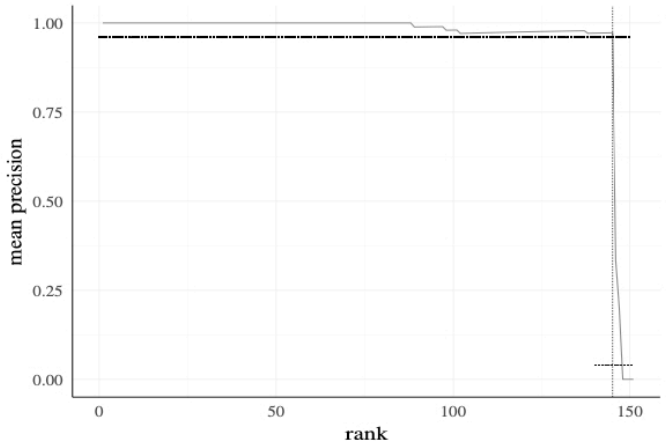


Figure 2·2: Mean precision for identifying core and junk vertices for the Enron networks by using the row permutation test.

Core detection performance is substantially better than chance, as represented by the dotted horizontal lines. The top 88 are all core vertices indicating good overall performance for core identification. For junk identification, the junk vertices are ranked 63, 54, 50, 14, 6, 5 according to which have the lowest score, indicating that some junk vertices are difficult to identify.

2.2.2 Example: *C.Elegans*

The *C.Elegans* networks consist of the chemical synapses network and the electrical synapses network of the roundworm, where each of 279 nodes represents a neuron and each edge represents the intensity of synapse connections between two neurons (Chen et al., 2016). Matching the chemical synapses network to the electrical synapses network is essential for understanding how the brain functions. These networks are quite sparse with edge densities of

0.03 and 0.01 in each graph and the empirical correlation between two graphs is 0.1.

A challenging task

For simplicity, we made the networks unweighted and undirected for the experiments, and we assume the ground truth is known to be the identity. Matching the *C.Elegans* networks is a challenging task. Figures 2·3 and 2·4 depict the edge discrepancies of two networks under the true alignment and the matching correspondence using FW algorithm initialized at the true alignment. Green pixels represents an edge in the chemical graph while no edge in the electrical graph. Red pixels represent only an edge in the electrical graph. Grey pixels represent there is an edge in both graphs and white represents no edge in both graphs. The alignment found using FW is not the identity with 124 out of 279 nodes correctly matched and improves upon the identity in terms of the number of edge discrepancies. For the true alignment, there are 116 edge errors and 1380 common edges while the alignment yielded by FW initialized at the true correspondence has 267.5 edge errors and 1078 common edges. Hence, this graph matching object does not have a solution at the true alignment. One can try to use other objective functions to enhance the matching result, however we do not investigate this here. Overall, while most performance measures are poor, our results illustrate the spectrum of challenges for graph matching.

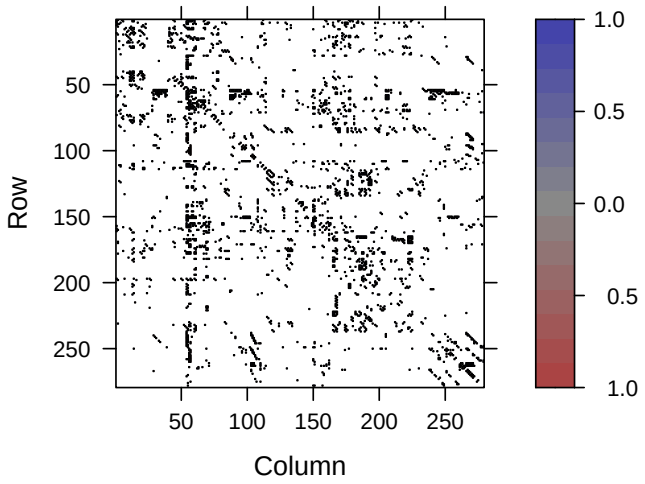


Figure 2·3: Edge discrepancies for the matched graphs with the true correspondence.

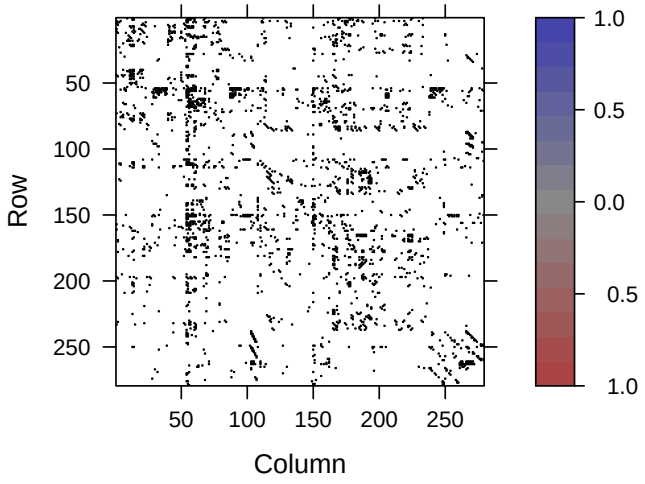


Figure 2·4: Edge discrepancies for the matched graphs using FW algorithm starting at the true correspondence.

Soft matching: MAP@3

Considering matching *C.Elegans* graphs is quite challenging, let's assume 20 pairs of vertices are known as seeds, which are chosen at random. Accordingly, we generate a similarity matrix with 1's corresponding to seeds, and the rest being barycenter. In addition to one-on-one matching, we will also conduct soft matching, which is to find three most promising matches to each non-seed vertex. We achieve the goal of soft matching by finding the top 3 largest values in each row of the doubly stochastic matrix from the last iteration of Frank Wolfe methodology with indefinite relaxation and PATH algorithm, as well as the normalized matrix from the last iteration of the power method for IsoRank algorithm. To evaluate the matching performance, we will look at both matching precision: $precision = \frac{1}{n_m - s} \sum_{i \in V_m \setminus S} P_{ii}$, and Mean Average Precision @ 3 (MAP@ 3): $MAP@3 = \frac{1}{n_m - s} \sum_{i \in V_m \setminus S} \mathbb{1}_{\{i \in T_i\}}$, where T_i is the set of 3 most promising matches to node i.

Table 2.6: Soft matching on the challenging *C.Elegans* graphs.

Metric	Frank Wolfe	PATH	IsoRank
Precision	0.0896	0.086	0.0789
MAP@3	0.0968	0.1111	0.0824

As shown in the table above, MAP@ 3 is slightly higher than precision for each method. Soft matching provides an alternative way of matching by generating a set of promising matching candidates.

2.2.3 Example: Britain Transportation

To demonstrate matching multi-layer networks-layers, we consider two graphs derived from the Britain Transportation network (Riccardo and Marc, 2015).

The network reflects the transportation connections in the UK, with five layers representing ferry, rail, metro, coach, and bus. A smaller template graph was constructed based on a random walk starting from a randomly chosen hub node, a node that has connections in all the layers. The template graph has 53 nodes and 56 connections in total and is an induced subgraph of the original graph.

Additionally, based on filter methods from Moorman et al. (2018a), the authors of that paper also provided a list of candidate matches for each template node, where the true correspondence is guaranteed to be among the candidates. The number of candidates ranges from 3 to 1059 at most, with an average of 241 candidates for each template vertex. Thus, we made an induced subgraph from the transportation network with only candidates, which gave us the world graph with 2075 vertices and 8368 connections.

Figure 2·5 visualizes the transportation connections for the induced subgraphs, where means of transportation are represented by different colors. Note that all edges in the template are common edges shared by two graphs, where 40%, 24.1%, 37.5%, 31.7%, and 25.6% of edges in the world graph are in template for each layer. All graphs are unweighted, directed, and do not have self-loops.

Table 2.7 further displays an overview and edge summary regarding each layer of the Britain Transportation Network. Correlation is calculted using the template graph and the aligned induced subgraph of the world graph. The final three columns indicate the number of common edges, missing edges, and extra edges in the aligned subgraph of the world graph.

A true correspondence exists for each template vertex in the world graph,

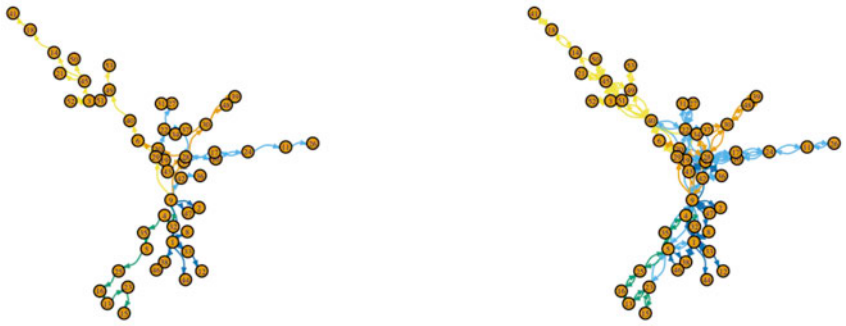


Figure 2·5: Visualization of the template graph (left) and the world graph (right) with corresponding vertices.

Table 2.7: Overview of the Britain Transportation Network layers.

Layer	Nodes	Edges	Correlation	Common	Missing	Extra
Ferry	53 / 2075	10 / 42	0.63	10	0	15
Rail	53 / 2075	14 / 4185	0.49	14	0	44
Metro	53 / 2075	9 / 445	0.61	9	0	15
Coach	53 / 2075	13 / 2818	0.56	13	0	28
Bus	53 / 2075	10 / 878	0.50	10	0	29

our goal is to locate each template vertex in the Britain Transportation network by matching two multi-layer graphs with different number of vertices.

Based on the candidates, we specify a start matrix that is row-stochastic which can be used for the `start` argument in the graph matching function for FW methodology. For each row node, its value is either zero or the inverse of the number of candidates for that node. To ensure that template nodes only get matched to candidates, we constructed a similarity score matrix by taking the start matrix $\times 10^5$, so that a high similarity score is assigned to all the template-candidate pairs.

Then we match the template graph with the world graph using PERCOLATION algorithm. The template graph and world graph are lists of 5 matrices of dimensions 53 and 2075 respectively. With no information on seeds, the

PERCOLATION algorithm initializes the mark matrix using prior information in the similarity score matrix.

The summary function in the package outputs edge statistics and objective function values for each layer separately. Table 2.8 displays the evaluation metrics as outputted by the summary function. To further improve matching performance, one can replicate all the analysis in the first example on Enron dataset, such as using the centering scheme and adaptive seeds. Finally, one can refer to the match report to compare matching performance and pick the best one.

Table 2.8: Summary of matching for each layer of Britain Transportation networks.

Layer	Common edges	Missing edges	Extra edges	Objective function
1	10	0	22	4.7
2	13	1	35	6.0
3	9	0	21	4.6
4	12	1	25	5.1
5	10	0	35	5.9

2.3 Matching trade-off analysis between efficiency and effectiveness

In this section, we will examine the performance of each graph matching algorithm with respect to different network structures, network scales, and the amount of available prior information in the form of seeds and similarity scores. We illustrate the trade-off between efficiency and effectiveness for the different graph matching algorithms on synthetic datasets and real datasets used in section 2.2. Note that in the rest of this section, anytime we select seeds or induce subgraphs we do this by selecting nodes uniformly at random, and

seeds are selected from the intersection of vertex sets of two graphs. Unless otherwise specified, in all the examples we generate similarity score matrices at the barycenter incorporating seeds, $I_{s\times s} \oplus \frac{1}{n-s}\mathbb{1}\mathbb{1}_{(n-s)\times(n-s)}$.

In addition to the goodness-of-match metrics introduced in section 2.1.4, we show the accuracy of matching non-seeds vertices, $\frac{1}{n-s}\sum_{i\in V\setminus S} P_{ii}$, the precision of matching non-seeds vertices, $\frac{1}{n_m-s}\sum_{i\in V_m\setminus S} P_{ii}$, where V is the set of all the vertices, V_m is the set of matched vertices, S is the set of seed vertices and P is the permutation matrix found by the algorithms. For some algorithms, these will be the same since a match is returned for each node, but for algorithms such as PERCOLATION the precision may vary substantially from accuracy. With both matching accuracy and matching precision, we can assess the performance of a graph matching algorithm with regard to both quantity and quality. In order to illustrate the computational trade-offs, we also plot the running time in seconds for the algorithms. The error bars in the figures indicate 95% intervals using the normal approximation.

2.3.1 Correlated Erdős-Rényi graphs

In this simulated data example, we will consider the graph matching problem for two correlated graphs from CorrER($pJ, \rho J$). The number of nodes in each graph are the same, with $n = 200, 400, 600, 800, 1000, 1200$ and 1400. Let two graphs be moderately correlated and set the edge correlations as $\rho = 0.5$. To investigate the impact of the density of graphs, we vary the average degree $d = 20, 100, 150$. Accordingly, the edge probability $p = d/n$. In each experiment, we randomly sample a pair of correlated Erdős-Rényi graphs and uniformly randomly choose 40 corresponding pairs of nodes as seeds. Figure

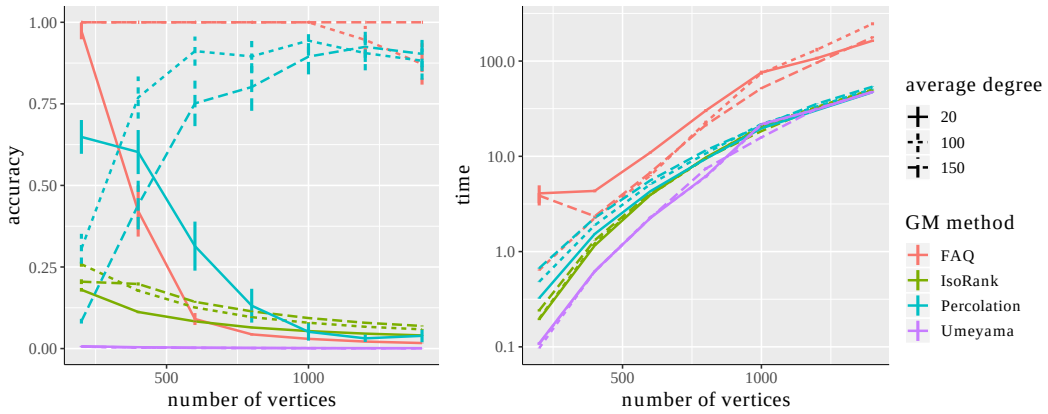


Figure 2·6: Matching performance for different graph matching algorithms versus graph size on correlated Erdős-Rényi graphs.

2·6 shows an average of 100 replicates for each value of n and d, and we plot the matching accuracy on the left and running time on the right versus the number of vertices in each graph.

For the sparse graphs case represented by the solid line, as the number of vertices grows which corresponds to decreasing edge probabilities, all the algorithms yield worse matching performances, with both lower matching accuracy and longer running time. Indeed, with too little information from the topological structrure, it's theoretically challenging to recover the true alignment with limited seeds. Among the different algorithms, FW matches the most correct pairs when the two graphs are small with less than 300 nodes at the cost of longer running time. When matching two large sparse graphs with $n > 300$, PERCOLATION outperforms the others by matching more nodes correctly in a shorter amount of time.

For matching denser graphs, as illustrated by the dashed lines, the FW algorithm always perfectly recovers the correspondence (except $d = 100$ and

$n = 1400$), but at a cost of a longer running time when graphs are large. With a more restricted computational costs, the PERCOLATION algorithm is much faster and has the second best performance. Also note that with fixed average degree $d = 100$ or 150, PERCOLATION has increasing matching accuracy for larger graphs, and finally shows a tendency of decreasing, indicating that PERCOLATION is preferable for matching relatively sparse density graphs.

The runtime plot on the right indicates the rank of the running times is FW $>$ PERCOLATION $\geq$ ISORANK $\geq$ UMEYAMA. Although ISORANK and UMEYAMA have advantages over the others in terms of speed, only a small proportion of nodes are matched correctly. Notably, in this moderate correlation case, UMEYAMA has particularly poor performance since the success of the algorithm highly depends on the isomorphism of two graphs. In the experiments with $\rho = 1$, when the graphs are isomorphic, UMEYAMA runs faster while recovers the true alignment as other algorithms as well.

2.3.2 Enron

In this illustrative example, we use the Enron network data as described in Section 2.2.1. We aim to match a randomly induced subgraph of the first week with a larger induced subgraph from the second week. In the first experiment, we aim to match a randomly induced subgraph of the first week with n_1 email addresses with a larger induced subgraph from the second week containing n_2 email addresses. Apart from varying the size of the larger network n_2, we also compared matching the original adjacency matrices A, B to matching A and the centered the matrix $2B - 1$. For the latter non-edges of the second graph have value -1 with all entry for a padded node of the first graph set to

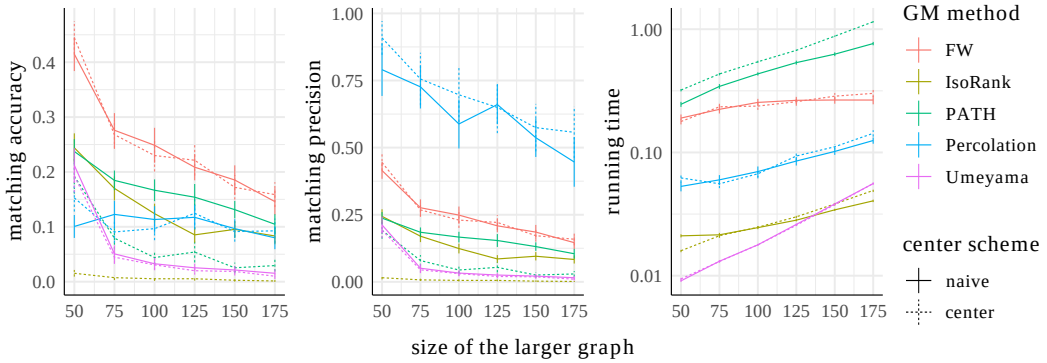


Figure 2·7: Matching accuracy, matching precision, and running time as a function of size of the larger induced subgraph on Enron dataset.

zero. By doing this, we penalize missing edges in A but there is no penalty for extra edges in B. For each combination of $n_2 \in \{50, 75, 100, 125, 150, 175\}$ and whether the centering scheme was applied, we ran 50 Monte Carlo replicates with the number of employees in the first network $n_1 = 50$ and the number of seeds $s = 20$ fixed.

