Joint Graph Embedding and Alignment with Spectral Pivot

Paris A. Karakasis*
University of Virginia
Charlottesville, Virginia, USA
karakasis@virginia.edu

Aritra Konar*
University of Virginia
Charlottesville, Virginia, USA
aritra@virginia.edu

Nicholas D. Sidiropoulos
University of Virginia
Charlottesville, Virginia, USA
nikos@virginia.edu

ABSTRACT

Graphs are powerful abstractions that naturally capture the wealth of relationships in our interconnected world. This paper proposes a new approach for graph alignment, a core problem in graph mining. Classical (e.g., spectral) methods use fixed embeddings for both graphs to perform the alignment. In contrast, the proposed approach fixes the embedding of the 'target' graph and jointly optimizes the embedding transformation and the alignment of the 'query' graph. An alternating optimization algorithm is proposed for computing high-quality approximate solutions and compared against the prevailing state-of-the-art graph aligning frameworks using benchmark real-world graphs. The results indicate that the proposed formulation can offer significant gains in terms of matching accuracy and robustness to noise relative to existing solutions for this hard but important problem.

CCS CONCEPTS

• **Mathematics of computing** → **Graph algorithms**; **Approximation algorithms**; • **Information systems** → *Data mining*.

KEYWORDS

graph alignment; graph embedding; spectral methods; alternating optimization

ACM Reference Format:
Paris A. Karakasis, Aritra Konar, and Nicholas D. Sidiropoulos. 2021. Joint Graph Embedding and Alignment with Spectral Pivot. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD '21), August 14–18, 2021, Virtual Event, Singapore.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3447548.3467377

1 INTRODUCTION

Graph-based representation models have become powerful for encoding relational structures that are encountered in our interconnected world, such as co-authorship networks, protein-protein interaction networks, and social networks. In this paper we study the fundamental problem of graph alignment, which aims to find an alignment of the vertices of two graphs such that the number of induced edge disagreements is minimized. This problem has applications in a wide variety of disciplines, including spanning computer vision, image and social network analysis, pattern recognition, bioinformatics and neuroscience (see [11] and references therein). On the other hand, graph alignment is a demanding combinatorial problem. As a result, the need for efficient and scalable algorithms capable of providing good approximate solutions is imperative.

The graph alignment problem can be divided into different subcategories. Given a pair of graphs with the same number of nodes, the case of exact graph alignment is known as the graph isomorphism problem. Correspondingly, the problem of exact alignment of a graph to a part of another graph is called the subgraph isomorphism problem. Inexact graph alignment refers to problems where exact matching is impossible. In its most general form, the graph alignment problem is equivalent to a quadratic assignment problem (QAP) [43], which is known to be NP-hard [36]. In that regime, there is a pressing need for algorithms which are both theoretically sound and highly scalable.

Towards this end, various concepts and tools used for mining graph information are being increasingly employed for graph alignment as well. One such concept that has recently received significant attention is that of learning efficient representations of graphs, known as graph embeddings. Graph embedding approaches can be usually viewed as finding mappings that embed nodes, subgraphs, or even whole graphs in some cases, as points in a low-dimensional vector space [5, 22]. The main goal of these approaches is to capture valuable graph structural information and properties in geometric relationships in the embedding space, where downstream tasks can be performed naturally and at lower complexity. Finding efficient graph representations is challenging and determining which type of embedding method is most suitable is application-specific [5].

In the context of graph alignment, the most common category of embedding methods is that of *node embeddings*. For a given n-th order graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, the representation learning problem can be stated as finding a function $f : \mathcal{V} \to \mathbb{R}^d$, usually for $d \ll n$, that encodes structural information and properties of each node. Regarding the usefulness of such mappings in aligning graphs, one may argue that an efficient embedding method should represent nodes that have similar structural properties by similar embedding vectors. If this holds, then two nodes, that belong to different graphs and have similar embedding vectors, would be candidates for a possible match. Hence, the graph alignment problem can be expressed in terms of the node embeddings of the given pair of graphs, and as a result, may potentially lead to more tractable formulations of the graph alignment problem.

The majority of node embedding algorithms relies on what we call direct encoding methods [22] and can be roughly divided into two subcategories: i) the matrix factorization-based approaches, like the Graph Factorization algorithm [1], GraRep [6], and HOPE

*Both authors contributed equally to this research.

KDD '21, August 14–18, 2021, Virtual Event, Singapore

ACM ISBN 978-1-4503-8332-5/21/08…$15.00
https://doi.org/10.1145/3447548.3467377

[35]; and ii) the random walk-based approaches, like DeepWalk [37], node2vec [21], and HARP [8]. Surprisingly, the authors of [38] showed that some of the random walk-based models can be accommodated under the matrix factorization framework with closed form expressions, when viewed under a negative sampling 'lens'. Based on that observation, the same authors proposed the NetMF method for computing node embeddings [38]. Under the subcategory of matrix factorization-based methods, we can also place the rich family of the so called spectral methods. The spectral decomposition of a matrix representation of a graph, such as its adjacency or Laplacian matrix, has been also proven to be a solid and efficient basis for creating graph representations for numerous graph mining problems, including graph alignment.

Contributions: In this paper, we introduce a new framework for graph alignment that "pivots" around the (fixed) spectral embedding of the 'target' graph to jointly compute the optimal node embeddings of the 'query' graph and the one-to-one correspondence mapping that aligns the two graphs. We cast our formulation as a non-convex optimization problem and propose an alternating optimization algorithm. Finally, we test our method on real life graphs, and compare it with other state-of-the-art baselines. Our results reveal that the proposed algorithm can obtain significant improvement in performance (with regards to correctly aligning edges) relative to the prior art at affordable complexity.

2 RELATED WORK

Previous approaches [10, 12, 30] have formulated graph alignment as a Quadratic Programming (QP) problem. Because of its combinatorial nature, graph alignment is either solved exactly in very restricted settings (e.g., on trees [26], planar graphs [24], and graphs with bounded eigenvalue multiplicity [3]) or else approximately, using various optimization approaches (see [25, 27, 45, 48]).

A different approach to the graph alignment problem employs spectral methods [43], which exploit information conveyed by the eigenvectors of the adjacency matrix. Since [43], numerous spectral approaches have been proposed. Prominent among these is IsoRank [42], which uses random walks with restarts to align nodes across two graphs based on their topological similarities. More recent methods include EigenAlign (EA) and LowRankAlign (LRA)[18]. EA computes the leading eigenvector of a function of the adjacency matrices of the two graphs and then performs a maximum weight bipartite matching optimization to construct an alignment. LRA solves the QAP problem over the relaxed feasible set of orthogonal matrices and then produces an estimate of the permutation matrix by searching in a reduced space, which is determined by the solution of the first step.