Figure 2·7 shows that both matching accuracy and matching precision decrease as the size of the larger induced subgraph grows under almost all cases. The FW algorithm achieves the highest matching accuracy among all the algorithms at the expense of slightly longer running time. The PERCOLATION algorithm yields the highest matching precision with the highest quality set of matched vertices but it only matches a fraction of vertices in the smaller graph.

The matching precision for the PERCOLATION algorithm is significantly enhanced by deploying the centering scheme, shown by the gap between the blue solid line and blue dotted line in the middle plot. By centering the graphs, mismatches of edges to non-edges were penalized and isolated vertices in the

Enron networks were distinguishable from the padded ones. Also notice that for the PERCOLATION algorithm, centering the graphs doesn't increase the running speed by much.

2.3.3 *C.Elegans*

Although accurately aligning the *C.Elegans* networks is hard as shown in Section 2.2.2, it is still worthwhile to explore the relative performance of each graph matching algorithm by matching chemical synapses network and the electrical synapses network. For each $s \in \{20, 40, 60, 80, 100\}$, we match the chemical synapses network and the electrical network. In general, all the algorithms fail to correctly match most vertices, but the matching performance is significantly better than chance, as indicated by the pink lines in figure 2·8. Among the different methods, ISORANK yields the best performance in terms of matching accuracy, precision and running time, except for slightly lower matching precision than PERCOLATION when there are 100 seeds but notably, only a small number of vertices get matched. From the last column in Table 2.9, we can also see that PERCOLATION yields a much smaller LCCS than the other methods. The table also shows the edge correctness for two graphs under the alignment found by GM methods, where FW yields the highest edge correctness for both graphs. As illustrated above by figures 2·3 and 2·4, vertex alignment performance for *C.Elegans* does not correspond well to measures of edge-correctness.

In light of the fact that the overall matching performance is relatively poor, we turn our attention to the accuracy within top-ranked matches. After matching the *C.Elegans* networks with 60 uniformly chosen seeds, we ranked

Table 2.9: Edge correctness and size of LCCS for different GM methods for *C.Elegans* networks example.

GM Method	Edge Correctness G_1	Edge Correctness G_1	LCCS
FW	0.20	0.74	248
IsoRank	0.13	0.50	248
PATH	0.10	0.38	248.
Percolation	0.11	0.44	131

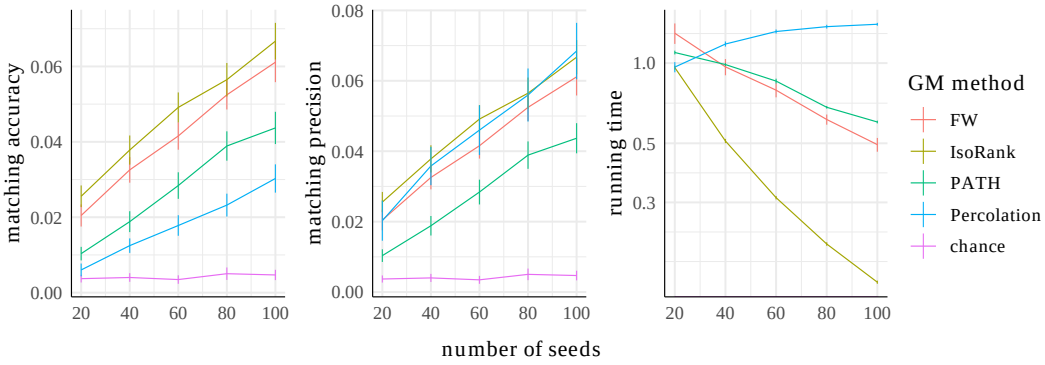


Figure 2·8: Matching accuracy, matching precision, and running time as a function of number of seeds on *C.Elegans* dataset.

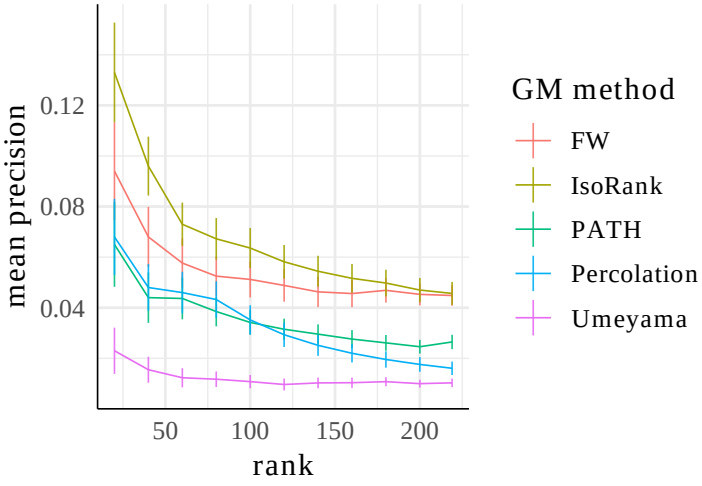


Figure 2·9: Mean precision as a function of varying number of top matches on *C.Elegans* dataset.

the matched pairs using the row permutation statistics for row difference measure. Finally, we computed the matching precision for the top-ranked matches. Figure 2·9 suggests that IsoRank outperforms the other algorithms in coming up with a small set of matched pairs with higher precision.

2.3.4 Britain Transportation

Again, we resort to the Britain Transportation multi-layered networks with more details of the networks elaborated in Section 2.2.3. We are still interested in matching the template graph and the world graph, but with the focus on comparing the different graph matching algorithms implemented in the `iGraphMatch` package including the FW, Percolation, ExpandWhenStuck, IsoRank and PATH algorithms. For the FW methodology with indefinite relaxation, we adopt the same specification of the start matrix as in Section 2.2.3. When matching the transportation networks using the Percolation and the ExpandWhenStuck algorithms, since there were no hard seeds we only inputted the similarity score. Also, we considered applying the centering scheme on the world graph so that we not only rewarded matched edges but also penalized mismatched edges. We also tried to only center the template graph and center both graphs for comparison, but only centering the world graph yielded the most appealing performance.

A summary of the matching results is displayed in table 2.10 including the number of correctly matched nodes, the number of true signals in the world graph that are in the matched set, regardless of if they are correctly matched or not, and running time in seconds. The PATH algorithm and ExpandWhenStuck algorithm did not have competitive matching performance so

the results are omitted. FW without centering the world graph outperforms the other methods in terms of all three criteria, and we found that templates with fewer candidates are more likely to be correctly matched. Notably, except for the FW algorithm, other algorithms achieved better matching performance by centering the world graph, either matched more template nodes correctly or extracted more core vertices, or both at the expense of longer running time.

Table 2.10: Matching performance comparison for matching two induced Britain Transportation networks.

GM Method	Center	True	# Recall	Running time (s)
FW	No	9	17	2.99
FW	Yes	8	16	14.68
IsoRank	No	3	15	5.54
IsoRank	Yes	5	14	8.95
Percolation	No	6	13	5.89
Percolation	Yes	8	17	38.55

Table 2.10 demonstrates the matching performance of each method by showing the result of one experiment which is representative of performance on average. To study the variability of each method on this dataset, we ran multiple replicates with different random number generators. FW and IsoRank performs in a rather stable manner while the performance of Percolation varies the most depending on the random number generator used. Although FW outperforms the other algorithms on average, sometimes Percolation can match much more nodes correctly than the others.

Chapter 3

Graph matching via mutual nearest neighbors

In this chapter, we introduce the Mutual Nearest Neighbor (MNN) algorithm, which starts with a set of seeds, and maps node pairs only if they are more promising than all the other candidate pairs for each node, with the matched set evolving over iterations.

Principal advantages of the proposed MNN algorithm include (a) MNN can match a subset of high-quality node pairs which can serve as additional prior information in subsequent GM analysis, (b) MNN can potentially remove matches if there is not sufficient evidence for the match, (c) MNN requires no hyperparameter, (d) MNN has a computational complexity of $O(n^2)$ for each iteration, compared with $O(n^3)$ for the FW methodology and can easily exploit parallel operations. In the rest of the chapter, we will describe the proposed algorithm, and illustrate its properties through its empirical performance and theoretical properties.

3.1 GM algorithms using seeded neighboring statistics

Before introducing the MNN algorithm, let us first review the related seeded GM algorithms that propagate mappings with neighborhood statistics.

Among the GM algorithms implemented in the `iGraphMatch` package, the Percolation algorithm utilizes neighboring topology and grows a mapping from a set of seeds. Each iteration of the Percolation algorithm expands the matched set by adding one unmatched pair of nodes with the most common seeded neighbors. The matching process ends when there are no additional unmatched pairs with more than r seeded neighbors. Also, note that new matches are fixed and won't evolve over iterations. See section 2.1.1 for a more detailed descriptions of the algorithm. Under the correlated Erdős-Rényi model parameterized by n, p, s, it is stated that the Percolation algorithm doesn't make an error in matching a pair of nodes with high probability whenever $p \ll n^{-\frac{3}{r}-\epsilon}/logn$ for $\epsilon > 0$ and $r \geq 4$.

Another set of seeded GM algorithms not included in the `iGraphMatch` package but relevant to our proposed method is the l-PATH and d-HOP WITNESS algorithms proposed by Mossel and Xu (2020). These algorithms were designed primarily for their amenability to theoretical analysis and generally enjoy strong theoretical guarantees. The l-PATH algorithm is designed for matching graphs in the sparse graph regime and is composed of two steps. In the first step, the algorithm counts the number of seeded neighbors with paths of length l to each pair of nodes and matches high degree vertices with m independent l-path seeded neighbors. Then, the algorithm incorporates matches obtained in the first step as seeds to match the remaining nodes in the second step. Notably, the algorithm forfeits matching any candidate pairs when conflicts occur. A conflict occurs if two nodes in one graph match to the same node in the second graph. The l-PATH algorithm is stated in algorithm 5. It is shown that under the correlated Erdős-Rényi model $CorrER(n, p, s)$,

assume $np \leq n^{1/2-\epsilon}$ and $s = \Theta(1)$, if $nps^2 - logn \to +\infty$ and the fraction of seeds over the entire graphs is at least $n^{-1/2+3\epsilon}$ for fixed $\epsilon > 0$, then with probability of $1 - o(1)$, the l-PATH algorithm exactly recovers the underlying true alignment.

Algorithm 5 l-PATH algorithm

Input : A, B, a set of seeds S, $m, l \in \mathbb{Z}$
Output: permutation matrix P
Count the number of l-paths for each pair of nodes $(i, j), i \in V_1, j \in V_2$ to seeds;
Set $P_{ij} \leftarrow 1$ for (i, j) with at least m l-paths;
Set $P_{i\cdot} \leftarrow 0$ and $P_{\cdot j} \leftarrow 0$ if conflicts occur;
Compute $M = APB$;
For unmatched nodes, set $P_{ij} \leftarrow 1$ for (i, j) with $M_{ij} > 0$;
Set $P_{i\cdot} \leftarrow 0$ and $P_{\cdot j} \leftarrow 0$ if conflicts occur;

The d-HOP WITNESS algorithm is proposed to solve the seeded GM problem in the dense graph regime with the assumption that $np = bn^a$ where a, b are fixed constants in $(0, 1]$. The idea of the algorithm is to match a pair of nodes based on their seeded neighbors within distance d, while the l-PATH algorithm counts the number of seeded neighbors wherein a connected path of length exactly l exists. For each node in G_1, it is matched to the node in G_2 with the most seeded neighbors within d-distance, where $d = \lfloor \frac{1}{a} \rfloor$. Algorithm 6 states the d-HOP WITNESS algorithm. It is theoretically proved that the algorithm successfully recovers the true alignment when $b = O(s)$ and $\Omega(\frac{logn}{(nps^2)^{\lfloor 1/a \rfloor}})$ fraction of nodes are seeds.

Algorithm 6 d-HOP WITNESS algorithm

Input : A, B, a set of seeds S, $d \in \mathbb{Z}$
Output: permutation matrix P
Count the number of seeded neighbors within distance d, M_{ij} for each pair of nodes (i, j);
Set $P_{ij*} \leftarrow 1$ where $j* = \text{argmax}_j M_{ij}$;
Set $P_{i\cdot} \leftarrow 0$ if conflicts occur;

3.2 Mutual Nearest Neighbor Algorithm

Similar to the PERCOLATION algorithm described in section 2.1.1, the MNN algorithm also depends on the mark matrix, $M = APB$, where P is a sub-permutation matrix encoding the matches at the current iteration, and A and B are the adjacency matrices of two graphs. Suppose $u, u' \in V_1$, $v, v' \in V_2$, $[u, u'] \in E_1$ and $[v, v'] \in E_2$, then (u', v') is a neighboring pair of (u, v). See figure 3·1 for an illustrative example of neighboring pairs. The mark matrix M can also be interpreted as encoding the number of matched neighboring pairs for each pair of nodes $u \in V_1$, $v \in V_2$. Let m_{uv} denote the number of matched neighboring pairs for (u, v). If $m_{uv} \geq m_{uj}$ and $m_{uv} \geq m_{iv}$ for all $i \neq u$ and $j \neq v$, then we call (u, v) a mutual nearest neighbor. Mutual nearest neighbors get matched in each iteration, and notably, matches are not fixed through iterations. We deal with ties by assigning equal probability of matching to each of them. See Algorithm 7 for a step-by-step overview of MNN.

To facilitate understanding of the MNN algorithm, let us consider its relationship with other existing methods. MNN is closely related to the PERCOLATION algorithm but existing matched node pairs can be replaced with more plausible candidates which prevents a cascade of errors. PERCOLATION

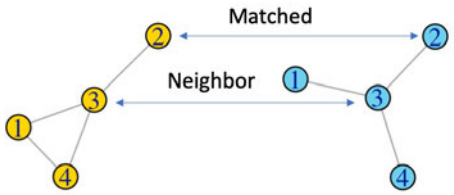


Figure 3·1: An example of neighboring pairs. (3,3) is a neighboring pair of (2,2). Different node color represents different graphs.

also uses a user-specified threshold to determine new matches while MNN requires no parameters. Compared to the FW methods, MNN updates a partial permutation matrix with high-quality matches instead of a full permutation matrix as in the FW over iterations. The MNN algorithm is also closely related to the l-PATH algorithm in the sense that each iteration of the MNN algorithm can also be regarded as a special case of the first matching step of the l-PATH algorithm where $l = 1$. As for the hyperparameter m for the l-PATH algorithm, since the number of seeded neighbors is compared with that number for all the other candidates, the threshold for matching a pair of nodes in the first step can be considered as varied in the MNN algorithm. Additionally, when $d = 1$, the d-HOP WITNESS algorithm finds the nearest neighbor for each node in the first graph while the MNN algorithm matches the intersection of the nearest neighbors for each node in both graphs.

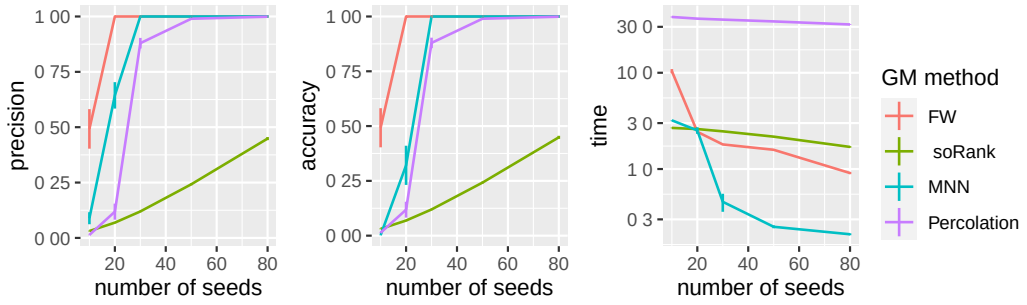


Figure 3·2: Performance comparison between four GM algorithms as a function of different number of seeds.

Algorithm 7 Mutual Nearest Neighbor Algorithm

Input : $A, B \in \{0,1\}^{n\times n}$, α pairs of seeds, maximum iterations *niter*
Output: (sub-)permutation matrix P
Initialize the sub-permutation matrix P^0 incorporating seeds;
Set $i = 0$;
while $P^i \neq P^{i-1}, \text{trace}P^T P < n, i < niter$ **do**
  $M \leftarrow AP^iB$;
  $P^i_{uv} \leftarrow 1$, for all mutual nearest neighbors (u, v) based on M with ties broken at random;
  $i \leftarrow i + 1$;
end

3.3 Experimental results

In this section, we are going to demonstrate the performance of MNN algorithm by comparing it with the FW algorithm with indefinite relaxation, Percolation algorithm, IsoRank algorithm, and the d-hop witness algorithm on synthetic data, Enron email communication networks, and the *C.Elegans* networks.

3.3.1 Erdős-Rényi graphs

In the first simulated data example, we will consider matching two correlated Erdős-Rényi graphs of the same size $n = 400$ from CorrER$(pJ, \rho J)$, where J is a 400×400 matrix with all 1's. We simulate graphs with fixed correlation $\rho = 0.5$ and density $p = 0.5$. We vary the number of seeds $\alpha \in \{10, 20, 30, 50, 80\}$. Seeds are uniformly randomly sampled from all the vertices and we apply hard seeding for the FW algorithm with indefinite relaxation. We fix the threshold $r = 2$ for the Percolation algorithm so that at least 2 seeded neighbors are required for a pair of nodes to match.

Figure 3·2 displays an average of 100 Monte Carlo replicates and provides a 95% confidence bar on precision, accuracy, and running time. We can see that FW outperforms others in all cases in terms of matching precision and matching accuracy. However, MNN algorithm still shows a significant advantage over the Percolation algorithm and IsoRank algorithm, especially in terms of precision. With sufficient seeds (seeds greater than 30), MNN algorithm matches all the nodes correctly. Notably, MNN algorithm runs significantly the faster than the other methods in nearly all cases.

In appendix B, we present another experiment result on the simulated Erdős-Rényi graphs, wherein our focus is on comparing one step of the MNN and the 1-hop witness algorithm on large graphs. Let the number of seeds $\alpha = logn/(p \times \rho)$ be a fixed function of n, p and ρ. As we increase the number of nodes in each graph, we observe a dramatic improvement in matching precision for the MNN algorithm which also slightly increases as we enlarge the graph size. One can refer to appendix B for more details of the experiment setting and results.

3.3.2 Enron Email Network Data

In the first real data example, we consider the Enron email network data which we describe in detail in section 2.2.1. We aim at aligning 184 email accounts from two different weeks based on the communication interactions between them. At least one email sent from one email account to another in a week will incur a directed unweighted edge between the corresponding nodes in the graph. There are 488 and 482 edges in the two networks respectively and the Pearson's correlation between the two networks is 0.85.

Seeds are chosen uniformly at random from all corresponding nodes with the number of seeds varying as $\alpha \in \{5, 10, 20, 40, 60, 80, 100\}$. We run experiments on matching the unseeded nodes with different graph matching methods. In particular, we compare the matching performance of our proposed MNN algorithm with the FW method with the indefinite relaxation, the Percolation algorithm with its hyperparameter $r = 2$, and the d-HOP WITNESS algorithm with $d = 1$. The FW method is included here as a competitive GM method which always returns a mapping of the entire graphs. The other two methods adopt the similar idea of growing a partial alignment based on the neighborhood topology, and we are interested in the comparison between these methods and the MNN algorithm. Note that we don't conduct any hyperparameter tunings for the Percolation algorithm and the d-HOP WITNESS algorithm in this experiment and choose the default settings. In the following experiment, we will discuss the impact of different choices of hyperparameters on the same dataset.

Figure 3·3 displays an average of 300 Monte Carlo replicates for each graph matching method and number of seeds $\alpha \in \{5, 10, 20, 40, 60, 80, 100\}$. Un-

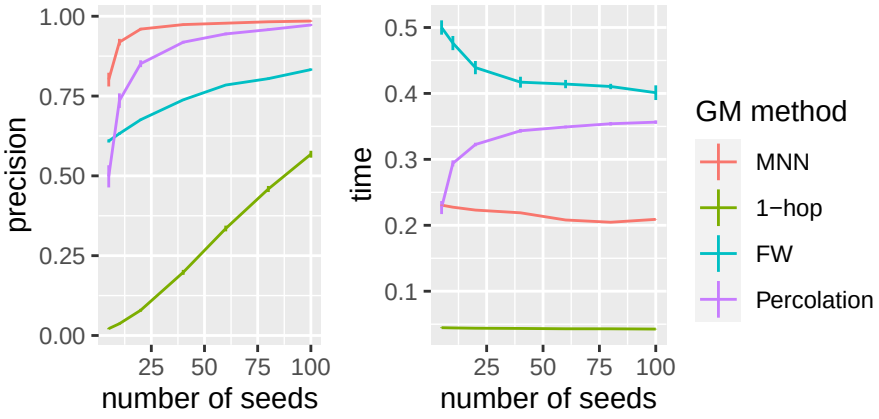


Figure 3·3: Averaged precision and running time for Enron graphs.

der all cases, MNN shows significantly better performance in finding a set of matched vertices with higher precision in a smaller amount of running time. Although the 1-HOP WITNESS method runs even faster than the MNN algorithm, it is less preferable in terms of its matching precision.

As a follow-up to the previous experiment, we apply additional hyperparameter tuning steps and analysis to extract smaller sets of matched nodes for the FW method and Percolation algorithm. For the FW method, we further apply a permutation-test-based matching metric to the matched pairs obtained from the previous experiment and rank them in terms of goodness of matching. We fix the number of top-ranked matches to be the averaged number of nodes matched by MNN over 300 Monte Carlo replicates for each of $\alpha \in \{20, 40, 60, 80, 100\}$. In general, as we take a larger subset of matched nodes, the matching precision of the matched set would decrease. More details on the nodal matching metric are provided in section 2.1.4 with usage examples described in section 2.2.1. For the Percolation algorithm, the hy-

perparameter r controls a tradeoff between the number of matched nodes and the confidence in the obtained matched set. A larger threshold r would result in a smaller matched set with higher precision. We vary the threshold $r \in \{2, 3, 4, 5, 6\}$ for the Percolation algorithm, and pick the threshold which generates a matched set with the closest number of matches compared with the MNN algorithm. We observe that when $r = 3$, PERCOLATION matches a comparable number of nodes.

Figure 3·4 visualizes the matching precision of the smaller set of matched nodes versus the number of matches for the FW method with the application of node-wise matching metric, Percolation algorithm with larger r and the MNN algorithm. The different number of seeds are represented by point sizes. A larger point is an indicator of more seeds used in the matching process. Note that there are no hyperparameters for the MNN algorithm, the result for this method would be the same as displayed in figure 3·3. To interpret the results in figure 3·4, we would prefer methods that generate more matched pairs and higher precision, which correspond to points to the upper right side of the plot. With a sufficient number of seeds, MNN matches more seeds with higher precision and more convenient without the need to tune hyperparameters. When the size of seeds is smaller, the FW method can outperform MNN algorithm. However, one has to decide the size of the subset of matches that optimizes a balance between the size of the subset of matches and its matching precision of it, which is not trivial.

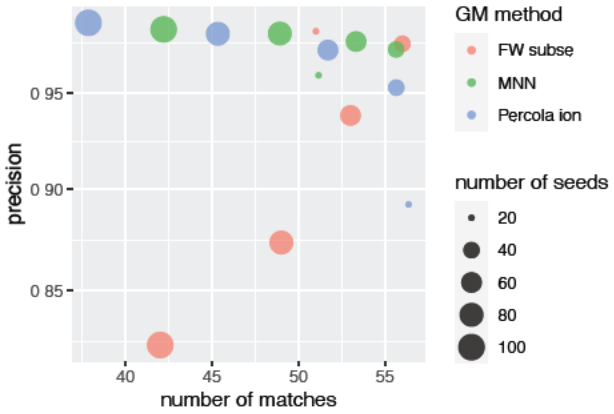


Figure 3·4: Averaged precision and size of the subset of matches for Enron graphs.

3.3.3 *C.Elegans*

The *C.Elegans* networks are composed of a chemical synapses network and an electrical synapses network of the roundworm. There are 279 nodes in each network with each one representing a neuron. There are 2194 and 1031 edges in two networks respectively with each weighted directed edge corresponding to interactions between two neurons. For simplicity, we will transform each networks to be unweighted and undirected by setting two nodes to be adjacent when at least one interaction exists between them. The correlation between the resulting networks is 0.1. Section 2.2.2 states more details of the networks and illustrates the reasons that matching the *C.Elegans* networks is a challenging task.

To align the neurons in the chemical synapses network and the electrical synapses network, we match the two networks with seeds selected uniformly at random. Figure 3·5 displays averaged precision and running time as a function of the number of seeds and different GM methods over 300 Monte Carlo repli-

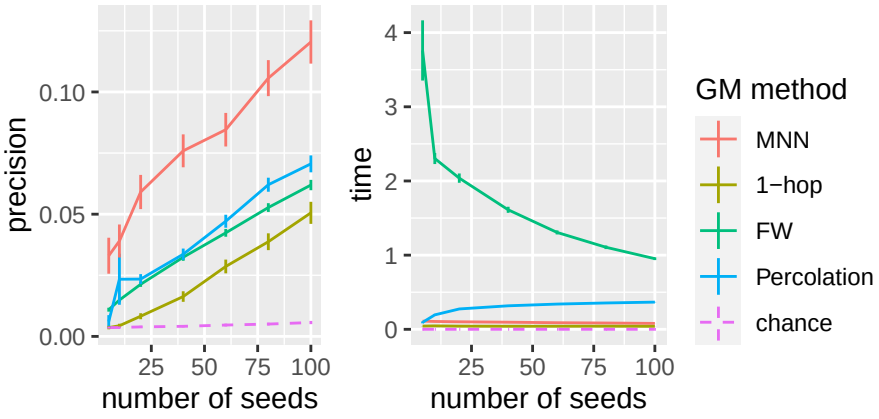


Figure 3·5: Averaged precision and running time for *C.Elegans* graphs.

cates. We vary the number of seeds among $\{5, 10, 20, 40, 60, 80, 100\}$. Similar to the settings in the previous Enron data example, we will first compare the MNN algorithm to the FW algorithm with indefinite relaxation, Percolation algorithm with threshold hyperparameter $r = 2$ and d-HOP WITNESS algorithm with $d = 1$.

As shown in figure 3·5, the performance of MNN dominates the other algorithms in terms of matching precision. Although all the methods fail to match many nodes correctly, they are significantly better than matching by chance as indicated by the pink dashed line in the plots. The MNN algorithm is also computationally more efficient compared with the FW method and the Percolation algorithm, especially when the number of seeds is greater than 10.

Furthermore, we investigate the performance of the MNN algorithm compared with a more competitive smaller set of matches obtained by the FW algorithm and Percolation algorithm. Analogous to the experiment conducted on the Enron dataset, we generate subsets of matches by applying a metric

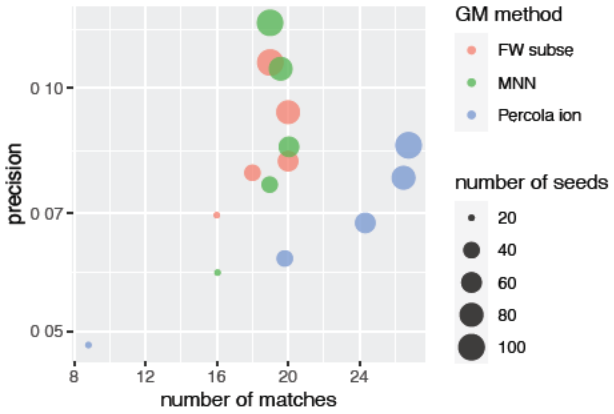


Figure 3·6: Averaged precision and size of the subset of matches for *C.Elegans* graphs.

on the matched pairs which measures the likelihood of being correct matches on the matching result of the FW method obtained on the entire graphs. For the Percolation algorithm, we use larger r values to get more reliable matches at the expense of a smaller matched set. We pick the number of top-ranked matches for the FW method to be equal to the averaged number of matched nodes by using MNN, and the threshold r for the Percolation algorithm so that the generated matched sets have the closest number of matched node pairs to the one obtained by the MNN algorithm. More specifically, we pick $r = 4$ when there are 20 seeds and $r = 6$ for the other cases.

We observe similar results here as in the Enron data example. The results in figure 3·6 suggest that when there are enough seeds, the MNN algorithm has the best performance among three algorithms in terms of precision and number of matches. Therefore, the MNN algorithm provides a hyperparameter-free and computationally more efficient alternative method for generating a high-quality partial alignment with sufficient prior information. When there is

limited prior information, one can resort to the FW method and measure the matching performance over all the matched pairs. However, additional analysis is required to determine the size of the subset of matches.

3.4 Theoretical results

As a theoretical complement to the experimental results, in this section, we target providing conditions when the MNN algorithm achieves high matching precision under the correlated Erdős-Rényi graph model. Under mild conditions, it is theoretically well studied that the d-HOP WITNESS algorithm recovers the exact alignment with probability $1 - 4n^{-1}$ with the full theorem stated in theorem 3.4.2. With more strict matching criteria, our conjecture is that MNN should find a high-precision set of matches even when the the seeds grow at a slower rate than that required for the d-HOP WITNESS algorithm. Ideally, this high-precision set can be used to improve subsequent analysis. Additionally, we hope to theoretically prove another conjecture that MNN can fix incorrect matches with enough information for true correspondence. In this work, we focus on initial steps toward first conjecture and leave the second conjecture to future work.

Assuming that two graphs are sampled from the correlated Erdős-Rényi graph model, we consider a probabilistic framework for analyzing the event that the MNN algorithm introduces incorrect matches. The proof consists of two steps and we provide conditions for bounding each of them. It remains an open question how to balance the results of two steps so that we can get an improved rate theoretically compared with the one for the d-HOP WITNESS algorithm.

For the remaining of this section, we are going to adopt the following notations. Two graphs $G_1, G_2 \sim CorrER(n, p, s)$ are independently sampled conditioned on a graph G. G is an Erdős-Rényi graph with n nodes and edge probability p. Both G_1 and G_2 maintain the same set of nodes as the original graph G. Each edge of G is present in $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$ independently with probability s, so that G_1 and G_2 are marginally $ER(n, ps)$. Under this sampling scheme, the true underlying alignment is the identity. Let S be the set of seeds with α pairs of seeds and they are all correct. Without loss of generality, let us assume the first α pairs of nodes are seeds.

3.4.1 Mathematical formulation

To mathematically formulate the MNN algorithm's high matching precision, we consider the event of incorrectly matching a pair of non-seeded nodes $(u, v), u \in V_1, v \in V_2$, and $u \neq v$ in the first matching iteration. This happens when (u, v) is a mutual nearest neighbor, that is (u, v) has more seeded neighbors than all the other candidate pairs $(u, j), j \neq v$ and $(i, v), i \neq u$. Let X_{uv} denote the event that a pair of nodes (u, v) is matched. Then, the probability of it is upper bounded by the following:

$$\begin{aligned} P(X_{uv}) &\leq P(\bigcap_{j \neq v} m_{uv} \geq m_{uj}, \bigcap_{i \neq u} m_{uv} \geq m_{iv}) \\ &= \sum_{r=1}^{\alpha} P(m_{uv} = r, \bigcap_{j \neq v} m_{uj} \leq r, \bigcap_{i \neq u} m_{iv} \leq r) \\ &= \sum_{r=1}^{\alpha} P(m_{uv} = r, m_{uu} \leq r, m_{vv} \leq r, \bigcap_{i \neq u,v} \{m_{ui} \leq r, m_{iv} \leq r\}) \end{aligned} \tag{3.1}$$

Note that this is an upper bound due to the case of ties, eg. $m_{uv} = m_{uu}$, since if there are multiple MNNs for a given node, one is selected at random.

Upper bounding the probability of introducing at least one wrong match into the matched set can be achieved by taking a union bound over $u \neq v$.

Let $E_{ui}^{(1)}$ and $E_{iv}^{(2)}$ be binary variables that indicate the presence of an edge between node u and i in G_1, and an edge between node i and v in G_2, respectively. If an edge exists between two nodes $u, i \in V_1$, then $E_{ui}^{(1)} = 1$, otherwise, $E_{ui}^{(1)} = 0$. Recall that for a pair of nodes $(u, v), u \in V_1, v \in V_2$, m_{uv} denotes the number of seeded neighbors to (u, v). Let $I_{uv}(i)$ be an indicator function of the event that the i^{th} pair of seeds is a neighbor of (u, v). We can rewrite m_{uv} as:

$$m_{uv} = \sum_{i=1}^{\alpha} I_{uv}(i) = \sum_{i=1}^{\alpha} E_{ui}^{(1)} \cdot E_{iv}^{(2)}$$

We include a lemma from (Yartseva and Grossglauser, 2013) on the distributions and statistical properties of $I_{uv}(i)$'s. The lemma is originally stated under the propagation setting where the matched set gets updated after processing each new pair of seeds. The authors assume that such propagation updates don't introduce matching errors. Here we only consider the first matching iteration where all the seeded information is processed before an updating decision is made. Thus, we don't need to make additional assumptions about our problem.

Lemma 3.4.1 (Yartseva and Grossglauser (2013))**.** *Let the first α pairs of nodes be correct seeds. Assume $u \in V_1, v \in V_2$ and $u, v > \alpha$. For each $i \leq \alpha$,*

1. $P(I_{uv}(i) = 1) = \begin{cases} (ps)^2, & u \neq v \\ ps^2, & u = v \end{cases}$
2. *For fixed i, $\{I_{u,v}(i)\}$ are not independent when $u \neq v$.*
3. *For fixed i and u, $\{I_{u,v}(i)\}$ are independent when $v \neq u$. Similar result holds for fixed i and v.*

4. For any $i_1 \neq i_2, i_1, i_2 \leq \alpha$, $\{I_{u,v}(i_1)\}$ and $\{I_{u,v}(i_2)\}$ are independent.

The first result in the lemma shows that a pair of nodes corresponding to a true match has a higher probability of being a neighbor to a pair of seeds compared with a pair from an incorrect match. Intuitively, the probability of an incorrect pair having more seeded neighbors and getting matched should be low. The rest of the results are useful for determining the distributions of random variables involved in the proof for the propositions in section 3.4.2.

Recall that our proposed MNN algorithm resembles the d-HOP WITNESS algorithm with $d = 1$. Under the dense setting, which is achieved when $a = 1$ under the assumption that $np = bn^a$ and $d = \lfloor \frac{1}{a} \rfloor + 1$. Theorem 3.4.2 states the conditions in terms of the density of graphs and number of seeds needed for a successful recovery of the true alignment (Mossel and Xu, 2020).

Theorem 3.4.2 (Mossel and Xu (2020))**.** *Suppose $G_1, G_2 \sim CorrER(n, p, s)$. Assume*

$$b \leq \frac{s}{16(2-s)^2},$$

and

$$\alpha \geq \frac{300n \log n}{(nps^2)^d}.$$

Then the d-HOP WITNESS algorithm recovers the true alignment with probability $1 - 4n^{-1}$.

Compared with the 1-HOP WITNESS algorithm, which finds the best matching candidate in G_2 for each node in G_1, MNN performs an additional step to ensure that each matched node in G_2 is also matched to an optimal candidate in G_1. If we ignore the failure case of the 1-HOP WITNESS algorithm wherein more than one node in G_1 is matched to the same node in G_2, a node pair matched by the MNN algorithm must be in the matched set of the 1-HOP

WITNESS algorithm. However, a node pair matched by the 1-HOP WITNESS algorithm can fail in being a mutual nearest neighbor and thus not be matched by the MNN algorithm. With more stringent matching criteria, the MNN algorithm matches a smaller set of nodes often with higher precision compared with the 1-HOP WITNESS algorithm.