Going beyond spectral embeddings, employing node embeddings has proven to be very fruitful for graph matching, although much remains to be done in this direction. One example of embedding based methods is REGAL [23], which computes degree-based features from a node's neighborhood at different hop-lengths, and then uses an implicit factorization of the resulting feature matrix to form embeddings that are suitable for network alignment. Recently, the same authors proposed CONE-Align[9], which uses the NetMF node embeddings [38] of the two graphs and adopts the framework presented in [20] to produce a matching based on aligning the representations of the embedding subspaces. In both methods, the final matching is performed by using kd-trees for fast nearest node embedding search. The problem of jointly learning embedding vectors and matching the nodes of two graphs has also attracted interest, including [44], where a Gromov-Wasserstein discrepancy based learning framework was proposed. The main motivation behind joint embedding and matching is that if we know that the two graphs are quasi-isomorphic, this can help learn better node embeddings *from and for both* graphs. Conversely, such embeddings will naturally be better-suited to aid the alignment task.

3 PROBLEM FORMULATION

Given a pair of (possibly directed) unweighted graphs of the same order, graph alignment is the problem of finding a bijection between the vertex sets that minimizes the number of edge disagreements. In other words, let $\mathcal{G}_1$ and $\mathcal{G}_2$ be a pair of graphs on n vertices with respective adjacency matrices $\mathbf{A}_1 \in \{0,1\}^{n\times n}$ and $\mathbf{A}_2 \in \{0,1\}^{n\times n}$. Then, the problem can be formulated as the optimization problem

$$\min_{\mathbf{P}\in\mathcal{P}^n} \left\|\mathbf{A}_1 - \mathbf{P}\mathbf{A}_2\mathbf{P}^T\right\|_F^2, \tag{1}$$

where $\|\cdot\|_F$ denotes the Frobenius norm of a matrix, while the feasible set $\mathcal{P}^n$ denotes the set of all $n \times n$ permutation matrices, which can be compactly expressed as

$$\mathcal{P}^n := \left\{\mathbf{P} \in \mathbb{R}_+^{n\times n} \mid \mathbf{P}^T\mathbf{P} = \mathbf{P}\mathbf{P}^T = \mathbf{I}_n\right\}. \tag{2}$$

Note that the cost function of (1) can be expressed as

$$\left\|\mathbf{A}_1 - \mathbf{P}\mathbf{A}_2\mathbf{P}^T\right\|_F^2 = \|\mathbf{A}_1\|_F^2 + \|\mathbf{A}_2\|_F^2 - 2\,\mathrm{trace}\left(\mathbf{A}_1^T\mathbf{P}\mathbf{A}_2\mathbf{P}^T\right).$$

Hence, problem (1) is equivalent to

$$\max_{\mathbf{P}\in\mathcal{P}^n} \mathrm{trace}\left(\mathbf{A}_1^T\mathbf{P}\mathbf{A}_2\mathbf{P}^T\right), \tag{3}$$

which, in its general form, corresponds to a NP-hard problem; the Quadratic Assignment Problem (QAP). Moreover, computing an approximate solution within a factor better than $2^{\log n^{1-\epsilon}}$ of the optimum is, in general, not feasible in polynomial time either [39]. However, several algorithms have been designed to provide approximate solutions. We refer the reader to [18] for a brief review of these methods and to [4, 17, 33] for a more extensive review.

An alternative approach to graph alignment, that has constituted the basis of many recently proposed works, aims to perform alignment based on learned representations of the nodes of the two graphs. The advantage of using node embedding methods for graph alignment is that, in contrast to the NP–hard problem (3), if $\mathbf{E}_1, \mathbf{E}_2 \in \mathbb{R}^{n\times d}$ denote d-dimensional node embeddings (with $d << n$) of $\mathcal{G}_1$ and $\mathcal{G}_2$ respectively, then the alignment problem can be casted as the following Linear Assignment Problem (LAP)

$$\min_{\mathbf{P}\in\mathcal{P}^n} \|\mathbf{E}_1 - \mathbf{P}\mathbf{E}_2\|_F^2 \Leftrightarrow \max_{\mathbf{P}\in\mathcal{P}^n} \mathrm{trace}\left(\mathbf{P}\mathbf{E}_2\mathbf{E}_1^T\right), \tag{4}$$

which can be solved optimally using the Hungarian algorithm [36]. However, for large number of nodes, the Hungarian algorithm is impractical, since it has a complexity of $O\left(n^3\right)$. Hence, devising scalable algorithms that can provide approximate solution has attracted the interest of many researchers. One popular choice that has been adopted lately by many [9, 20, 46] is using an approximate Earth Mover Distance solver based on the Sinkhorn algorithm

[13], the complexity of which is $O\left(n^2\right)$ (up to logarithmic terms [2, 13]). Therefore, solving the graph alignment problem by utilizing discriminative node embeddings has great potential.

4 PROPOSED APPROACH

Most of the state-of-the-art graph embedding based approaches aim to solve the graph alignment problem after fixing the embeddings, which are learned separately for each graph. While using fixed node embeddings to cast graph alignment as a LAP is a promising approach, the main drawback of such a scheme is that the quality of the obtained solution is dependant, to a large degree, on the adequacy of the selected embeddings for the graph alignment task. In contrast, we propose to learn the embeddings of the target graph after fixing those of the query, while simultaneously searching for the permutation that best aligns the embeddings. This offers additional degrees of freedom compared to the fixed node embeddings based formulations, and greater potential for devising high performance graph alignment methods.

More specifically, in our present work, we aim to combine the merits of (learned) node embedding and spectral based methods for the task of performing graph alignment by considering the following joint problem: first, we compute the spectral embedding of the target graph and fix it. Next, we "pivot" around this fixed node representation to simultaneously search for the optimal node embeddings of the second graph and the optimal one-to-one correspondence mapping. Specifically, given two adjacency matrices $\mathbf{A}_1$ and $\mathbf{A}_2$, we consider low dimensional linear embeddings of the form

$$\mathbf{E}_i = \mathbf{A}_i\mathbf{Q}_i, \quad \text{for } i = 1, 2, \tag{5}$$

where $\mathbf{Q}_i \in \mathbb{R}^{n\times d}$ denotes the dimensionality reduction matrix of graph i. Then, after fixing the node embeddings of the first graph to be the top-d principal components of $\mathbf{A}_1$, we calculate a node assignment between the two graphs and the linear node embeddings of the second graph by solving an optimization problem of the form

$$\begin{aligned} \min_{\mathbf{P},\,\mathbf{Q}_2} \ & \|\mathbf{E}_1 - \mathbf{P}\mathbf{A}_2\mathbf{Q}_2\|_F^2, \\ \text{s.t.} \ & \mathbf{P} \in \mathcal{P}^n. \end{aligned} \tag{6}$$

Although the above problem is not convex, conditionally optimal solutions can be obtained for the two optimization sub-problems that emerge after fixing one of the optimization variables. This observation suggests a natural alternating optimization algorithm for (6). However, as it was noted for a closely related optimization problem in [20, 46], and we also verified in our context, plain vanilla alternating minimization tends to quickly converge to bad local minima. In order to evade the phenomenon described above, we propose a variation of problem (6), based on two observations, and an algorithmic approach.