Note that we didn't consider the failure case of the 1-HOP WITNESS algorithm when analyzing the empirical precision gains of the MNN algorithm. When there are matching conflicts, the d-HOP WITNESS algorithm would directly declare failure and return with no matches. This is not realistic in many real-world cases where the conditions provided in the theoretical results fail. On the other hand, MNN never declares failure and hence will always provide a set of matches which can potentially be used in subsequent analysis.

3.4.2 Sketch of theoretical analysis

In this section, we first characterize the objective function 3.1 by a series of independent events along with their distributions. Then we provide an outline of the proof steps.

Conditionally Independent events

We consider fixed u, v, with $u \neq v$. To begin with, let $L = \{i_1, ..., i_l\} \in S$ be the set of seeded vertices in G_1 that have an edge connected to node u. Let $K = \{j_1, ..., j_k\} \in S$ be the set of seeded vertices in G_2 that have an edge connected to node v. Let $|\cdot|$ denote the cardinality of a set. Under the assumption that the true alignment is the identity, for any $r = 1, 2, ..., \alpha$, each probability term to the right side of equation 3.1 can be computed by

establishing a series of events as follows:

1. $|K \cap L| = |R| = r$.

2. Out of L, at most r corresponding vertices connected to node u in G_2, denoted by J.

3. Out of K, at most r corresponding vertices connected to node v in G_1, denoted by H.

4. Let $X \subseteq R$ denote the set of seeded vertices connected to node i, for $i \neq u, v$ and $i > \alpha$, in both G_1 and G_2, which are also connected to node u in G_1 and node v in G_2. Let $Y \subseteq R \setminus X$ denote the set of vertices with edges connected to node i in G_2, but no edge to node i in G_1.

   (a) Out of $L \setminus R$, at most $r - |X| - |Y|$ connected to i in G_2, denoted by W.

   (b) Out of $K \setminus (X \cup Y)$, at most $r - |X|$ connected to i in G_1, denoted by V.

Conditional on the cardinality of L and K, the right side of the objective function 3.1 can be decomposed into independent events described in the above events. The first event amounts to saying a wrong pair (u, v) has r seeded neighboring pairs, corresponding to the term $m_{uv} = r$ in the objective function 3.1. The second and third events characterize $m_{uu} \leq r$ and $m_{vv} \leq r$ respectively. The last events altogether define that $m_{ui} \leq r$ and $m_{iv} \leq r$ both hold, for $i \neq u, v$ and $i > \alpha$. According to the second result in Lemma 3.4.1, the number of seeded neighbors of a node pair (u, v) and its candidate pair

$(u,i), i \neq v$ are not independent. Conditional on $|R|, |L|$ and $|K|$, we break down the set of seeded nodes connected to i in G_2 as a union of three mutually exclusive sets X, Y and W to handle the dependence. Similarly, the set of seeded nodes connected to i in G_1 is composed of X and V. Rewrite the objective function 3.1 in terms of the independent events conditional on the size of L and K, and apply the law of total probability, we get

$$(3.1) = \sum_{l=1}^{\alpha}\sum_{k=1}^{\alpha}\sum_{r=1}^{min(l,k)} P(|L|= l)P(|K|= k)P(|R|= r)P(|J|\leq r)P(|H|\leq r)\cdot$$
$$[\sum_{x=0}^{r} P(|X|= x)\cdot \sum_{y=0}^{r-x} P(|Y|= y)P(|W|\leq r-x-y)P(|V|\leq r-x)]^{n-2}. \tag{3.2}$$

Notably, in the above equation we write conditional probabilities as probabilities except for $P(|L|= l)$ and $P(|K|= k)$. For example, $P(|R|= r)$ stands for $P(|R|= r \mid |L|= l, |K|= k)$. Unless otherwise specified, we adopt this notation in the rest of the proof.

Table 3.1 provides an overview of all the random variables defined above with their definitions in terms of edge connectness indicators $E^g_{uv}, u \in V_1, v \in V_2, g = 1, 2$. In the third column, we state the distribution of each random variable under the correlated Erdős-Rényi model $CorrER(n,p,s)$. Specifically, for fixed $|L|= l, |K|= k$, the size of set R follows the Hypergeometric distribution parameterized by α, l and k. Accordingly, the probability of node (u,v) has r seeded neighbors is

$$P(|R|= r \mid |L|= l, |K|= k) = \frac{\binom{l}{r}\binom{\alpha-l}{k-r}}{\binom{\alpha}{k}}.$$

Based on part 4 of Lemma 3.4.1, each seed being a neighboring pair to a non-seeded node pair is independent of each other. The other random variable can be treated as the sum of i.i.d. Bernoulli random variables with corresponding sample size and success probability.

The objective function 3.2 can also be viewed as an expected value in terms of a function of $|L|, |K|$ and $|R|$. The first three probabilities altogether define the joint probability of $|L|, |K|$ and $|R|$ and let the joint probability be denoted by $g(r, l, k)$. The remaining terms make up a function of these three random variables and we denote the function by $f(r, l, k)$. Then the probability of (u, v) being incorrectly matched is upper bounded by $\mathbb{E}[f(r, l, k)]$. The only caveat is that when summing over all the possible values of $|R|, |L|$ and $|K|$, we exclude the case when (u, v) and all the other matching candidates are not neighbors to any seeds, i.e. $|R| = 0$, but (u, v) still gets matched. This is because in the algorithm we make the requirement that at least one seeded neighbor is required for a pair of nodes to get matched.

Proof outline

To get an upper bound for $\mathbb{E}[f(r, l, k)]$, we split the summation over $\{1 \leq l, k \leq \alpha, 1 \leq r \leq min(l, k)\}$ into two parts. The overall proof idea is that L, K, and R will concentrate around typical values and around these typical values, the value of function $f(r, l, k)$ is small.

Notice that $f(r, l, k)$ increases as l, k decrease, and as r increases which is shown in proposition A.0.1 in the appendix A. Recall that $f(r, l, k)$ is the conditional probability of correct pairs (u, u) and (v, v) and the other $2 \times (n-2)$ incorrect candidate pairs all having fewer seeded neighbors than (u, v).

Table 3.1: Definitions and distributions of random variables in the objective function 3.2. Suppose $G_1, G_2 \sim CorrER(n, p, s)$, the first α node pairs are seeds, and $i \neq u, v$.

Random variable	Definition	Distribution
$\|L\|$	$\sum_{j=1}^{\alpha} E_{uj}^{(1)}$	$Bin(\alpha, ps)$
$\|K\|$	$\sum_{j=1}^{\alpha} E_{vj}^{(2)}$	$Bin(\alpha, ps)$
$\|R\|$	$\sum_{j=1}^{\alpha} E_{uj}^{(1)} E_{vj}^{(2)}$ given $\|L\|= l, \|K\|= k$	$Hyper(\alpha, l, k)$
$\|J\|$	$\sum_{j=1}^{\alpha} E_{uj}^{(2)}$ given $\|L\|= l$	$Bin(l, s)$
$\|H\|$	$\sum_{j=1}^{\alpha} E_{vj}^{(1)}$ given $\|K\|= k$	$Bin(k, s)$
$\|X\|$	$\sum_{j=1}^{\alpha} E_{ij}^{(1)} E_{ij}^{(2)} E_{uj}^{(1)} E_{vj}^{(2)}$ given $\|R\|= r$	$Bin(r, ps^2)$
$\|Y\|$	$\sum_{j=1}^{\alpha} (1 - E_{ij}^{(1)}) E_{ij}^{(2)} E_{uj}^{(1)} E_{vj}^{(2)}$ given $\|R\|= r, \|X\|= x$	$Bin(r - x, ps)$
$\|W\|$	$\sum_{j=1}^{\alpha} (1 - E_{ij}^{(1)}) E_{ij}^{(2)} E_{uj}^{(1)} (1 - E_{vj}^{(2)})$ given $\|L\|= l, \|R\|= r$	$Bin(l - r, ps)$
$\|V\|$	$\sum_{j=1}^{\alpha} E_{ij}^{(1)} (1 - E_{ij}^{(2)} E_{uj}^{(1)}) E_{vj}^{(2)}$ given $\|K\|= k, \|X\|= x, \|Y\|= y$	$Bin(k - x - y, ps)$

Conditions of the probability include there being l seeded nodes connected to u in G_1, k seeded nodes connected to v in G_2, and that (u, v) have r seeded neighbor. When the subgraph containing candidate nodes and seeded nodes are sparse, and (u, v) has many seeded neighbors, it is unlikely that the other candidate pairs will have more links with seeded nodes.

Accordingly, let's split the summation area at $l_0 < E[|L|]$, $k_0 < E[|K|]$, and $r_0 > E[|R|]$. The summation over $\{l_0 \leq l \leq \alpha, k_0 \leq k \leq \alpha, 1 \leq r \leq r_0\}$ will yield a high probability and we will upper bound $f(r, l, k)$ by $f(r_0, l_0, k_0)$ in this area. The other part covers the complement, representing values that $|R|, |L|$ and $|K|$ are unlikely to take. For simplicity, we will bound $f(r, l, k)$ by 1 in this complement set. Then the objective function 3.2 is upper bounded by adding up these two parts:

$$(3.2) \leq f(r_0, l_0, k_0) + \sum_{\{l_0 \leq l \leq \alpha, k_0 \leq k \leq \alpha, 1 \leq r \leq r_0\}^c} g(r, l, k)$$

The remaining proof steps include providing upper bounds for each part separately, and then determining r_0, l_0, k_0 as functions of parameters α, p, s.

Note that it's not trivial to derive an upper bound for the probabilities inside the bracket in 3.2. Analysis analogous to the one in terms of r, l, k can be applied here. Let $g_{XY}(x, y)$ denote the joint probability mass function for $|X|$ and $|Y|$, and $f_{XY}(x, y)$ denote the product of last two left tail binomials in the bracket. We split the summation over x, y at x_0 and y_0 to the left of $|X|$ and $|Y|$'s means. Similarly, the guiding idea is that $f_{XY}(x, y)$ decreases as x, y increases. Thus after splitting the entire summation area into two parts, we bound the first part by $f_{XY}(x_0, y_0)$. The other part is bounded by the probability of $|X|$ and $|Y|$ taking values in the compliment set.

3.4.3 Proposition results

Propositions in this section provide upper bounds for each component of the objective function separately. Note that it's a challenging task to balance the probabilities of each component by tuning the split values over r, l, k, and thus we could hardly obtain a better rate than the d-HOP WITNESS algorithm at this point. We leave this to future work and would focus on the proposition results and interpretations of the results in this section. We state the results here and leave the proof in appendix A. Unless otherwise specified, all the propositions are stated under the following settings. Let $G_1, G_2 \sim CorrER(n, p, s)$, and α node pairs are seeds. We split the summation over r, l, k in the objective function 3.2 at $r_0 = (1 + \epsilon_1)\alpha p^2 s^2, l_0 = k_0 = (1 - \epsilon_2)\alpha ps$ for some positive ϵ_1

and ϵ_2. In general, we assume that ϵ_1 and ϵ_2 are small.

When $|L|, |K|$ and $|R|$ are concentrated around their typical values, proposition 3.4.3 upper bounds the event that corresponding correct matched nodes (u, u) and (v, v) are not matched, denoted by f_1. Proposition 3.4.5 provides an upper bound to the event that $|R|, |L|, |K|$ take values that are unlikely to happen, denoted by f_2. Proposition 3.4.4 upper bounds the event that $|X|$ and $|Y|$ take values with small probability, denoted by g_1. Recall that deriving an upper bound for the probability terms inside the bracket in equation 3.2 consists of two steps. Furthermore, let us denote the upper bound of the second step by g_2 and $0 \leq g_2 \leq 1$. Note that we will provide conditions when g_1 goes to 0 and it is sufficient to use 1 as the upper bound for g_2.

Proposition 3.4.3. *Suppose there are $|L| = l_0$ seeded nodes connected to u in G_1, and $|K| = k_0$ seeded nodes connected to v in G_2. Assume $p < 0.6, 0 < \epsilon_1, \epsilon_2 < 1/4$, then*

$$P(|J| \leq r_0) \leq exp(-\frac{\alpha p s^2 (\frac{3}{4} - \frac{5}{4}p)^2}{2})$$

and

$$P(|H| \leq r_0) \leq exp(-\frac{\alpha p s^2 (\frac{3}{4} - \frac{5}{4}p)^2}{2}).$$

Proposition 3.4.4. *Let $x_0 = (1 - \delta_1) r p s^2$, $y_0 = (1 - \delta_2)(r - x) p s$. For $0 < \epsilon_1 \leq 1$:*

$$\sum \sum_{\{x \geq x_0, y \geq y_0\}^c} P(|X| = x) \cdot P(|Y| = y) \leq exp(-\frac{(1 + \epsilon_1)(log\alpha)^2 p s^2}{2}) + exp[-\frac{(1 + \epsilon_1)(log\alpha)^2 p s}{2} + (e^{\frac{(log\alpha)^2}{2\alpha p s}} - 1)(1 + \epsilon_1)\alpha p^3 s^4]$$

Proposition 3.4.5. *Assume $0 < \epsilon_1 < \frac{1}{p^2 s^2} - 1, 0 < \epsilon_2 < 1$. Let $g(r, l, k) =$*

$P(Bin(\alpha, ps) = l) \cdot P(Bin(\alpha, ps) = k) \cdot P(Hyper(\alpha, l, k) = r)$, *then*

$$\sum\sum\sum_{\{r\le r_0, l\ge l_0, k\ge k_0\}^c} g(r,l,k) \le exp(-\frac{1}{16}\alpha p^3 s^3 \epsilon_1^2) + 2exp(-\frac{1}{48}\alpha ps\epsilon_1^2) + 2 \cdot exp(-\frac{\alpha ps\epsilon_2^2}{2})$$

Based on proposition 3.4.3, when $|L|, |K|$ and $|R|$ are concentrated around their typical values, the probability that the corresponding correct node pairs are not matched approaches 0 w.h.p. when $\alpha ps^2 = \Omega(logn)$. Proposition 3.4.4 provides an upper bound for the probability that the other incorrect candidate pairs get matched. Condition $(log\alpha)^2 ps^2 = \Omega(logn)$ needs to hold in order for the probability to be bounded. Based on the last proposition, $(log\alpha)^2 ps^2 = \Omega(logn)$ needs to hold so that the probability of r, l, k away from their typical values is small.

The upper bound for the objective function in equation 3.2 which defines the probability of matching a pair of nodes (u, v) incorrectly becomes

$$(3.2) \le f_1^2 \cdot (g_1 + g_2)^{n-2} + f_2.$$

Taking a union bound for all u, v, we get an upper bound for the probability of generating a pair of nodes over the entire graphs. Accordingly, we take union bounds for each of f_1, g_1, and f_2 and obtain conditions when they approach 0 as the graph size goes to infinity.

Chapter 4

Graph matching with combined prior information

In addition to the topological structure of networks, various attributes associated with nodes and edges are often available in real-world problems. The type of node and edge attributes is diverse and can vary from simple scalars or categorical variables to text or even video data. In social networks, there is often rich information available on the network participants and their interactions, such as user profiles and contextual content (Liao et al., 2018). In pattern recognition, nodes and edges can be attributed to encode features computed on the intersections of grids superimposed on the images (Conte et al., 2004). In addition, node statistics such as degree, centrality, and ego subgraph counts can be used as additional features for generating similarities.

Similarity scores can usually be generated from nodal covariates by using information embedded in attributes to quantify the relatedness of two nodes (Kelley et al., 2004; Belongie et al., 2002). How to effectively utilize the prior information and combine them to enhance graph matching becomes a key question. In this chapter, we first describe a de-anonymization problem that motivates our work, then we review related work, before introducing a new approach for combining similarities for graph matching. Finally, we compare

approaches on both synthetic data and on the motivating problem.

4.1 Venmo transaction knowledge graphs

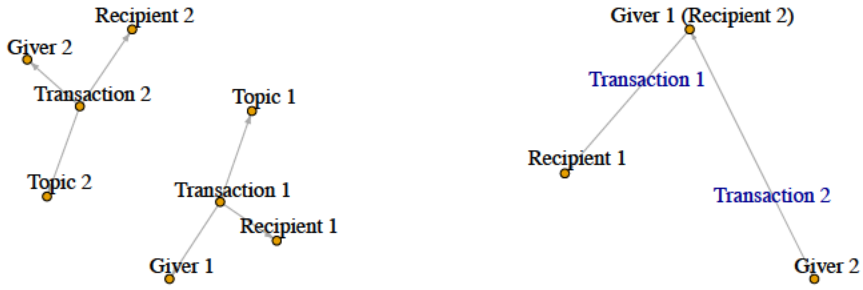
We consider a de-anonymization problem proposed as part of the DARPA Modeling Adversarial Activity program. The exploration of solving this problem motivates us to propose a similarity-based learning method adapted to the general attributed graph matching task which we will illustrate in section 4.3.2. Here, the goal is to align users based on their Venmo transactions in August and October 2018 along with additional attributes associated with users and their transactions. This is a simplified version of a larger problem and the transaction graphs are derived from a larger knowledge graph encompassing a longer time frame and additional individuals.

The transactions between participants and their additional information are originally stored in knowledge graphs (Hu et al., 2018), which are graphical representations of the relationship between real-world entities, such as objects, events, and locations. In the Venmo transaction knowledge-based graphs, there are four types of nodes including transaction nodes, giver nodes, recipient nodes, and topic nodes. Each transaction node is connected with a giver node and recipient node. For most transactions, there are also topic nodes connected to it containing information on the purchase category. For example, topic nodes can refer to laundry, gas stations, or badminton, to name a few. Note that even if some users are involved in more than one transaction, they are represented by different nodes. In this way, each transaction forms a directed star graph and is disconnected from the other transactions.

To conform with the form of input for most graph matching algorithms, in

the pre-processing step, we first transform the original knowledge-base graphs by into graph where each edge represents a transactions. The nodes represent users and the directed edges point from givers to recipients. Also, the same identities across different transactions are combined into a single node. Finally, we maintain the topic nodes as attributes associated with each edges.

After transforming the graphs, node degrees indicate the number of transactions in which the corresponding users are involved. In particular, the out-degree of a node indicates the number of transactions when the corresponding user is a giver, and the in-degree indicates how many times the user receives transactions. See figure 4·1 for an example with two transactions. After transforming the graph structure, Venmo transaction graphs have 2846 nodes in each graph. There are 2326 and 2074 edges in each graph respectively. The Pearson's correlation between the two graphs is 0.274.



(a) (b)

Figure 4·1: Example of two representations of the Venmo transaction graph: (a) knowledge-based graph; and (b) regular graph. Giver 1 and recipient 2 correspond to the same identity.

In addition to the nodes and edges in the original knowledge base graphs which depict the relationship between users, transactions, and transaction topics, we are provided with more prior information on dates and locations of transactions embedded in the node attributes in the knowledge base graphs.y Transaction topics are stored in 1024-dimensional word2vec embeddings (Mikolov et al., 2013). Topics with closer semantic meanings are closer to each other in the word2vec space. Transaction dates are stored in the form of year, month, and day, and transaction locations are stored in the form of latitude and longitude. In the transformed graphs, we associate all attributes with user nodes. Table 4.1 lists all the user features used in the modeling phase and their descriptions. For transaction topics and locations, we take an average of word2vec embeddings, latitude, and longitude of all the transactions associated with each node. Regarding missing values, we impute using the mean of non-NA values. We also define a transaction diversity metric by adding the diagonal entries of the variance-covariance matrix calculated over all the transaction topic vector embeddings for a particular user. Additionally, we introduce degree, out-degree, and the percentage of out-degree based on the topological network structure. Also, we include a user engagement metric to measure the number of active days that a user makes at least one transaction during a month, denoted by lness. Degree per lness is another feature defined as the average number of transactions per active day.

To come up with an interpretable model, we generalize the objective function for the graph matching problem in Equation 2.4 to incorporate multiple similarity scores built under certain distance metrics. Formally, suppose multiple similarity scores $S_1, \ldots, S_p$ are available, and the objective function

Table 4.1: Summary of user features for Venmo transaction graphs.

Feature	Description
Degree	Number of transactions.
Out degree	Number of out-sending transactions.
Out degree%	Out degree / degree * 100%.
Lness	Number of days with at least 1 transaction.
Degree per lness	Number transactions per active day.
Transaction topic	Averaged 1024-dimensional word2vec embedding.
Transaction diversity	Sum of diagonal entries of variance covariance matrix for 1024-dimensional word2vec embeddings.
Location	Averaged latitude and longitude.

becomes:

$$\underset{P\in\Pi}{\operatorname{argmin}} \sum_{l=1}^{m} \|A^{(l)} - (I_s \oplus P)B^{(l)}(I_s \oplus P)^T\|_F^2 - \sum_{i=1}^{p} w_i \cdot \operatorname{trace} S_i^T (I_s \oplus P) \quad (4.1)$$

The main issue is how to determine the weights w_i's that best address the relative importance of each similarity score and optimize the matching performance. A naive solution would be assigning equal importance to each similarity score by setting $w_i = \frac{1}{S_i}$. However, averaging similarity scores may not always be helpful and we hope to develop a method for learning optimal weights.

Additionally, the Venmo user transaction graphs are extremely sparse with edge probabilities in each graph being smaller than 0.001 and the two graphs are weakly correlated on the topological level. Therefore, it is not sufficient to use the topological structure alone in the Venmo user de-anonymization problem. To improve performance, we will need to utilize the various node and edge attributes to build similarity scores.

4.2 Related work on combining multiple features

There is limited work in the literature on combining multiple features to aid the process of graph matching. Among them, the most relevant ones include filtering methods (Moorman et al., 2018b), record linkage algorithms (Binette and Steorts, 2022), and similarity-based learning algorithms. We will review each of these in this section.

4.2.1 Filtering methods

Moorman et al. (2018b) proposed filtering methods for finding all the subgraphs in a larger graph which are isomorphic to a smaller template graph. The filtering method is composed of a node-level statistics filter, topology filter, and repeated set filter. Firstly, node-level statistics such as in-degree, out-degree, and the number of reciprocal edges are considered to build filters which are applied to each template node in the smaller graph and all the candidates in the larger graph sequentially. Each node-level statistic of a candidate node in the larger graph has to be larger than the corresponding property of a template node in the smaller graph. For example, a template node with 5 out-degrees can only be matched to a candidate node with at least the same number of out-degrees. Topological filters are built to ensure that the connectivity between two template nodes and their filtered candidates is consistent. If an edge exists between two nodes in the template graph, there should also be an edge between their filtered candidates. Finally, the repeated set filter is applied to prevent multiple template nodes getting matched to one candidate node. Ideally, after applying all the filters, each template node will be nominated with a set of candidates that is small enough to enumerate all

the isomorphic subgraphs.

This heuristic method succeeds in solving the task on several real datasets and can be possibly extended to incorporating node and edge attributes by building additional filters. However, it is designed specifically for finding an isomorphic mapping of the template graph while in many real cases, we are only able to find alignments that minimize edge disagreements. Therefore, this method can not be directly applied to matching the Venmo transaction graphs since users' behavior across different months are related but not identical.

4.2.2 Record linkage

Record linkage (RL), which is also commonly known as identity resolution and de-duplication, aims at integrating information from multiple databases which can be beneficial for downstream analysis. Record linkage methods are applied to a wide range of real-world problems, such as the United States decennial census, voter registration data, and network analysis for identifying co-authorship. The analogy of RL to the graph matching problem of interest is that in many downstream tasks, the goal of RL is to ensure every identity is counted once and only once based on common attributes among datasets; For the graph matching problem, we also target aligning same identities across graphs by using graph structural information and node and edge attributes.

Many record linkage methods are improved upon the Fellegi-Sunter method proposed by Fellegi and Sunter (1969). The method first computes a comparison vector $\gamma = (\gamma_1, ...\gamma_p)$, where γ_i indicates the comparison between attribute i of the record pair. Let $m(\gamma)$ and $u(\gamma)$ be probabilities of observing γ for two records that are or are not an actual match. In the following step, the method

estimates these probabilities by using statistical techniques such as the EM algorithm (Dempster et al., 1977). Let $W(\gamma) = log(m(\gamma)) - log(u(\gamma))$ be the log likelihood. Records are linked based on criteria with the pre-selected thresholds T_μ and T_λ: a pair of records with comparison vector γ is linked if $W(\gamma) > T_\mu$; a pair of records is a possible link if $T_\mu \geq W(\gamma) > T_\lambda$; a pair of records is not linked if $W(\gamma) \leq T_\lambda$.

Marco Fortini and Scanu (2001) proposed a Bayesian variant of the Fellegi-Sunter method to solve the bipartite record linkage problem. That is, a record from one database can only be linked with at most one record from the other database. The Bayesian Fellegi-Sunter method allows for dependencies between attributes, which adds flexibility to it. The method consider using Dirichlet distribution for $m(\gamma)$ and $u(\gamma)$ as prior distributions. Parameters are estimated through Markov Chain Monte Carlo.

In essence, the Fellegi-Sunter method is based on a mixture model and corresponding likelihood-ratio test. Thus, it is an unsupervised method without requirements on labeled data. For our problem, we are interested in the setting when the correspondence between some of the users is revealed. When labeled data is available, the record linkage problem can also be approached in a supervised manner by approximating it to a classification problem. Pairs of records with known linkage status are treated as training data to infer the linkage status for the rest of the records. Comparison vectors of pairs of records are the features for training machine learning models. The `RecordLinkage` package (Sariyar and Borg, 2010) implemented several prevalent machine learning options such as support vector machine, bagging with classification trees, and stochastic boosting model. However, these supervised methods are specifi-

cally designed for the classification downstream task. Results obtained from simplifying a GM problem to a classification problem can be inaccurate.

4.2.3 Similarity-based learning

Similarity-based learning has wide applications in the machine learning field. It is especially useful when machine learning algorithms depend on building similarity metrics between observations. Such machine learning algorithms include the k-Nearest Neighbors (KNN) classifier (Cover and Hart, 1967), the k-means algorithm for clustering (Macqueen, 1967), as well as more complex approaches (Bai et al., 2020; López-Sánchez et al., 2019). It is also related to the record linkage problem described in the previous section in the sense that the comparison vectors used for linking records are also representations of the relatedness between two records in terms of each feature. Similarity-based learning, on the other hand, tries to solve the problem in a supervised way and is designed for specific downstream tasks.

Recently, studies on deep graph similarity learning are developing rapidly. Graph neural networks are used to learn embeddings of graph structures and similarities based on node attributes without using predefined metrics (Li et al., 2019; Ma et al., 2019). These deep strategies have outstanding performance but are hard to interpret. Thus, we will not focus on deep learning methods in this work. To the best of our knowledge, apart from deep strategies, there is no existing work on similarity-based learning specifically for the graph matching task. We will review related metric learning methods in general.

Distance metric learning and Mahalanobis distance

To begin with, let $x_i, \ldots, x_n \in \mathbb{R}^p$ be the feature vectors associated with n input data. A combined similarity score for a pair of input data is generated by computing a distance metric between them using all the features in the $p-$dimensional space. We restrict our discussion on the distance metric to the Mahalanobis distance which serves as a prototypical example. Given the feature vectors $x_i, x_j \in \mathbb{R}^p$ for two observations, the Mahalanobis distance with distance matrix M is defined as

$$d_M(x_i, x_j) = \sqrt{(x_i - x_j)^T M (x_i - x_j)}, \quad x_i, x_j \in \mathbb{R}^p. \tag{4.2}$$

The Euclidean distance is a special case of the Mahalanobis distance with M being an identity matrix. In the general case, M is a positive semi-definite matrix. An alternative way is to decompose the distance matrix as $M = L^T L$ and the Mahalanobis distance can be rewritten as

$$d_L(x_i, x_j) = \sqrt{(Lx_i - Lx_j)^T (Lx_i - Lx_j)}, \quad x_i, x_j \in \mathbb{R}^p. \tag{4.3}$$

Then the goal of the distance metric learning problem is to learn the linear transformation L that optimizes the objective function designed for the downstream task.

Neighborhood Components Analysis

Goldberger et al. (2004) proposed Neighborhood Components Analysis (NCA) for learning a linear transformation on the original feature value space to improve the accuracy of the nearest neighbor (KNN) classification algorithm.

The algorithm defines the probability of a sample i being the nearest neighbor of another sample j by applying a softmax function over the Mahalanobis distance between their feature vectors:

$$p_{ij}(L) = \frac{exp(-\|Lx_i - Lx_j\|^2)}{\sum_{k \neq i} exp(-\|Lx_i - Lx_k\|^2)}, \quad p_{ii} = 0. \tag{4.4}$$

The objective function maximizes the expected number of correctly classified nearest neighbors:

$$f(L) = \sum_{i=1}^{n} \sum_{j \in C_i} p_{ij}(L)$$

where C_i denotes the set of samples in the same class as i. The objective function can be optimized using a gradient ascent algorithm.

4.3 Metric learning for Graph Matching

Motivated by the Venmo transaction graphs alignment problem, we are interested in developing methods for learning a metric and finding an alignment of two point clouds following different distributions. We'll discuss two techniques, one being the classical Procrustes Analysis method and the other being a novel modification of the NCA method.

4.3.1 Procrustes Analysis

Procrustes analysis (PA) (Gower, 1975) is a statistical method developed for aligning shapes as represented by point clouds. Although it doesn't fall into the category of a distance metric learning problem, these two problems are related in the sense that they both try to learn a transformation on the feature vector space. PA starts with two point clouds following different distributions. For

feature vectors x_i from the first shape distribution and x_j from the second one, the distance between them is defined as

$$d_Q(x_i, x_j) = \sqrt{(x_i - Qx_j)^T(x_i - Qx_j)}. \tag{4.5}$$

The method aims at learning a transformation Q on the feature vector space of the second point cloud such that the sum of squared distance is minimized. This is realized by a sequence of steps including translation, scaling, and rotation.

4.3.2 Neighborhood Components Analysis for Graph Matching

We proposed a novel Neighborhood Components Analysis method designed for the Graph Matching task (NCA4GM) to conduct similarity-based learning when feature spaces for two point clouds are distributed differently. We also introduce an objective function more suitable for handling the graph matching problem.

In what follows, we are going to describe the method in the remaining of this section with discussions on the relationship between NCA method and PA method. In the next sections, we will further demonstrate the effectiveness of the proposed method under a random correlated feature model introduced for simulations. Finally, we will come back to the Venmo transaction graphs alignment problem and provide guidelines for it.

There are two main assumptions we make for the Venmo transaction networks alignment problem: 1) users' relative behavior tends to be consistent in August and October 2018. For example, users who use Venmo as a main plat-

form for making transactions in August would also lead to more transactions in October. 2) The distributions of certain features may vary a lot over different months. An illustrative example would be the location feature. People are more likely to travel during the summer season in August, so their transaction location during the vacation can be quite different from their living location. With a single linear transformation learned on the feature vector space for node features in both graphs, NCA can address the first assumption while failing to address the second one. This motivates us to learn separate linear transformations for feature vectors in each graph. Accordingly, we define a pseudo distance metric by generalizing the Mahalanobis distance in equation 4.3:

$$d_{AB}(x_i, x_j) = \sqrt{(Ax_i - Bx_j)^T(Ax_i - Bx_j)}, \quad x_i, x_j \in \mathbb{R}^p. \tag{4.6}$$

The distance metric $d_{AB}(x_i, x_j)$ can also be viewed as an integration of the $d_L(x_i, x_j)$ and $d_Q(x_i, x_j)$ if we apply PA first then conduct NCA on the original feature vectors for the first graph and the transformed feature vectors for the second graph. Then the correspondence between linear transformations learned by NCA4GM method and the combined method is $A = L, B = LQ$.

Figure 4·2 provides an example that visualizes the difference between NCA, PA and NCA4GM in terms of their impact on the raw data points after applying the learned linear transformations. We first generate one heart-shaped point cloud which is displayed by red circles and rotate it by 45 degrees and shrink it by 1/2 as displayed in the figures by blue triangles. For the first point cloud, data on the first dimension is independently and identically distributed as standard normal. In the second dimension, we take the absolute value of the random data generated in the first dimension and add standard normal

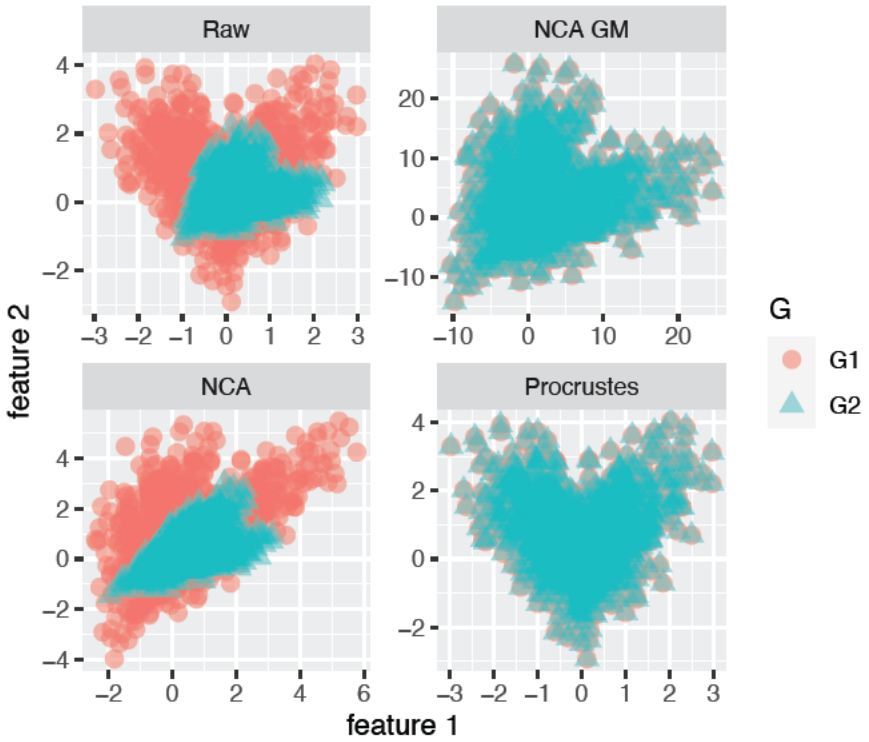


Figure 4·2: Visualization of linear transformations applied on the raw data with different methods.

random noise to it.

Another limitation of the NCA algorithm when applying it to the graph matching downstream task is that it is designed specifically for the KNN task. This is reflected in learning a single linear transformation on the feature space. Also, for a fixed observation i, the objective function is designed such that the Mahalanobis distance between a targeted pair of observations i and j is compared with the Mahalanobis distance between i and all the other candidate observations except j. In the graph matching framework, each observation would be a pair of candidate matches. For a fixed target match (i, j), its candidate matched pairs include all of $(i'j), i' \neq i$ and $(i, j'), j' \neq j$. We adopt

the idea of designing the objective function from the original NCA and adjust it to the graph matching task. We instead compare the Mahalanobis distance between a target pair with the Mahalanobis distance between all its candidate pairs. In this way, we aim at ensuring that each pair of correctly matched nodes has the smallest Mahalanobis distance between their corresponding feature vectors on the transformed space compared with their candidate pairs.