First, from (2), we note that the set of n-dimensional permutation matrices can be expressed as the set of non-negative matrices that are also orthonormal. Although seemingly redundant, another equivalent and useful description of this set can be obtained by considering it as the intersection of the set of left stochastic matrices (non-negative matrices with each column summing to 1) and the set of orthonormal matrices. Moreover, in the case of isomorphic graphs, the optimal matrices $\{\mathbf{Q}_i\}_{i=1}^2$ should exhibit the same alignment structure as the optimal embeddings. In other words, in the case where the adjacency matrices $\mathbf{A}_1$ and $\mathbf{A}_2$ are permuted versions of each other obeying the relation $\mathbf{A}_2 = \mathbf{P}\mathbf{A}_1\mathbf{P}^T$, the optimal matrices $\mathbf{Q}_1^*$ and $\mathbf{Q}_2^*$ should satisfy

$$\mathbf{Q}_1^* = \mathbf{P}\mathbf{Q}_2^*. \tag{7}$$

In applications, the target and query graphs are seldom isomorphic. The further the two graphs are from being isomorphic, the less relation (7) will hold.

Taken together, these considerations suggest the following variation of (6)

$$\begin{aligned} \min_{\mathbf{P},\,\mathbf{Q}_2} \ & \|\mathbf{E}_1 - \mathbf{P}\mathbf{A}_2\mathbf{Q}_2\|_F^2 + \lambda\,\|\mathbf{Q}_1 - \mathbf{P}\mathbf{Q}_2\|_F^2, \\ \text{s.t.} \ & \mathbf{P} \geq \mathbf{0}, \quad \mathbf{1}_n^T\mathbf{P} = \mathbf{1}_n^T, \quad \mathbf{P}^T\mathbf{P} = \mathbf{P}\mathbf{P}^T = \mathbf{I}_n, \end{aligned} \tag{8}$$

where $\mathbf{1}_n$ denotes the n-dimensional vector of ones and $\lambda > 0$ is a penalty parameter that regulates the emphasis placed on the second term and should be tuned accordingly to how far are the considered graphs from being isomorphic. Meanwhile, the constraints $\mathbf{P} \geq \mathbf{0}$ (element-wise non-negativity) and $\mathbf{1}_n^T\mathbf{P} = \mathbf{1}_n^T$ impose that matrix $\mathbf{P}$ is left stochastic. In the next section, we devise an algorithmic approach for solving the above problem.

5 ALGORITHMIC APPROACH & EVALUATION

Note that problem (8) is non-convex, as the cost function is the composition of a bilinear function of the variables with a convex function, and the permutation constraints are combinatorial. In order to effectively deal with the permutation constraints, we consider the following equivalent form of (8)

$$\begin{aligned} \min_{\mathbf{U},\,\mathbf{W},\,\mathbf{Q}_2} \ & \|\mathbf{E}_1 - \mathbf{U}\mathbf{A}_2\mathbf{Q}_2\|_F^2 + \lambda\,\|\mathbf{Q}_1 - \mathbf{U}\mathbf{Q}_2\|_F^2 \\ \text{s.t.} \ & \mathbf{W} \geq \mathbf{0}, \quad \mathbf{1}_n^T\mathbf{W} = \mathbf{1}_n^T, \\ & \mathbf{U}^T\mathbf{U} = \mathbf{U}\mathbf{U}^T = \mathbf{I}_n, \quad \mathbf{U} = \mathbf{W}, \end{aligned} \tag{9}$$

where we have replaced $\mathbf{P}$ with the new variables $\{\mathbf{U}, \mathbf{W}\}$ which collectively impose the permutation constraint. Instead of enforcing the equality constraint between $\mathbf{U}$ and $\mathbf{W}$ exactly, we consider the following "penalty" formulation

$$\begin{aligned} \min_{\mathbf{U},\,\mathbf{W},\,\mathbf{Q}_2} \ & \|\mathbf{E}_1 - \mathbf{U}\mathbf{A}_2\mathbf{Q}_2\|_F^2 + \lambda\,\|\mathbf{Q}_1 - \mathbf{U}\mathbf{Q}_2\|_F^2 + \rho\|\mathbf{U} - \mathbf{W}\|_F^2 \\ \text{s.t.} \ & \mathbf{W} \geq \mathbf{0}, \quad \mathbf{1}_n^T\mathbf{W} = \mathbf{1}_n^T, \quad \mathbf{U}^T\mathbf{U} = \mathbf{U}\mathbf{U}^T = \mathbf{I}_n, \end{aligned} \tag{10}$$

where we have augmented the cost function of the previous problem with a proximal regularization term that penalizes the violation of the equality constraint between the variables $\mathbf{U}$ and $\mathbf{W}$, with the level of tolerable violation regulated by the non-negative penalty parameter ρ.

Note that problem (10) is block-separable in the variables $\mathbf{U}$ and $\{\mathbf{W}, \mathbf{Q}_2\}$. We exploit this structure to devise an alternating optimization algorithm for obtaining approximate solutions of this problem. At each iteration of the algorithm, we compute the conditionally optimal solution w.r.t. one of the variable blocks, while the other is kept fixed. Specifically, given the current iterates $\mathbf{U}^k, \mathbf{W}^k, \mathbf{Q}_2^k$, at the k-th iteration, the proposed algorithm entails solving the following sub-problems in cyclic fashion.

$$\mathbf{Q}_2^{k+1} = \operatorname*{argmin}_{\mathbf{Q}_2} \left\|\mathbf{E}_1 - \mathbf{U}^k\mathbf{A}_2\mathbf{Q}_2\right\|_F^2 + \lambda\left\|\mathbf{Q}_1 - \mathbf{U}^k\mathbf{Q}_2\right\|_F^2, \tag{11}$$

$$\mathbf{W}^{k+1} = \underset{\mathbf{W}}{\operatorname{argmin}} \left\| \mathbf{W} - \mathbf{U}^k \right\|_F^2, \text{ s.t. } \mathbf{W} \geq \mathbf{0}, \quad \mathbf{1}_n^T \mathbf{W} = \mathbf{1}_n^T, \tag{12}$$

$$\mathbf{U}^{k+1} \in \underset{\mathbf{U}}{\operatorname{argmin}} \left\| \mathbf{E}_1 - \mathbf{U}\mathbf{A}_2\mathbf{Q}_2^{k+1} \right\|_F^2 + \lambda \left\| \mathbf{Q}_1 - \mathbf{U}\mathbf{Q}_2^{k+1} \right\|_F^2 + \rho \left\| \mathbf{U} - \mathbf{W}^{k+1} \right\|_F^2, \quad \text{s.t. } \mathbf{U}^T\mathbf{U} = \mathbf{U}\mathbf{U}^T = \mathbf{I}_n. \tag{13}$$

Note that for a given $\mathbf{U}^k$, the updates of $\mathbf{W}^{k+1}$ and $\mathbf{Q}_2^{k+1}$ can be carried out simultaneously. Since we compute the conditionally optimal updates at each step, it follows that the algorithm monotonically reduces the cost function of problem (10) over iterations. Coupled together with the fact that the cost function is bounded from below (a trivial lower bound being 0), we conclude that the iterates generated by the algorithm attain convergence in terms of the cost function. Furthermore, we have the following result.