We introduce the probability of node $i \in V_1$ being a mutual nearest neighbor of node $j \in V_2$ using a softmax over all the candidate pairs in the linear transformed space.

$$p_{ij}(A, B) = \frac{exp(-\|Ax_i - Bx_j\|^2)}{\sum_{i' \in V_1, i' \neq i} exp(-\|Ax_{i'} - Bx_j\|^2) + \sum_{j' \in V_2} exp(-\|Ax_i - Bx_{j'}\|^2)} \tag{4.7}$$

We aim at maximizing the number of seeded node pairs getting correctly matched, i.e., the probability of a particular pair of seeded nodes being a mutual neared neighbor is higher than the other candidate pairs under the definition in Eq. 4.7:

$$f(A, B) = \sum_{(i,j) \in S} P_{ij}(A, B) \tag{4.8}$$

where $S \subset V_1 \times V_2$ denotes the seed set.

Let $E_{ij} = \exp(-\|Ax_i - Bx_j\|^2)$. We can then use stochastic gradient ascent Algorithm 8 to learn A, B where the gradients of the objective function

$f(A, B)$ with respect to A, B are

$$\frac{\partial f(A,B)}{\partial A} = \sum_{(i,j)\in S} \frac{\partial P_{ij}(A,B)}{\partial A} \qquad \frac{\partial f(A,B)}{\partial B} = \sum_{(i,j)\in S} \frac{\partial P_{ij}(A,B)}{\partial B}$$

$$\frac{\partial P_{ij}(A,B)}{\partial A} = -2P_{ij}(x_{ij} \cdot x_i^T - \sum_{i'\in V_1, i'\neq i} \frac{E_{i'j}}{\sum_{j'\in V_2} E_{ij'} + \sum_{i'\in V_1, i'\neq i} E_{i'j}} \cdot x_{i'j} \cdot x_{i'}^T) - \sum_{j'\in V_2} \frac{E_{ij'}}{\sum_{j'\in V_2} E_{ij'} + \sum_{i'\in V_1, i'\neq i} E_{i'j}} \cdot x_{ij'} \cdot x_i^T)$$

$$\frac{\partial P_{ij}(A,B)}{\partial B} = 2P_{ij}(x_{ij} \cdot x_j^T - \sum_{i'\in V_1, i'\neq i} \frac{E_{i'j}}{\sum_{j'\in V_2} E_{ij'} + \sum_{i'\in V_1, i'\neq i} E_{i'j}} \cdot x_{i'j} \cdot x_j^T) - \sum_{j'\in V_2} \frac{E_{ij'}}{\sum_{j'\in V_2} E_{ij'} + \sum_{i'\in V_1, i'\neq i} E_{i'j}} \cdot x_{ij'} \cdot x_{j'}^T)$$

Algorithm 8 NCA using Stochastic Gradient Ascent

Input : Feature vectors for G_1: $x_{i_1}, ..., x_{i_n}$, feature vectors for G_2: $x_{j_1}, ..., x_{j_n}$, initial linear transformations A_0, B_0, seed set S, epoch $niters$, learning rate η

Output: A_{new}, B_{new}

Set $k = 1$;

while $k < niters$ **do**

  for $(i, j) \in S$ **do**

    $A_k \leftarrow A_{k-1} + \eta \cdot \frac{\partial f(A_{k-1}, B_{k-1})}{\partial A}$;

    $B_k \leftarrow B_{k-1} + \eta \cdot \frac{\partial f(A_{k-1}, B_{k-1})}{\partial B}$;

  end

  $k \leftarrow k + 1$;

end

4.4 Results

4.4.1 Simulations

To demonstrate the effectiveness of our proposed NCA4GM algorithm adapted for aligning two correlated feature spaces with diverse distribution, We will first introduce a random correlated feature model for generating simulated data. Then we illustrate the algorithm performance under different simulated data settings.

Random correlated feature Model

Let $y_{1i}, y_{2i}, i = 1, \ldots, n$ be random feature vectors for corresponding nodes in graphs G_1 and G_2. We introduce the following scheme for generating pairs of high-dimensional feature vectors for simulations. First, we generate a feature vector x_i for node i following a normal distribution with variance covariance matrix Σ_X. Without loss of generality, we assume zero mean. Then we apply a rotation to the feature vector to get a feature vector for node i in graph G_2. Finally, we add some normally distributed random noise to the feature vector pair and get $y_{1i} = x_i + \epsilon_{1i}$, $y_{2i} = Rx_i + \epsilon_{2i}$. For each node $i = 1, \ldots, n$,

$$y_{1i} \overset{i.i.d.}{\sim} N(0, \Sigma_X + \Sigma_\epsilon), \quad y_{2i} \overset{i.i.d.}{\sim} N(0, RR^T\Sigma_X + \Sigma_\epsilon).$$

The sampled feature vectors for each node are independent unless they are associated with corresponding nodes in two graphs with cross-covariance $Cov(y_{1i}, y_{2i}) = \Sigma_X R^T$.

Bayes Classifier under the random correlated feature model

In addition to the NCA algorithm designed specifically for the KNN task and the Prospectus Analysis algorithm, we also include the Bayes classifier as a baseline which indicates the best possible performance when simplifying the GM problem as a classification task. Theoretically, the Bayes classifier minimizes the error rate among all the classifiers under the condition that the distribution of samples of each class is properly addressed. In simulations, we sample feature data with known distributions which enables us to come up with the Bayes classifier in closed form.

To begin with, assume we have a sample of feature vectors $y_{1i}, y_{2i} \in \mathbb{R}^p, i = 1, \ldots, n$. To approximate the original problem as a classification problem, we take each pair of feature vectors for node $i \in G_1$ and node $j \in G_2$, i.e. (y_{1i}, y_{2j}) for all $i, j = 1, \ldots, n$ as input data. Therefore, there n^2 observations in total which can be categorized into two classes c_1 and c_2. Let's assume $(y_{1i}, y_{2j}) \in c_1$ if $i = j$, corresponding to the case of being a correct match; $(y_{1i}, y_{2j}) \in c_2$ if $i \neq j$, corresponding to the case of an incorrect match. Thus there are n observations that belong to class 1 and $n^2 - n$ observations that belong to class 2.

Let's denote the difference between two observations, y_{1i} and $R^{-1}y_{2j}$ by a random variable y. For both class 1 and class 2 y is also normally distributed with zero mean, i.e. $\mu_1 = \mu_2 = 0$, and variance covariance matrices are

$$\Sigma_1 = (I + R^{-1}R^{-T})\Sigma_\epsilon \quad \text{when } i = j, \tag{4.9}$$

and

$$\Sigma_2 = (I + R^{-1}R^{-T})\Sigma_\epsilon + 2\Sigma_X \quad \text{when } i \neq j. \tag{4.10}$$

Not surprisingly, the feature differences on the transformed space for an incorrect match are distributed with a larger variance.

We classify observation pairs based on their difference in the transformed space. Since the variance-covariance matrices are different in two classes, we use Quadratic Discriminant Analysis (QDA) to derive the Bayes classifier. QDA assigns an observation to class $k \in \{1, 2\}$ with the largest δ_k defined as:

$$\delta_k(y) = -\frac{1}{2}y^T\Sigma_k^{-1}y + y^T\Sigma_k^{-1}\mu_k - \frac{1}{2}\mu_k^T\Sigma_k^{-1}\mu_k - \frac{1}{2}log|\Sigma_k| + log\pi_k$$

where π_k represents the prior probability that an observation belongs to the k^{th} class. Therefore, $\pi_1 = \frac{1}{n}$ and $\pi_2 = \frac{n-1}{n}$. Since $\mu_1 = \mu_2 = 0$, δ_k can be simplified as

$$\delta_k(y) = -\frac{1}{2}y^T\Sigma_k^{-1}y - \frac{1}{2}log|\Sigma_k| + log\pi_k$$

Then the decision boundary is achieved at $\delta_1 = \delta_2$, thus we get the following decision boundary

$$\frac{1}{2}(\Sigma^{1/2}y)^T(\Sigma^{1/2}y) + \frac{1}{2}log\frac{|\Sigma_2|}{|\Sigma_1|} - log(n-1) = 0$$

where $\Sigma = \Sigma_2^{-1} - \Sigma_1^{-1}$. We are looking for linear transformations on the feature spaces so that feature vectors are spherically distributed instead of elliptically distributed so that the decision boundary is spherical. Therefore, we obtain linear transformations for feature vectors belonging to G_1 and G_2 as

$$A = (\Sigma_1^{-1} - \Sigma_2^{-1})^{1/2}, \quad B = AR^{-1} \tag{4.11}$$

with Σ_1 and Σ_2 defined in 4.9 and 4.10.

Simulations with spherical covariance

In the first simulation example, we sample a pair of correlated feature data following the random data generating model described in above. We assume that the features are uncorrelated before rotation. We consider sampling $n = 200$ pairs of feature vectors from a feature space of dimension $d = 3$. Let variance-covariance matrices of feature vectors in G_1 and noise be diagonal matrices with diagonal values of Σ_X and Σ_ϵ being 1 and 0.01. For the rotation matrix R, we first sample a $d \times d$ matrix R' which entry-wise follows the standard normal distribution. Let the singular value decomposition of R' be denoted as $R' = U\Sigma_{svd}V^T$, where U is a $d \times d$ matrix containing the orthonormal eigenvectors of $R'R'^T$, V is a $d \times d$ matrix of the orthonormal eigenvectors of $R'^T R'$, and Σ_{svd} is a $d \times d$ diagonal matrix with singular values on its diagonal. Then we take the rotation matrix $R = \lambda UV^T$ and vary $\lambda \in \{.2, .5, 1, 1.5, 2\}$. In this way, we vary the scale of the second feature vector samples.

Figure 4·3 displays a performance comparison between the original NCA method, our proposed NCA method adjusted for the graph matching problem, and the Bayes classifier based on the true sampling distributions. Each panel shows an averages from 100 Monte Carlo replicates and the error bars represent 95% confidence intervals. The figure on the left shows the matching accuracy obtained from applying the Hungarian method (Kuhn, 1955) on the cost matrix where ij-th entry represents the distance between feature i in G_1 and feature j in G_2 on the transformed feature space using the linear

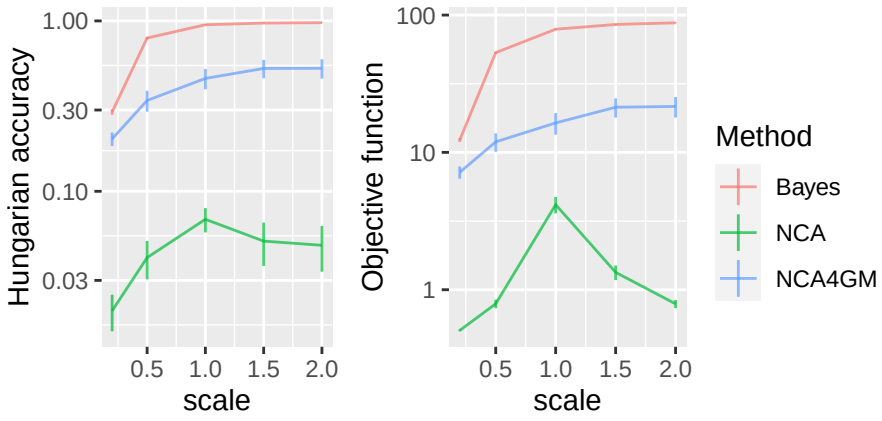


Figure 4·3: Performance comparison between NCA, NCA4GM and Bayes method as a function of the scale of feature vectors of G_2 with independent noise.

transformation matrices for different methods. Formally speaking, the ij-th entry is defined as $\|Ay_{1i} - By_{2j}\|^2$. For the original NCA method, A and B refer to the single linear transformation learned by the method. For the Bayes method, A and B are closed-form solutions in equation 4.11 and calculated using the simulation parameters. The figure on the right shows the objective function values for the NCA4GM model as shown in equation 4.8. Similarly, for different methods, the objective function is calculated using the corresponding learned or closed-form linear transformations. The objective function depicts the expected number of correctly matched nodes based on their feature embeddings.

As it is shown in figures 4·3, our proposed method improves a lot compared with the original NCA method. With a random rotation applied to the first feature vector space, NCA can hardly recover the underlying alignment with a single linear transformation. In addition, as λ gets larger, our proposed method is performing better in terms of both metrics. The intuition is that

as λ grows larger, the variations of the generated feature plus noise vectors in G_2 would be increasingly dominated by the feature vectors, and noise would play a minor role. On the other hand, we don't see a similar trend for NCA and the performance of NCA is not monotonic in terms of λ. NCA performs the best when $\lambda = 1$ and two feature vector spaces share the same scale. Note that we include the Bayes method in the figures to indicate the performance of an optimal classifier. However, in most real data cases it is not possible to recover the true underlying distributions and additional assumptions and analysis may be needed to obtain estimated distributions.

Simulations with non-spherical covariance

In the previous example, we assume independent noise and consider the impact of the scale of features in the second feature vector space under the condition that the scale of features in G_1 is fixed. In the next simulation example, we target at investigating whether or not breaking the independence setting would make an impact on the performance of different methods. Let us still fix the number of sampled observations to be $n = 200$, the dimension of feature space to be $d = 3$, and the variance-covariance matrix of feature vectors in G_1, Σ_X be a 3×3 diagonal matrix with diagonal terms equal to 1. We follow the same steps for generating rotation matrices as described in the previous example and fix the scale of the second feature space to be $\lambda = 2$. Noise terms are identically distributed as $N(0, 0.01)$. Let us denote the covariance between any two noise entries by cov_ϵ, and we vary $cov_\epsilon \in \{0, 0.000625, 0.0025, 0.005625, 0.01\}$.

Figures 4·4 display an average of 100 replicates for each value of cov_ϵ in terms of the same metrics as in the previous example. We also observe a

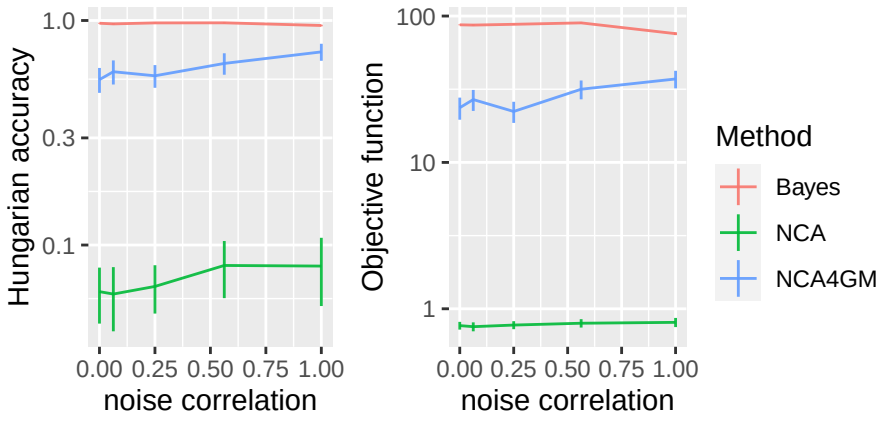


Figure 4·4: Performance comparison between NCA, NCA4GM and Bayes method as a function of correlation between features.

significant enhancement in terms of both matching accuracy on the cost matrix and the expected number of correct matches calculated using the learned and closed form linear transformations after applying the NCA method adjusted for feature vectors from two point clouds. Additionally, the proposed method performs better and its performance approaches the Bayes classifier as we increase the correlation between noise terms.

4.4.2 DARPA Venmo transaction graphs

Recall that the ultimate goal is to align Venmo users from two different months using the Venmo transactions between them and additional information associated with users and their transactions. In section 4.1, we reduce the problem to a metric learning problem defined on the DARPA Venmo transaction graphs. In particular, we target building a similarity measure by integrating eight user features as defined in table 4.1. Then we can either solve a LAP directly on the similarity scores or utilize them as additional input to the graph matching

algorithms to obtain an alignment between users in the Venmo transaction graphs.

Note that with a d-dimensional feature space, the original NCA method, the PA, and the proposed NCA4GM method would need to estimate d^2, d^2, and $2 \times d^2$ parameters respectively. Additionally, because there are only $n = 2846$ users in each graph, it is necessary to reduce the dimension of feature space to prevent overfitting. Therefore, in the feature engineering step, we conduct a principal component analysis on the 1024-dimensional transaction topic feature associated with two graphs separately. We take the first 12 principal components as approximates to the original information in word to vector embeddings. The proportions of variance explained by the first 12 principal components are 47.0% and 47.1% in two graphs. After that, the dimension of feature space is reduced to $d = 20$. Notably, the location feature consists of two dimensions, longitude, and latitude.

Then we apply the proposed NCA4GM algorithm, NCA algorithm, and PA to the reduced feature space. We randomly split 2846 users into 30% for training and 70% for testing. Linear transformations are learned on the training set and we build similarity scores on the transformed feature space for matching the rest of the users in the test set. For a pair of users $(i, j), i \in V_1$ and $j \in V_2$ in the test set, their corresponding integrated similarity score using the linear transformations A, B obtained from the NCA4GM method is defined as:

$$S_{ij} = \|Ax_i - Bx_j\|^2 \quad i \in V_1, j \in V_2.$$

The similarity scores for the other two methods can also be defined accordingly.

Table 4.2 summarizes the matching results of the different methods by

applying the Hungarian method to their corresponding similarity scores for users in the test set directly. The results are averages of 100 Monte Carlo replicates with random splits of the entire Venmo users into the training set and the test set.

Table 4.2: Matching accuracy on Venmo transaction graphs.

Method	Hungarian accuracy	Accuracy sd
NCA GM	0.0026	0.0011
NCA	0.0023	0.0011
PA	0.0025	0.0010
Chance	0.00037	0

Note that although all three methods listed in the table only match a small number of users correctly, their matching results are significantly better than random matching. Among the candidate metric learning methods, our proposed NCA method adapted for the GM problem matches slightly more Venmo users correctly on average. However, the differences between NCA for GM, NCA and PA are not significant. When we run paired t-test on NCA for GM versus NCA, and NCA for GM versus PA, the corresponding p-values are 0.1097 and 0.348 respectively. However, in light of the dramatic improvement shown in the simulated data in section 4.4.1, the proposed method is still a good candidate when solving a graph matching task. To further verify its performance, we plan to test its performance on other attributed graphs in future work. Additionally, to enhance the matching performance, we can also resort to graph matching algorithms and utilize the combined similarity scores as additional prior information. In this way, we can exploit the topological information more thoroughly.

Chapter 5

Conclusions

This work focuses on developing methodologies and practical tools to analyze the graph matching problem with different types of prior information. This research consists of three components; first, an software development component, wherein a R package is developed for common GM algorithms under generalized settings and steps for the analysis of GM with standardized code, data, and documentation; second, a methodology component, focused on developing a convenient and scalable GM algorithm using the neighborhood topological information incorporating seeds, and a mathematical framework for analyzing the performance of the proposed algorithm; finally, an application component, targeted at developing similarity-based learning methods for integrating multiple attributes associated with graph nodes and edges, and applying methods to solving a real-world user de-anonymization problem on the Venmo transaction networks.

For the first software development component, we detail the methods and usage of the R package `iGraphMatch` for finding and assessing an alignment between the vertex sets of two edge-correlated graphs. The package implements common steps for the analysis of graph matching: seamless matching of generalized graphs, evaluation of matching performance, and visualization. For each of the graph matching methodologies, we provide versatile options for

the form of input graphs and the specification of available prior information. Through the discussion in Section 2.2, we demonstrate the broad functionality and flexibility of the package by analyzing diverse graph matching problems on real data step by step. The package also provides tools for simulating correlated graphs which can be used in the development and enhancement of graph matching methods. Additionally, we provide an analysis of the trade-off between efficiency and effectiveness for the GM algorithms in the package and give useful insights into the relative preference in the graph matching methodologies under different scenarios.

One of the biggest challenges for graph matching is evaluating the quality of a match, especially at the vertex level. This has received minimal attention in the previous literature. In the first work, we provide measures of goodness of matching on the vertex level and demonstrate their effectiveness empirically. These baseline methods implement a permutation testing framework for assessing matches that can be readily extended to other metrics.

For the GM methodology component, we proposed a scalable GM algorithm that starts with a handful of seeds and is free of hyperparameters. The proposed MNN algorithm can obtain a matched set with high precision within a short amount of time. Instead of always growing a mapping to the entire graphs, the MNN algorithm stops when there is no sufficient information supporting further matches. This high-quality matched set can be utilized as a starting point that is closer to the true alignment for other algorithms that requires an initialization like the FW methodology or algorithms that incorporates similarity scores. Also, this algorithm can be further optimized in terms of computational efficiency by running in parallel when extracting

mutual nearest neighbors in each iteration. Additionally, we mathematically formulate the proposed algorithm by a series of events with probability distributions and provide conditions for not introducing erroneous matches. The proposed method's high precision and computational efficiency are demonstrated through simulated data and real-world networks.

For the last component, we apply similarity-based learning to the real-world Venmo users' de-anonymization problem. The original Venmo transaction graphs are knowledge-based with various side information on users and their Venmo transactions. We adjust the Neighborhood Component Analysis method to adapt for the graph matching problem and use it to integrate multiple prior information into a similarity score. Assume that the feature vector space for users involved in Venmo transactions from two different varies, the proposed NCA4GM method learns separate linear transformations for two feature vector spaces. The assumption is reasonable since users' behavior can differ a lot in terms of certain features, such as transaction location and transaction topic due to seasonality. Eventually, we generate an alignment between Venmo users involved in transactions from two different months by solving a LAP on the similarity scores.

Methods for graph matching are still under active development. We plan to include other novel methods into the package as the field continues to develop. As for the MNN algorithm, we hope to improve upon the current theoretical result on the method's high matching precision. Also, we plan to provide theoretical guarantees on the number of matched nodes. Additionally, as it is shown in section 4, matching the knowledge base Venmo transaction graphs with combined similarity scores is quite challenging. As for future

work, potential ways to enhance the matching result include updating the objective function for the NCA4GM method so that we can also optimize the topological agreement, and extract more features based on the node attributes and edge attributes in the knowledge-based graphs.

Appendix A

Proof of propositions

Proposition A.0.1. *Let* $f(r,l,k)$ *denote*

$$f(r,l,k) = P(|J| \leq r)P(|H| \leq r) \cdot [\sum_{x=0}^{r} P(|X| = x) \cdot \sum_{y=0}^{r-x} P(|Y| = y)P(|W| \leq r-x-y)P(|V| \leq r-x)]^{n-2}, \tag{A.1}$$

where the random variables are defined in table 3.1. Then $f(r,l,k)$ *monotonically increases as* l,k *decrease and* r *increases.*

Proof. For the first two probability terms to the right of equation A.1,

$$P(|J| \leq r) \cdot P(|H| \leq r) = P(Bin(l,s) \leq r) \cdot P(Bin(k,s) \leq r).$$

It is obvious that the product of two left tail Binomials increases as l,k decreases and as r increases.

The product of probabilities inside the bracket is summed over r terms in terms of x. As r increases, we are adding more non-negative probabilities to the summation and thus, making it larger. Finally, recall that random variables $|W| = Bin(l-r, ps)$ and $|V| = Bin(k-x-y, ps)$. For fixed x, y and r, the left tail binomials for $|W|$ and $|V|$ both increase as we decrease l,k.

when decreasing l,k and increasing r, since both the product of the first two probability terms and the remaining terms in the bracket increase, $f(r,l,k)$ also increases.

□

Proposition A.0.2. *Suppose there are* $|L| = l_0$ *seeded nodes connected to* u

in G_1, and $|K|= k_0$ seeded nodes connected to v in G_2. Assume $p < 0.6, 0 < \epsilon_1, \epsilon_2 < 1/4$, then

$$P(|J|\leq r_0) \leq exp(-\frac{\alpha p s^2(\frac{3}{4}-\frac{5}{4}p)^2}{2})$$

and

$$P(|H|\leq r_0) \leq exp(-\frac{\alpha p s^2(\frac{3}{4}-\frac{5}{4}p)^2}{2}).$$

Proof. We apply the Chernoff bound for left tail Binomial,

$$P(|J|\leq r_0) = P(Bin(l_0, s) \leq r_0) \leq exp(-\frac{(1-\frac{r_0}{l_0 s})^2 \cdot l_0 s}{2})$$

Then, rewrite the above upper bounds by plugging in $r_0 = (1+\epsilon_1)\alpha p^2 s^2$, $l_0 = (1-\epsilon_2)\alpha p s$ and simplification, we get

$$P(|J|\leq r_0) \leq exp(-\frac{(1-\frac{(1+\epsilon_1)p}{1-\epsilon_2})^2 \cdot (1-\epsilon_2)\alpha p s^2}{2})$$

Let $\phi = (1-\frac{(1+\epsilon_1)p}{1-\epsilon_2})^2 \cdot (1-\epsilon_2)$, then,

$$\phi = \frac{(1-\epsilon_2-(1+\epsilon_1)p)^2}{1-\epsilon_2} > (1-\epsilon_2-(1+\epsilon_1)p)^2 > (\frac{3}{4}-\frac{5}{4}p)^2$$

under the assumption that $\epsilon_1, \epsilon_2 \leq 1/4$. Note that in order to apply the Chernoff bound, $\phi < 1$ needs to hold. Accordingly, edge probability p needs to be smaller than 0.6. Thus, subsequent conditions only work in the case when two graphs are not too dense. Plug in the lower bound of ϕ and we get

$$P(|J|\leq r_0) \leq exp(-\frac{\alpha p s^2(\frac{3}{4}-\frac{5}{4}p)^2}{2})$$

Similarly,

$$P(|H|\leq r_0) = P(Bin(l_0, s) \leq r_0) \leq exp(-\frac{\alpha p s^2(\frac{3}{4}-\frac{5}{4}p)^2}{2})$$

□

Proposition A.0.3. *Let* $x_0 = (1-\delta_1)rps^2$, $y_0 = (1-\delta_2)(r-x)ps$. *For* $0 < \epsilon_1 \leq 1$*:*

$$\sum\sum_{\{x\geq x_0, y\geq y_0\}^c} P(|X| = x)\cdot P(|Y| = y) \leq exp(-\frac{(1+\epsilon_1)(log\alpha)^2 ps^2}{2}) + exp[-\frac{(1+\epsilon_1)(log\alpha)^2 ps}{2} + (e^{\frac{(log\alpha)^2}{2\alpha ps}} - 1)(1+\epsilon_1)\alpha p^3 s^4]$$

Proof. Let's denote $f(x, y|r, p, s) = P(Bin(r, ps^2) = x)\cdot P(Bin(r-x, ps) = y)$, then the summation of $P(Bin(r, ps^2) = x)\cdot P(Bin(r-x, ps) = y)$ over the compliment set of $\{x \geq x_0, y \geq y_0\}$ can be rewritten as

$$\sum\sum_{\{x\geq x_0, y\geq y_0\}^c} f(x, y|r, p, s) = \sum_{x=0}^{x_0}\sum_{y=0}^{r-x} f(x, y|r, p, s) + \sum_{x=x_0}^{r}\sum_{y=0}^{y_0} f(x, y|r, p, s)$$

For the first summation term, by Chernoff bound:

$$\sum_{x=0}^{x_0}\sum_{y=0}^{r-x} P(Bin(r, ps^2) = x)\cdot P(Bin(r-x, ps) = y) = P(Bin(r, ps^2) \leq x_0) \leq exp(-\frac{\delta_1^2 rps^2}{2})$$

For the second summation term,

$$\sum_{x=x_0}^{r}\sum_{y=0}^{y_0} P(Bin(r,ps^2)=x)\cdot P(Bin(r-x,ps)=y)$$

$$= \sum_{x=x_0}^{r} P(Bin(r,ps^2)=x)\cdot P(Bin(r-x,ps)\leq y_0)$$

$$\leq \sum_{x=x_0}^{r} P(Bin(r,ps^2)=x)\cdot exp(-\frac{(1-\frac{y_0}{(r-x)ps})^2(r-x)ps}{2})$$

$$= exp(-\frac{\delta_2^2 rps}{2})\cdot\sum_{x=x_0}^{r} P(Bin(r,ps^2)=x)\cdot exp(\frac{\delta_2^2 psx}{2})$$

$$\leq exp(-\frac{\delta_2^2 rps}{2})\cdot\sum_{x=0}^{r} P(Bin(r,ps^2)=x)\cdot exp(\frac{\delta_2^2 psx}{2})$$

$$= exp(-\frac{\delta_2^2 rps}{2})\cdot\mathbb{E}[e^{\frac{\delta_2^2 ps}{2}X}]$$

$$= exp(-\frac{\delta_2^2 rps}{2})\cdot(1-ps^2+ps^2e^{\frac{\delta_2^2 ps}{2}})^r$$

$$\xrightarrow{r\to\infty} exp[-\frac{\delta_2^2 rps}{2}+(e^{\frac{\delta_2^2 ps}{2}}-1)rps^2].$$

The first inequality uses the Chernoff bound for left tail binomial. The second equality uses moment generating function $M_X(t)$ for $X\sim Bin(r,ps^2)$ and $t=\frac{\delta_2^2 ps}{2}$. The last line applies $lim_{r\to\infty}(1+\frac{x}{r})^2\to e^x$, where $x=(e^{\frac{\delta_2^2 ps}{2}}-1)rps^2$.

Let us combine two parts and substitute r in the upper bounds by $(1+\epsilon_1)\alpha p^2 s^2$. Also, let us assume

$$\delta_1=\delta_2=\frac{log\alpha}{\sqrt{\alpha}ps}.$$

Then,

$$\sum\sum_{\{x\geq x_0, y\geq y_0\}^c} P(|X|=x)\cdot P(|Y|=y)\leq exp(-\frac{(1+\epsilon_1)(log\alpha)^2 ps^2}{2})+$$
$$exp[-\frac{(1+\epsilon_1)(log\alpha)^2 ps}{2}+(e^{\frac{(log\alpha)^2}{2\alpha ps}}-1)(1+\epsilon_1)\alpha p^3 s^4]$$

□

Proposition A.0.4. *Assume* $0 < \epsilon_1 < \frac{1}{p^2 s^2} - 1, 0 < \epsilon_2 < 1$. *Let* $g(r,l,k) = P(Bin(\alpha, ps) = l) \cdot P(Bin(\alpha, ps) = k) \cdot P(Hyper(\alpha, l, k) = r)$, *then*

$$\sum\sum\sum_{\{r\le r_0, l\ge l_0, k\ge k_0\}^c} g(r,l,k) \le exp(-\frac{1}{16}\alpha p^3 s^3 \epsilon_1^2) + 2exp(-\frac{1}{48}\alpha ps\epsilon_1^2) + 2\cdot exp(-\frac{\alpha ps\epsilon_2^2}{2})$$

Proof. Firstly, summing $g(r,l,k)$ over the compliment set of $\{r \le r_0, l \ge l_0, k \ge k_0\}$ can be rewritten as:

$$\sum\sum\sum_{\{r\le r_0, l\ge l_0, k\ge k_0\}^c} g(r,l,k) = \sum_{l=l_0}^{\alpha}\sum_{k=k_0}^{\alpha}\sum_{r=r_0}^{min(l,k)} g(r,l,k) + \sum_{l=1}^{l_0}\sum_{k=k_0}^{\alpha}\sum_{r=1}^{min(l,k)} g(r,l,k) + \sum_{l=1}^{\alpha}\sum_{k=1}^{k_0}\sum_{r=1}^{min(l,k)} g(r,l,k). \tag{A.2}$$

We will provide upper bounds to each summation term to the right of equation A.2.

For the first summation term, let's split l, k at $(1+\epsilon_3)b$, $\epsilon_3 > 0$ such that $(1+\epsilon_1)a$ is greater than the mean of the hypergeometric distribution $Hyper(\alpha, l, k)$ for all the possible values of $l, k \in [(1-\epsilon_2)b, (1+\epsilon_3)b]$, i.e.,

$$((1+\epsilon_3)b)^2/\alpha < (1+\epsilon_1)a$$

and thus,

$$0 < \epsilon_3 < \sqrt{1+\epsilon_1} - 1 \tag{A.3}$$

Then we rewrite $\sum_{l=l_0}^{\alpha}\sum_{k=k_0}^{\alpha}\sum_{r=r_0}^{min(l,k)} g(r,l,k)$ as a summation of three terms:

$$\sum_{l=l_0}^{\alpha}\sum_{k=k_0}^{\alpha}\sum_{r=r_0}^{min(l,k)} g(r,l,k) = \sum_{l=l_0}^{(1+\epsilon_3)b}\sum_{k=k_0}^{(1+\epsilon_3)b}\sum_{r=r_0}^{min(l,k)} g(r,l,k)+ \tag{A.4}$$

$$\sum_{l=l_0}^{\alpha}\sum_{k=(1+\epsilon_3)b}^{\alpha}\sum_{r=r_0}^{min(l,k)} g(r,l,k)+ \tag{A.5}$$

$$\sum_{l=(1+\epsilon_3)b}^{\alpha}\sum_{k=k_0}^{(1+\epsilon_3)b}\sum_{r=r_0}^{min(l,k)} g(r,l,k) \tag{A.6}$$

$$\begin{aligned}
A.4 &\leq \sum_{l=l_0}^{(1+\epsilon_3)b} P(|L|=l)\cdot \sum_{k=k_0}^{(1+\epsilon_3)b} P(|K|=k)\cdot P(Hyper(\alpha,l,k)\geq(1+\epsilon_1)a) \\
&\leq \sum_{l=l_0}^{(1+\epsilon_3)b} P(|L|=l) \sum_{k=k_0}^{(1+\epsilon_3)b} P(Bin(\alpha,ps)=k)\cdot exp(-\frac{2((1+\epsilon_1)a-\frac{kl}{\alpha})^2}{k}) \\
&\leq \sum_{l=l_0}^{(1+\epsilon_3)b} P(|L|=l) exp(-\frac{2((1+\epsilon_1)a-\frac{(1+\epsilon_3)bl}{\alpha})^2}{(1+\epsilon_3)b}) \sum_{k=k_0}^{(1+\epsilon_3)b} P(Bin(\alpha,ps)=k) \\
&\leq exp(-\frac{2((1+\epsilon_1)a-\frac{((1+\epsilon_3)b)^2}{\alpha})^2}{(1+\epsilon_3)b})\cdot \sum_{l=l_0}^{(1+\epsilon_3)b} P(Bin(\alpha,ps)=l)\cdot 1 \\
&\leq exp(-\frac{2((1+\epsilon_1)a-\frac{((1+\epsilon_3)b)^2}{\alpha})^2}{(1+\epsilon_3)b})
\end{aligned}$$

The second inequality uses the Hoeffding's inequality. For the third inequality, $\phi(k) = exp(-\frac{2((1+\epsilon_1)a-\frac{kl}{\alpha})^2}{k})$ monotonically increases as $k \in [(1-\epsilon_2)b,(1+\epsilon_3)b)]$ increases, for all $l \in [(1-\epsilon_2)b,(1+\epsilon_3)b)]$. Thus, $\phi(k) \leq \phi((1+\epsilon_3)b)$. Similarly, for the fourth inequality, $\psi(l) = exp(-\frac{2((1+\epsilon_1)a-\frac{(1+\epsilon_3)bl}{\alpha})^2}{(1+\epsilon_3)b})$ increases as $l \in [(1-\epsilon_2)b,(1+\epsilon_3)b)]$ increases.

$$
\begin{aligned}
A.5 &= \sum_{l=l_0}^{\alpha} P(|L|=l) \sum_{k=(1+\epsilon_3)b}^{\alpha} P(|K|=k) \sum_{r=r_0}^{min(l,k)} P(Hyper(\alpha,l,k)=r) \\
&\leq \sum_{l=l_0}^{\alpha} P(Bin(\alpha,ps)=l) \sum_{k=(1+\epsilon_3)b}^{\alpha} P(Bin(\alpha,ps)=k)\cdot 1 \\
&= P(Bin(\alpha,ps)\geq(1-\epsilon_2)b)\cdot P(Bin(\alpha,ps)\geq(1+\epsilon_3)b) \\
&\leq 1\cdot exp(-\frac{\alpha ps\epsilon_3^2}{3}) = exp(-\frac{\alpha ps\epsilon_3^2}{3})
\end{aligned}
$$