Proposition 5.1. *Every limit point of the iterates generated by Algorithm* 1 *is a stationary point of problem* (10).

Proof. We invoke the main result of [7], which asserts that every limit point of the iterates generated by two-block alternating optimization is a stationary point provided that: (a) each block update is conditionally optimal, and (b) the constraint set in each block sub-problem is compact. It is easy to see that the first condition is satisfied by Algorithm 1. Regarding the second condition, note that constraint set of sub-problem (12) is compact, as each column lies in the probability simplex, whereas the set of orthonormal matrices (which constitutes the constraint set of sub-problem (13)) is also compact [41, p. 4]. This leaves sub-problem (11), which being unconstrained, at first glance, does not appear to have a compact constraint set. However, since we apply a proximal term in the cost function that restricts $\mathbf{Q}_2$ to be close to $(\mathbf{U}^k)^T\mathbf{Q}_1$, this *implicitly* enforces $\mathbf{Q}_2$ to lie in a compact set. In other words, for every choice of regularization parameter λ, there exists a bounded ℓ_2-ball (with radius dependent on λ) inside which the solution of (11) lies. Having guaranteed that the requisite conditions for convergence are obeyed by the iterates generated by Algorithm 1, the result then follows. □

However, attaining convergence does not guarantee that equality between the iterates $\mathbf{U}$ and $\mathbf{W}$ is achieved in general, which in turn implies that the algorithm may not return a valid permutation matrix. Consequently, as a post-processing step, we employ a greedy maximum-weight bipartite matching algorithm [28] on the last $\mathbf{U}$ iterate to obtain the final alignment. The overall algorithm is outlined in Algorithms (1) and (2).

Next, we consider the computational complexity of solving each of the above sub-problems and demonstrate that they admit closed-form solutions.

5.1 Updating matrix $\mathbf{Q}_2$

The sub-problem (11) that corresponds to the update of $\mathbf{Q}_2^k$ is an unconstrained least squares problem, as it is equivalent to

$$\mathbf{Q}_2^{k+1} = \underset{\mathbf{Q}_2}{\operatorname{argmin}} \left\| \begin{bmatrix} \mathbf{E}_1 \\ \sqrt{\lambda}\mathbf{Q}_1 \end{bmatrix} - \begin{bmatrix} \mathbf{U}^k\mathbf{A}_2 \\ \sqrt{\lambda}\mathbf{U}^k \end{bmatrix} \mathbf{Q}_2 \right\|_F^2. \tag{14}$$

Algorithm 1 Proposed Algorithm for Graph Alignment

Input: $\mathbf{A}_1 \in \{0,1\}^{n\times n}$, $\mathbf{A}_2 \in \{0,1\}^{n\times n}$, $\mathbf{U}^0 \in \mathbb{R}^{n\times n}$, $d \in [n]$, $\lambda \in \mathbb{R}_+$, $\rho \in \mathbb{R}_+$
Output: $\mathbf{P} \in \mathcal{P}^n$
function Graph_Alignment($\mathbf{A}_1$, $\mathbf{A}_2$, $\mathbf{U}^0$, d, λ, ρ)
  $[\mathbf{U}_1, \Sigma_1, \mathbf{V}_1] = \text{svd}(\mathbf{A}_1, \text{'econ'})$
  $\mathbf{E}_1 = \mathbf{U}_1(:, 1:d)$
  $\mathbf{Q}_1 = \mathbf{V}_1(:, 1:d)\,[\Sigma_1(1:d, 1:d)]^{-1}$
  $\mathbf{C}_1 = \left[\mathbf{E}_1,\ \sqrt{\lambda}\mathbf{Q}_1\right]$
  $\mathbf{L} = \text{chol}\left(\mathbf{A}_2^T\mathbf{A}_2 + \lambda\mathbf{I}_n\right)$
  $k = 0$
  while terminating_condition is FALSE **do**
    $\mathbf{Q}_2^{k+1} = \mathbf{L}^{-1}\left(\mathbf{L}^T\right)^{-1}\left[\mathbf{A}_2^T\left(\mathbf{U}^k\right)^T\mathbf{E}_1 + \lambda\left(\mathbf{U}^k\right)^T\mathbf{Q}_1\right]$
    $\mathbf{C}_2 = \left[\mathbf{A}_2\mathbf{Q}_2^{k+1},\ \sqrt{\lambda}\mathbf{Q}_2^{k+1}\right]$
    $\mathbf{W}^{k+1} = \text{ColumnwiseSimplexProjection}\left(\mathbf{U}^k\right)$
    $\mathbf{U}^{k+1} = \text{Procustes}\left(\mathbf{C}_1, \mathbf{C}_2, \mathbf{W}^{k+1}, \rho\right)$
    $k = k + 1$
  end while
  $\mathbf{P} = \text{GreedyMatching}\left(\mathbf{U}^k\right)$
end function

Algorithm 2 Orthogonal Procrustes

Input: $\mathbf{C}_1 \in \mathbb{R}^{n\times 2d}$, $\mathbf{C}_2 \in \mathbb{R}^{n\times 2d}$, $\mathbf{W} \in \mathbb{R}^{n\times n}$, $\rho \in \mathbb{R}_+$
Output: $\mathbf{U}^* \in \left\{\mathbf{U} \in \mathbb{R}^{n\times n} \middle|\ \mathbf{U}^T\mathbf{U} = \mathbf{U}\mathbf{U}^T = \mathbf{I}_n\right\}$
function Procustes($\mathbf{C}_1$, $\mathbf{C}_2$, $\mathbf{W}$, ρ)
  $\mathbf{M} = \mathbf{C}_1\mathbf{C}_2^T + \rho\mathbf{W}$
  $[\mathbf{X}, \Sigma, \mathbf{Y}] = \text{svd}(\mathbf{M})$
  $\mathbf{U}^* = \mathbf{X}\mathbf{Y}^T$
end function