The last inequality uses the Chernoff bound. Similarly:

$$
\begin{aligned}
A.6 &= \sum_{l=(1+\epsilon_3)b}^{\alpha} P(|L|=l) \sum_{k=k_0}^{(1+\epsilon_3)b} P(|K|=k) \sum_{r=r_0}^{min(l,k)} P(|R|=r) \\
&\leq exp(-\frac{\alpha ps\epsilon_3^2}{3})\cdot 1\cdot 1 = exp(-\frac{\alpha ps\epsilon_3^2}{3})
\end{aligned}
$$

To sum up,

$$
\begin{aligned}
\sum_{l=l_0}^{\alpha}\sum_{k=k_0}^{\alpha}\sum_{r=r_0}^{min(l,k)} g(r,l,k) \leq & exp(-2\alpha p^3 s^3 \frac{[(1+\epsilon_1)-(1+\epsilon_3)^2]^2}{1+\epsilon_3}) + \\
& 2\cdot exp(-\frac{\alpha ps\epsilon_3^2}{3}),
\end{aligned}
$$

where $0<\epsilon_3<\sqrt{1+\epsilon_1}-1$. Let $\epsilon_3=\sqrt{1+\epsilon_1}-1-a$ and $\epsilon_3\geq a$. Accordingly,

$$
0<a\leq\frac{1}{2}(\sqrt{1+\epsilon_1}-1).
$$

Then,

$$
\begin{aligned}
(1+\epsilon_1)-(1+\epsilon_3)^2 &= (1+\epsilon_1)-(\sqrt{1+\epsilon_3}-a)^2 \\
&= a(2\sqrt{1+\epsilon_1}-a) > a\sqrt{1+\epsilon_1}
\end{aligned}
$$

and subsequently,

$$exp(-2\alpha p^3 s^3 \frac{[(1+\epsilon_1)-(1+\epsilon_3)^2]^2}{1+\epsilon_3}) < exp(-2\alpha p^3 s^3 \frac{(a\sqrt{1+\epsilon_1})^2}{1+\epsilon_3}) < exp(-2\alpha p^3 s^3 a^2(1+\epsilon_1))$$

where the first inequality applies the former bound and the second inequality assumes $\epsilon_3 < 1$. Additionally, since $\epsilon_3 > a$, then

$$exp(-\frac{\alpha ps\epsilon_3^2}{3}) < exp(-\frac{\alpha psa^2}{3})$$

Since both upper bounds decrease as a increases, we combine the above two upper bounds and take $a = \frac{1}{4}\epsilon_1$, then

$$\sum_{l=l_0}^{\alpha}\sum_{k=k_0}^{\alpha}\sum_{r=r_0}^{min(l,k)} g(r,l,k) \leq exp(-\frac{1}{16}\alpha p^3 s^3 \epsilon_1^2) + 2exp(-\frac{1}{48}\alpha ps\epsilon_1^2).$$

For the second and last summation terms in Eq. A.2,

$$\sum_{l=1}^{l_0}\sum_{k=1}^{\alpha}\sum_{r=1}^{min(l,k)} g(r,l,k) = \sum_{l=1}^{l_0} P(Bin(\alpha,ps)=l)\cdot\sum_{k=1}^{\alpha} P(Bin(\alpha,ps)=k)\cdot \sum_{r=1}^{min(l,k)} P(h(\alpha,l,k)=r) \leq exp(-\frac{\alpha ps\epsilon_2^2}{2})$$

The last inequality uses the Chernoff bound. Similar proof concludes the second result which we omit here. Combining the upper bounds for each summation term concludes the proof.

□

Appendix B

Simulations: MNN versus 1-hop witness algorithm

We present another experiment result on the simulated Erdős-Rényi graphs, $corrER(n, p, s)$ with large n. In particular, we compare the performance of running one iteration of the MNN algorithm with the 1-HOP witness algorithm. The principal difference between the two methods is that MNN performs an additional step in finding the most plausible matching candidates in G_1 for each node in G_2. We are interested in the impact of this additional matching step on the matching performance.

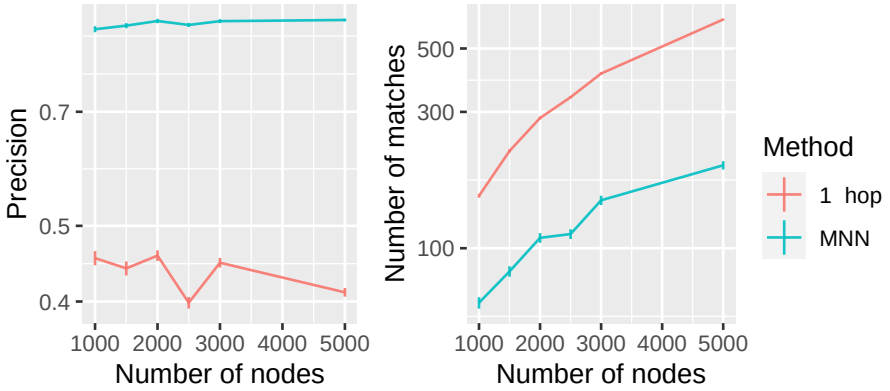


Figure B·1: Visualization of linear transformations applied on the raw data with different methods.

We simulate pairs of Erdős-Rényi graphs with fixed graph correlation $\rho =$

0.8 and edge probability $p = 0.5$. We let the number of seeds α as a function of n, p, and ρ. Specifically, let $\alpha = log(n)/(p \times \rho)$. The number of seeds increases with the size of graphs but at a lower rate. We vary the size of graphs and run 300 Monte Carlo replicates for each of $n \in \{1000, 1500, 2000, 2500, 3000, 5000\}$. The error bars represent 95% confidence intervals.

Figure B·1 displays the matching precision and the number of matches as a function of the graph size. The more stringent matching criteria of the MNN algorithm successfully improve the matching precision dramatically compared with the 1-HOP WITNESS. In addition, As we vary the number of nodes in each graph, the matching precision for MNN is stable and slightly increasing. On the other hand, the 1-HOP WITNESS has a decreasing matching precision as the graph size grows larger. Note that we also experimented with moderate correlation $\rho = 0.5$ and observed similar trends with lower matching precision and smaller matched sets.

References

Aflalo, Y., Bronstein, A., and Kimmel, R. (2015a). On convex relaxation of graph isomorphism. *Proceedings of the National Academy of Sciences*, 112(10):2942 2947.

Aflalo, Y., Bronstein, A., and Kimmel, R. (2015b). On convex relaxation of graph isomorphism. *Proceedings of the National Academy of Sciences*, 112(10):2942 2947.

Bai, Z., Zhang, X.-L., and Chen, J. (2020). Speaker verification by partial auc optimization with mahalanobis distance metric learning. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 28:1533 1548.

Belongie, S., Malik, J., and Puzicha, J. (2002). Shape matching and object recognition using shape contexts. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(4):509 522.

Berg, A. C., Berg, T. L., and Malik, J. (2005). Shape matching and object recognition using low distortion correspondences. *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, 1:26 33 vol. 1.

Binette, O. and Steorts, R. C. (2022). (almost) all of entity resolution. *Science Advances*, 8(12):eabi8021.

Burkard, R., Dell'Amico, M., and Martello, S. (2012). Assignment problems. *Society for Industrial and Applied Mathematics*.

Caelli, T. and Kosinov, S. (2004). An eigenspace projection clustering method for inexact graph matching. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(4):515 519.

Chen, L., Vogelstein, J. T., Lyzinski, V., and Priebe, C. E. (2016). A joint graph inference case study: the c. elegans chemical and electrical connectomes. *Worm*, 5(2):e1142041.

Clauset, A., Newman, M. E. J., and Moore, C. (2004). Finding community structure in very large networks. *Physical Review E*, 70(6).

Conte, D., Foggia, P., Vento, M., and Sansone, C. (2004). Thirty Years Of Graph Matching In Pattern Recognition. *International Journal of Pattern Recognition and Artificial Intelligence*, 18(3):265 298.

Cour, T., Srinivasan, P., and Shi, J. (2006). Balanced graph matching. In Schölkopf, B., Platt, J., and Hoffman, T., editors, *Advances in Neural Information Processing Systems*, volume 19. MIT Press.

Cover, T. and Hart, P. (1967). Nearest neighbor pattern classification. *IEEE Transactions on Information Theory*, 13(1):21 27.

Csardi, G. and Nepusz, T. (2006). The igraph software package for complex network research. *InterJournal*, Complex Systems:1695.

Dempster, A. P., Laird, N. M., and Rubin, D. B. (1977). Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistical*

Society. Series B (Methodological), 39(1):1 38.

Fan, Z., Mao, C., Wu, Y., and Xu, J. (2020). Spectral graph matching and regularized quadratic relaxations: Algorithm and theory. In III, H. D. and Singh, A., editors, *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 2985 2995. PMLR.

Fang, F., Sussman, D. L., and Lyzinski, V. (2018). Tractable graph matching via soft seeding.

Fellegi, I. P. and Sunter, A. B. (1969). A theory for record linkage. *Journal of the American Statistical Association*, 64(328):1183 1210.

Finke, G., Burkard, R. E., and Rendl, F. (1987). Quadratic assignment problems. In Martello, S., Laporte, G., Minoux, M., and Ribeiro, C., editors, *Surveys in Combinatorial Optimization*, volume 132 of *North-Holland Mathematics Studies*, pages 61 82. North-Holland.

Frank, M. and Wolfe, P. (1956). An algorithm for quadratic programming. *Naval Research Logistics Quarterly*, 3(1-2):95 110.

Goldberger, J., Hinton, G. E., Roweis, S., and Salakhutdinov, R. R. (2004). Neighbourhood components analysis. In Saul, L., Weiss, Y., and Bottou, L., editors, *Advances in Neural Information Processing Systems*, volume 17. MIT Press.

Gower, J. (1975). Generalized procrustes analysis. *Psychometrika*, 40(1):33 51.

Holland, P., Laskey, K., and Leinhardt, S. (1983). Stochastic blockmodels: First steps. *Social Networks*, 5:109 137.

Hu, S., Zou, L., Yu, J. X., Wang, H., and Zhao, D. (2018). Answering natural language questions by subgraph matching over knowledge graphs. *IEEE transactions on knowledge and data engineering*, 30(5):824 837.

Huang, Q., Wu, L.-Y., and Zhang, X.-S. (2013). Corbi: a new r package for biological network alignment and querying. *BMC Systems Biology*, 7:S6 S6.

Ito, T., Chiba, T., Ozawa, R., Yoshida, M., Hattori, M., and Sakaki, Y. (2001). A comprehensive two-hybrid analysis to explore the yeast protein interactome. *Proceedings of the National Academy of Sciences*, 98(8):4569 4574.

Jonker, R. and Volgenant, A. (2005). A shortest augmenting path algorithm for dense and sparse linear assignment problems. *Computing*, 38:325 340.

Kazemi, E., Hamed Hassani, S., and Grossglauser, M. (2015). Growing a graph matching from a handful of seeds. *Proceedings of the VLDB Endowment*, 8:1010 1021.

Kelley, B., Yuan, B., Lewitter, F., Sharan, R., Stockwell, B., and Ideker, T. (2004). Pathblast: a tool for alignment of protein interaction networks. *Nucleic acids research*, 32:W83 8.

Kivelä, M., Arenas, A., Barthelemy, M., Gleeson, J. P., Moreno, Y., and Porter, M. A. (2014). Multilayer networks. *Journal of Complex Networks*,

2(3):203 271.

Kuchaiev, O. and Przulj, N. (2011). Integrative network alignment reveals large regions of global network similarity in yeast and human. *Bioinformatics (Oxford, England)*, 27:1390 6.

Kuhn, H. (1955). The hungarian method for the assignment problem. *Naval Research Logistics Quarterly*, 2:83 98.

Leskovec, J., Lang, K. J., Dasgupta, A., and Mahoney, M. W. (2009). Community Structure in Large Networks: Natural Cluster Sizes and the Absence of Large Well-Defined Clusters. *Internet Mathematics*, 6(1):29 123.

Li, L. and Sussman, D. L. (2019). Graph matching via multi-scale heat diffusion. In *2019 IEEE International Conference on Big Data (Big Data)*, pages 1157 1162.

Li, Y., Gu, C., Dullien, T., Vinyals, O., and Kohli, P. (2019). Graph matching networks for learning the similarity of graph structured objects. In Chaudhuri, K. and Salakhutdinov, R., editors, *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*, volume 97 of *Proceedings of Machine Learning Research*, pages 3835 3845. PMLR.

Liao, L., He, X., Zhang, H., and Chua, T.-S. (2018). Attributed social network embedding. *IEEE Transactions on Knowledge and Data Engineering*, 30(12):2257 2270.

Liu, Z.-Y. and Qiao, H. (2012). A convex-concave relaxation procedure based

subgraph matching algorithm. In Hoi, S. C. H. and Buntine, W., editors, *Proceedings of the Asian Conference on Machine Learning*, volume 25 of *Proceedings of Machine Learning Research*, pages 237 252.

Lyzinski, V., Fishkind, D. E., Fiori, M., Vogelstein, J. T., Priebe, C. E., and Sapiro, G. (2016). Graph matching: Relax at your own risk. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 38(1):60 73.

Lyzinski, V., Fishkind, D. E., and Priebe, C. E. (2014). Seeded graph matching for correlated erdös-rényi graphs. *The Journal of Machine Learning Research*, 15(1):3513 3540.

Lyzinski, V. and Sussman, D. L. (2020). Matchability of heterogeneous networks pairs. *Information and Inference: A Journal of the IMA*, 9(4):749 783.

López-Sánchez, D., Arrieta, A. G., and Corchado, J. M. (2019). Visual content-based web page categorization with deep transfer learning and metric learning. *Neurocomputing*, 338:418 431.

Ma, G., Ahmed, N. K., Willke, T. L., and Yu, P. S. (2019). Deep graph similarity learning: A survey. *Data Mining and Knowledge Discovery*, abs/1912.11615.

Macqueen, J. (1967). Some methods for classification and analysis of multivariate observations. In *In 5-th Berkeley Symposium on Mathematical Statistics and Probability*, pages 281 297.

Marco Fortini, Brunero Liseo, A. N. and Scanu, M. (2001). On bayesian record linkage. *Research in official statistics*, 4:185 198.

Mateus, D., Horaud, R. P., Knossow, D., Cuzzolin, F., and Boyer, E. (2008). Articulated shape matching using laplacian eigenfunctions and unsupervised point registration. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*.

Mikolov, T., Chen, K., Corrado, G., and Dean, J. (2013). Efficient estimation of word representations in vector space. In *1st International Conference on Learning Representations, 2013, Scottsdale, Arizona, USA, May 2-4, 2013, Workshop Track Proceedings*.

Moorman, J. D., Chen, Q., Tu, T. K., Boyd, Z. M., and Bertozzi, A. L. (2018a). Filtering methods for subgraph matching on multiplex networks. In *2018 IEEE International Conference on Big Data (Big Data)*, pages 3980 3985. ieeexplore.ieee.org.

Moorman, J. D., Chen, Q., Tu, T. K., Boyd, Z. M., and Bertozzi, A. L. (2018b). Filtering methods for subgraph matching on multiplex networks. In *2018 IEEE International Conference on Big Data (Big Data)*, pages 3980 3985.

Mossel, E. and Xu, J. (2020). Seeded graph matching via large neighborhood statistics. *Random Structures and Algorithms*, 57.

Nabieva, E., Jim, K.-C., Agarwal, A., Chazelle, B., and Singh, M. (2005). Whole-proteome prediction of protein function via graph-theoretic analysis of interaction maps. *Bioinformatics (Oxford, England)*, 21 Suppl 1:i302 10.

Narayanan, A. and Shmatikov, V. (2009). De-anonymizing social networks. In *2009 30th IEEE Symposium on Security and Privacy*, pages 173 187.

Papadimitriou, C. H. and Steiglitz, K. (1998). *Combinatorial Optimization: Algorithms and Complexity*. Courier Corporation.

Riccardo, G. and Marc, B. (2015). The multilayer temporal network of public transport in great britain. *Scientific Data*, 2.

Sariyar, M. and Borg, A. (2010). The RecordLinkage Package: Detecting Errors in Data. *The R Journal*, 2(2):61 67.

Singh, R., Xu, J., and Berger, B. (2008). Global alignment of multiple protein interaction networks with application to functional orthology detection. *Proceedings of the National Academy of Sciences*, 105(35):12763 12768.

Sussman, D. L., Park, Y., Priebe, C. E., and Lyzinski, V. (2020). Matched filters for noisy induced subgraph detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 42(11):2887 2900.

Uetz, P., Giot, L., Cagney, G., Mansfield, T., Judson, R. S., Knight, J. R., Lockshon, D., Narayan, V. A., Srinivasan, M., Pochart, P., Qureshi-Emili, A., Li, Y., Godwin, B. C., Conover, D., Kalbfleisch, T. S., Vijayadamodar, G., Yang, M., Johnston, M., Fields, S., and Rothberg, J. M. (2000). A comprehensive analysis of protein protein interactions in saccharomyces cerevisiae. *Nature*, 403:623 627.

Umeyama, S. (1988). An eigendecomposition approach to weighted graph matching problems. *IEEE Transactions on Pattern Analysis and Machine*

Intelligence, 10(5):695 703.

Volgenant, A. (1996). Linear and semi-assignment problems: A core oriented approach. *Computer and Operations Research*, 23:917 932.

Yartseva, L. and Grossglauser, M. (2013). On the performance of percolation graph matching. *Proceedings of the First ACM Conference on Online Social Networks*, pages 119 130.

Young, S. J. and Scheinerman, E. R. (2007). Random dot product graph models for social networks. In *Proceedings of the 5th International Conference on Algorithms and Models for the Web-graph*, pages 138 149, Berlin, Heidelberg. Springer-Verlag.

Zaslavskiy, M., Bach, F., and Vert, J. (2009). A path following algorithm for the graph matching problem. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 31(12):2227 2242.

CURRICULUM VITAE