Algorithm 3 Columnwise Projection onto the probability simplex

Input: $\mathbf{W} \in \mathbb{R}^{n\times n}$
Output: $\mathbf{W}^* \in \left\{\mathbf{W} \in \mathbb{R}^{n\times n} \middle|\ \mathbf{W} \geq \mathbf{0}, \mathbf{1}^T\mathbf{W} = \mathbf{1}^T\right\}$
function ColumnwiseSimplexProjection($\mathbf{W}$)
  for i = 1,..., n **do**
    Sort $\mathbf{W}(:, i)$ into $\boldsymbol{\mu}^i$: $\mu_1^i \geq \mu_2^i \geq \ldots \geq \mu_n^i$
    Find $\rho^i = \max\left\{j \in [n] : \mu_j^i - \frac{1}{j}\left(\sum_{r=1}^j \mu_r^i - 1\right) > 0\right\}$
    $\theta^i = \frac{1}{\rho^i}\left(\sum_{r=1}^{\rho^i} \mu_r^i - 1\right)$
    $\mathbf{W}^*(:, i) = \max\left\{\mathbf{W}(:, i) - \theta^i\mathbf{1}_n, \mathbf{0}_{n\times 1}\right\}$
  end for
end function

Hence, the optimal solution must obey the normal equations, which, for problem (14), are given by

$$\left(\mathbf{A}_2^T(\mathbf{U}^k)^T\mathbf{U}^k\mathbf{A}_2 + \lambda(\mathbf{U}^k)^T\mathbf{U}^k\right)\mathbf{Q}_2^{k+1} = \mathbf{A}_2^T(\mathbf{U}^k)^T\mathbf{E}_1 + \lambda(\mathbf{U}^k)^T\mathbf{Q}_1.$$

As we enforce orthonormality constraints on $\mathbf{U}$ at every iteration, the above condition simplifies to

$$\left(\mathbf{A}_2^T\mathbf{A}_2 + \lambda\mathbf{I}_n\right)\mathbf{Q}_2^{k+1} = \mathbf{A}_2^T(\mathbf{U}^k)^T\mathbf{E}_1 + \lambda(\mathbf{U}^k)^T\mathbf{Q}_1, \tag{15}$$

which necessitates solving a total of d linear systems of equations. It is evident that the main computational overhead stems from computing the Cholesky factorization of the positive-definite matrix $\mathbf{A}_2^T\mathbf{A}_2 + \lambda\mathbf{I}_n$, which requires $n^3/3$ flops in the worst-case. Since it is only the right-hand sides of the linear equations which are updated over iterations, it suffices to perform this decomposition only *once* prior to the start of the algorithm, as by caching the factorization we can re-utilize it to solve the linear systems arising over iterations. Hence, the complexity that is initially incurred in computing the Cholesky decomposition is amortized over the iterations of the algorithm. Within each alternating optimization step, solving the d linear systems via forward-backward substitution incurs complexity $O(dn^2)$.

5.2 Updating matrix W

The update of variable $\mathbf{W}^{k+1}$ is given by problem (12), which corresponds to computing the Euclidean projection of the matrix $\mathbf{U}^k$ onto the set of left stochastic matrices. The optimal solution can be obtained by *separately* projecting each column of $\mathbf{U}^k$ onto the probability simplex. The projection of each column can be computed with complexity $O(n\log n)$ [15]. Therefore, the overall complexity of updating matrix $\mathbf{W}$ is $O(n^2\log n)$. In Algorithm 1, the function the computes the updated version of matrix $\mathbf{W}$ is denoted as ColumnwiseSimplexProjection and it is presented in Algorithm 3.

5.3 Updating matrix U

In order to derive a closed-from expression for the update of $\mathbf{U}^{k+1}$, we first expand the cost function of (13) and exploit the orthonormal property of $\mathbf{U}$ to arrive at the problem

$$\mathbf{U}^{k+1} \in \underset{\mathbf{U}^T\mathbf{U}=\mathbf{U}\mathbf{U}^T=\mathbf{I}_n}{\operatorname{argmax}} \operatorname{trace}\left(\mathbf{U}^T\mathbf{M}^k\right), \tag{16}$$

where we have defined the matrix

$$\mathbf{M}^k := \mathbf{E}_1(\mathbf{A}_2\mathbf{Q}_2^{k+1})^T + \lambda\mathbf{Q}_1(\mathbf{Q}_2^{k+1})^T + \rho\mathbf{W}^{k+1}. \tag{17}$$

Note that (16) is an instance of the well-known Orthogonal Procrustes problem [16, 19, 40]. Denoting the singular value decomposition of $\mathbf{M}^k$ by $\mathbf{X}\Sigma\mathbf{Y}^T$, it can be shown that the optimal solution of (16) is given by

$$\mathbf{U}^{k+1} = \mathbf{X}\mathbf{Y}^T, \tag{18}$$

which incurs complexity $O(n^3)$.

Overall complexity of the proposed algorithm. Let T denote the number of iterations of the alternating minimization procedure. Then, the complexity of the algorithm scales as

$$n^3/3 + O(n^2\log n) + O(Tn^3) + O(Tdn^2).$$

The main computational bottleneck in our algorithm is the calculation of the singular value decomposition for solving the Orthogonal Procrustes problem (update of matrix $\mathbf{U}$). Since we run our algorithm for few tens of iterations in our experiments across all datasets, we can treat T as a constant, which results in an overall complexity of $O(n^3)$.

6 EXPERIMENTAL EVALUATION

In this section, we compare the performance of our proposed technique against well known existing network alignment methods on real world graphs. The comparison is carried out in terms of accuracy and of required wall time.

6.1 Baselines

Next, we briefly introduce the algorithms that we employed as performance benchmarks in our experiments:

- **Umeyama's Method** [43]: A classical spectral-embedding based alignment method, which entails computing the complete set of eigenvectors of both adjacency matrices $\mathbf{A}_1$ and $\mathbf{A}_2$ in order to construct a similarity matrix between the vertex sets of the two graphs. Then, the matrix is used to instantiate a LAP, the solution of which corresponds to the final estimate of the alignment.
- **IsoRank**[1] [42]: A spectral method which considers a regularized form of the QAP (3) and applies random walks with restarts to compute the PageRank eigenvector of the normalized Kronecker product graph $\mathbf{A}_1 \otimes \mathbf{A}_2$ (with a predetermined alignment prior serving as the "teleportation" vector). The solution is projected onto the set of permutation constraints via a bipartite matching step to obtain the final alignment. We point out that for the case of undirected, unattributed graphs (which constitutes the focus of our experiments in this paper), IsoRank is a special case of the more general FINAL algorithm [47].
- **Low-Rank Align (LRA)**[2] [18]: A recent two-step spectral alignment method for approximately solving the QAP formulation of graph alignment (3) on undirected graphs. In the first step, LRA solves a rank-k approximation of the orthogonal relaxation of the underlying QAP. At the second step, a variation of the QAP problem with two optimization variables is proposed, and an alternating optimization framework is adopted. The update of one variable entails solving a discrete optimization problem, which is heuristically solved by searching in a reduced space (determined by the solution of the first step), while the update of the second variable boils down to solving a LAP. In our experiments, we use the default parameter setting, while the number of considered eigenvectors, k, was equal to 2.
- **CONE-Align**[3] [9]: A recently proposed joint embedding and alignment meta-algorithm comprising three stages. In the first stage, the NetMF method [38] is used to obtain node embeddings of the two graphs. Then, [9] aims to rule out any rotational ambiguities between the node embeddings by adopting the Wasserstein-Procrustes framework presented in [20]. In the last step, kd-trees for fast nearest node embedding search are employed with the aim of obtaining the final alignment. In our experiments, we use the default settings

[1] https://github.com/sizhang92/FINAL-network-alignment-KDD16
[2] https://github.com/SoheilFeizi/spectral-graph-alignment
[3] https://github.com/GemsLab/CONE-Align

Table 1: Summary of network statistics: the number of vertices (n), the number of edges (m), and the network type.

Graph	n	m	Network Type
C. Elegans	277	2,105	Interactome
arenas-Email	1,133	5,451	Communications
polBlog	1,224	16,714	Social
Airports	1,574	17,215	Infrastructure
A. Thaliana	2,082	4,145	Interactome
Japanese Book	3,177	7,998	Word Adjacency
HomoSapiens	3,890	38,292	Interactome
ca-GrQc	5,242	14,490	Co-authorship

for all the parameters except for the embedding dimension d where we also use the values presented in Table 2. We observed that the output of the final kd-tree stage does not find a one-to-one correspondence mapping in all cases. In these cases, we applied a greedy bipartite matching algorithm [28] on the kd-tree output to obtain a bijective mapping .

6.2 Datasets

The real-world datasets we used in our experiments were obtained from standard network repositories [29, 31] and are listed in Table 1. These include (i) interactomes (C. Elegans, A. Thaliana and HomoSapiens), where the vertices are proteins and the edges correspond to their interactions, (ii) a communications network representing email exchanges among members of a university (arenas-Email), (iii) a social network (polBlog) where the vertices represent blogs and the edges denote followings between blogs, (iv) an infrastructure network comprising the connections between airports in the US (Airports), (v) a word adjacency network constructed from a Japanese text (Japanese Book), and (vi) a co-authorship network representing collaborations between scientists involved in co-authoring a scientific publication (ca-GrQc).

6.3 Experimental Setup

In our experiments, we adopt the experimental setup that, among others, the authors of [14, 18, 27, 34] also consider. Given a dataset, we restrict ourselves to its largest (strongly) connected component. If the original graph is directed, we convert it into undirected form by performing a symmetrization step. Additionally, we remove all self-loops and edge weights. Hence, every graph that we consider in our experiments is a simple, undirected, unweighted graph ($\mathcal{G}_1$).

In order to create an instance of the graph alignment problem, for each graph ($\mathcal{G}_1$), we construct a "noisy" and permuted version of that graph, by randomly adding new edges with probability p_e, i.e., we generate a random Erdos-Renyi graph $\mathcal{G}(n, p_e)$, with adjacency matrix $\mathbf{Q}$. Then, we create the adjacency matrix of $\mathcal{G}_2$ as

$$\mathbf{A}_2 = \mathbf{P}\left[\mathbf{A}_1 + (\mathbf{1}_{n\times n} - \mathbf{A}_1) * \mathbf{Q}\right]\mathbf{P}^T, \tag{19}$$

where the operator "$*$" denotes the Hadamard (element-wise) product and $\mathbf{P} \in \mathcal{P}^n$ denotes a randomly generated permutation matrix. The number of additional edges that eventually appear in $\mathbf{A}_2$ is controlled by varying the noise level p_e, such that the expected percentage of extra edges in $\mathcal{G}_2$ is equal to a fixed number between 1% and 20% of the total edges in $\mathcal{G}_1$. Moreover, for each noise-level, we averaged our results over 20 Monte-Carlo runs. Notice, that while the considered problem setup guarantees the existence of an alignment (i.e., $\exists\ \mathbf{P}^* \in \operatorname{argmin}_{\mathbf{P}\in\mathcal{P}^n} \left\|\mathbf{A}_1 - \mathbf{P}\mathbf{A}_2\mathbf{P}^T\right\|_F^2$), the uniqueness of the minimizer is not guaranteed, in general, because of the possible presence of topologically-invariant subgraphs, such as cliques and star graphs [32]. Consequently, our aim is to compute an alignment between the two graphs which can minimize the induced edge disagreements without having any *apriori* knowledge.

6.4 Evaluation Metrics

In our experiments, we evaluate the performance of all considered methods based on the ratio of the number of edge overlaps induced by the algorithm over the number of edges in $\mathcal{G}_A$ (Edge correctness). In general, the higher the score, the better. Perfectly aligning the edge-set of $\mathcal{G}_1$ with its counterpart in $\mathcal{G}_2$ results in the maximum correctness score of 1.

6.5 Initialization, convergence criteria, and parameter selection

Initialization: In Algorithm 1, we require to specify an initial choice of the variable $\mathbf{U}^0$. As our problem formulation is non-convex, this choice of initialization is of significant importance. Given an instance of graph alignment, however, it is a non-trivial task to "handcraft" a meaningful initialization. Hence, in our experiments, we use our proposed algorithm to refine the solutions provided by the considered baselines. More specifically, we use the output produced by CONE-Align (i.e., the output of the kd-tree stage) to initialize our algorithm. This choice is determined by our empirical observation that CONE-Align achieves the best trade off between accuracy and complexity among the considered baselines, as well as by the substantial and consistent improvement we obtained after using it as initialization for our method.

Convergence criterion: In order to use Algorithm 1, we need to specify a termination criterion. In our experiments, we terminate Algorithm 1 when the relative change of factor $\mathbf{U}$, $\frac{\left\|\mathbf{U}^k - \mathbf{U}^{k-1}\right\|_F}{\sqrt{n}}$ is smaller than a predetermined *threshold* or when the number of iterations exceeds a predetermined number of iterations, K_{max}. Specifically, we set $threshold = 10^{-2}$ and $K_{max} = 60$.

Choice of parameters: Our algorithm requires specification of three parameters: (i) the embedding dimension $d \leq n$, (ii) the parameter λ which reflects the level of non-isomorphism between the two graphs, and (iii) the penalty parameter ρ which regulates the degree of violation of the equality constraint between variables $\mathbf{U}$ and $\mathbf{W}$. The complete set of parameters can be found listed in Table 2. Regarding the choice of the first parameter, we observed that increasing the embedding dimension linearly with the size of the graph $\mathcal{G}_1$ always yields best performance, i.e., the larger the graph, the greater the number of pivots of the target graph $\mathcal{G}_1$ and the node embedding dimension of the query graph $\mathcal{G}_2$ required for obtaining a high-quality solution. Meanwhile, in our experimental setup, as the noisy graph $\mathcal{G}_2$ with extra edges becomes less isomorphic compared to $\mathcal{G}_1$ with increase in noise-level, we gradually decrease the value of λ from its initial setting in the noiseless (i.e., isomorphic) case, which was determined via trial-and-error. Finally,

Table 2: Summary of parameters used for each dataset in our experiments: the embedding dimension (d), the "isomorphism" parameter (λ) for each noise-level (from left to right - noiseless (0%) to highest noise-level (20%)), and the penalty parameter (ρ).

Graph	d	λ	ρ
C. Elegans	128	[5, 5, 5, 5, 5, 5]	0.1
arenas-Email	256	[20, 20, 20, 20, 20, 20]	0.2
polBlog	256	[20, 20, 20, 20, 20, 20]	0.1
Airports	350	[30, 30, 30, 30, 30, 30]	0.3
A. Thaliana	520	[9, 9, 9, 9, 9, 9]	0.1
Japanese Book	750	[14.5, 14.5, 14.5, 14.5, 14.5, 14.5]	0.1
HomoSapiens	750	[13.3, 13.3, 13.3, 13.3, 13.3, 13.3]	0.1
ca-GrQc	950	[4, 4, 4, 4, 4, 3.9]	0.1

regarding the choice of the penalty parameter, for a given dataset, we observed that fixing ρ to a modest value in the interval $[0.1, 0.3]$ works well across all noise-levels. Throughout our experiments, we observed that our proposed approach is not very sensitive to parameter tuning.

6.6 Results and Discussion

In this section, we present the results of our experiments obtained with our proposed graph alignment method and contrast it with that of the selected baselines. In Figure (1), we illustrate the accuracy scores, in terms of edge correctness, that each method achieved. Our main findings are as follows

- In the presence of noise, our method outperforms all baselines, often by a significant margin. It is also evident that the performance of our method is the least sensitive to noise, as its accuracy score degrades more gracefully compared to the other methods. For the highest noise level scenario, where the considered graphs differ by up to 20% in terms of number of edges and are far away from being near-isomorphic, the improvement in performance can be as large as $80 - 100\%$ over the next best competitor (e.g., see results for polBlogs and A.Thaliana). This observation implies that the computed alignment successfully identifies a subset of edges in graph $\mathcal{G}_2$ that has high overlap with the edges of graph $\mathcal{G}_1$, even in the presence of strong noise.
- In the noiseless setting, (i.e., the isomorphic case), it can be seen that our method does not attain the optimal score in general. We attribute this to the fact that CONE-Align does not perform well in this scenario, and consequently the initialization offered by it to our algorithm is non-ideal. Furthermore, we noted that by increasing the maximum number of iterations to a larger value, we can refine the solution of CONE-Align to attain the optimal edge correctness score. However, such a step comes at the expense of increased complexity and is thus formally omitted from our experimental settings.
- In terms of timing, Figure (2) reveals that our algorithm is more time consuming compared to the baselines, which is a consequence of the fact that we have to perform an SVD at each step of the alternating minimization procedure. However, the the complexity of our algorithm is by no means unaffordable, exhibiting a run-time that is of the same order as that of CONE-Align on all datasets, with the exception of A.Thaliana.

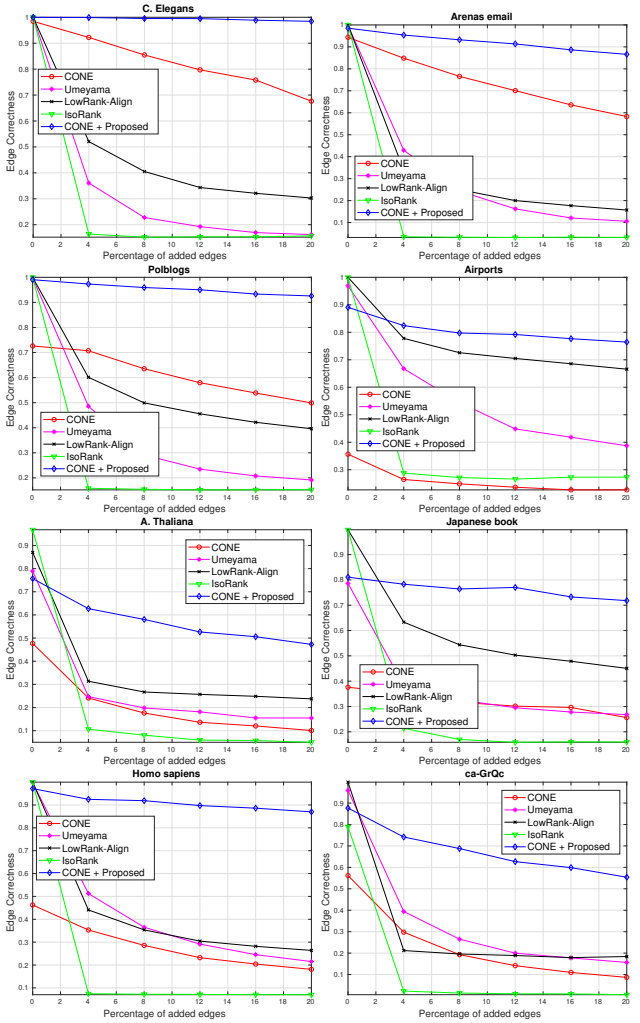

Figure 1: Edge correctness vs. noise level across networks. For each value of noise level (p_e), 10 different realizations of the graphs $\mathcal{G}_2$, with a certain percentage of additional edges and under a different and random permutation, were generated. The number of additional edges varied from 0 to 20% of the total number of edges of the fixed graph $\mathcal{G}_1$.

7 CONCLUSIONS & FUTURE WORK

In this work, we proposed a novel formulation of the classical graph alignment problem, which combines the merits of the prominent class of spectral methods and the promising class of graph embedding based methods. In order to examine its effectiveness, we developed an alternating minimization algorithm to solve it and we compared it against the prevailing *state-of-the-art* graph alignment

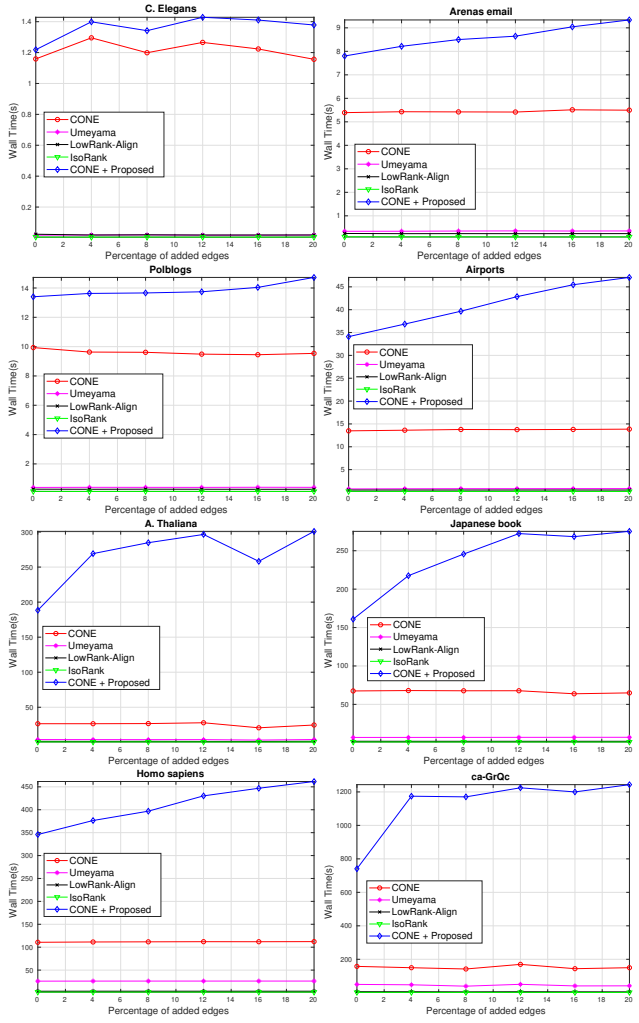

Figure 2: Wall time (in seconds) vs. noise level for different networks.

frameworks on real life graphs. Our results show that our proposed framework achieves *much higher* alignment accuracy relative to the prior art, in challenging problem instances involving noisy real-life graphs. This currently comes at the cost of higher complexity due to SVD, but a key point is that existing frameworks leave a lot of room for improvement in terms of the attainable alignment accuracy. Designing more efficient and scalable methods for solving the proposed formulation, as well as evaluating the quality of the obtained embeddings for other downstream tasks (such as link prediction) are the subject of ongoing research.

ACKNOWLEDGMENTS

This research has been partially supported by the National Science Foundation under Grants IIS-1908070 and ECCS-1807660.

REFERENCES

[1] Amr Ahmed, Nino Shervashidze, Shravan Narayanamurthy, Vanja Josifovski, and Alexander J Smola. 2013. Distributed large-scale natural graph factorization. In *Proceedings of the 22nd international conference on World Wide Web*. 37–48.

[2] Jason Altschuler, Jonathan Niles-Weed, and Philippe Rigollet. 2017. Near-linear time approximation algorithms for optimal transport via Sinkhorn iteration. In *Advances in neural information processing systems*. 1964–1974.

[3] László Babai, D Yu Grigoryev, and David M Mount. 1982. Isomorphism of graphs with bounded eigenvalue multiplicity. In *Proceedings of the fourteenth annual ACM symposium on Theory of computing*. 310–324.

[4] Rainer E. Burkard. 2013. Quadratic Assignment Problems. In *Handbook of Combinatorial Optimization*. Springer New York, 2741–2814. https://doi.org/10.1007/978-1-4419-7997-1_22

[5] Hongyun Cai, Vincent W Zheng, and Kevin Chen-Chuan Chang. 2018. A comprehensive survey of graph embedding: Problems, techniques, and applications. *IEEE Transactions on Knowledge and Data Engineering* 30, 9 (2018), 1616–1637.

[6] Shaosheng Cao, Wei Lu, and Qiongkai Xu. 2015. Grarep: Learning graph representations with global structural information. In *Proceedings of the 24th ACM international on conference on information and knowledge management*. 891–900.

[7] Bilian Chen, Simai He, Zhening Li, and Shuzhong Zhang. 2012. Maximum block improvement and polynomial optimization. *SIAM Journal on Optimization* 22, 1 (2012), 87–107.

[8] Haochen Chen, Bryan Perozzi, Yifan Hu, and Steven Skiena. 2017. Harp: Hierarchical representation learning for networks. *arXiv preprint arXiv:1706.07845* (2017).

[9] Xiyuan Chen, Mark Heimann, Fatemeh Vahedian, and Danai Koutra. 2020. Consistent Network Alignment with Node Embedding. *arXiv preprint arXiv:2005.04725* (2020).

[10] Minsu Cho, Jungmin Lee, and Kyoung Mu Lee. 2010. Reweighted random walks for graph matching. In *European conference on Computer vision*. Springer, 492–505.

[11] Donatello Conte, Pasquale Foggia, Carlo Sansone, and Mario Vento. 2004. Thirty years of graph matching in pattern recognition. *International journal of pattern recognition and artificial intelligence* 18, 03 (2004), 265–298.

[12] Timothee Cour, Praveen Srinivasan, and Jianbo Shi. 2007. Balanced graph matching. In *Advances in Neural Information Processing Systems*. 313–320.

[13] Marco Cuturi. 2013. Sinkhorn distances: Lightspeed computation of optimal transport. In *Advances in neural information processing systems*. 2292–2300.

[14] Tyler Derr, Hamid Karimi, Xiaorui Liu, Jiejun Xu, and Jiliang Tang. 2019. Deep adversarial network alignment. *arXiv preprint arXiv:1902.10307* (2019).

[15] John Duchi, Shai Shalev-Shwartz, Yoram Singer, and Tushar Chandra. 2008. Efficient projections onto the l 1-ball for learning in high dimensions. In *Proceedings of the 25th international conference on Machine learning*. 272–279.

[16] Lars Eldén and Haesun Park. 1999. A Procrustes problem on the Stiefel manifold. *Numer. Math.* 82, 4 (1999), 599–619.

[17] Frank Emmert-Streib, Matthias Dehmer, and Yongtang Shi. 2016. Fifty years of graph matching, network alignment and network comparison. *Information sciences* 346 (2016), 180–197.

[18] Soheil Feizi, Gerald Quon, Mariana Mendoza, Muriel Medard, Manolis Kellis, and Ali Jadbabaie. 2019. Spectral alignment of graphs. *IEEE Transactions on Network Science and Engineering* (2019).

[19] Gene H. Golub and Charles F. Van Loan. 1996. *Matrix Computations* (third ed.). The Johns Hopkins University Press.

[20] Edouard Grave, Armand Joulin, and Quentin Berthet. 2019. Unsupervised alignment of embeddings with wasserstein procrustes. In *The 22nd International Conference on Artificial Intelligence and Statistics*. 1880–1890.

[21] Aditya Grover and Jure Leskovec. 2016. node2vec: Scalable feature learning for networks. In *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*. 855–864.

[22] William L Hamilton, Rex Ying, and Jure Leskovec. 2017. Representation learning on graphs: Methods and applications. *arXiv preprint arXiv:1709.05584* (2017).

[23] Mark Heimann, Haoming Shen, Tara Safavi, and Danai Koutra. 2018. Regal: Representation learning-based graph alignment. In *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*. 117–126.

[24] John E Hopcroft and Jin-Kue Wong. 1974. Linear time algorithm for isomorphism of planar graphs (preliminary report). In *Proceedings of the sixth annual ACM symposium on Theory of computing*. 172–184.

[25] Bo Jiang, Haifeng Zhao, Jin Tang, and Bin Luo. 2014. A sparse nonnegative matrix factorization technique for graph matching problems. *Pattern Recognition* 47, 2 (2014), 736–747.

[26] Paul J Kelly et al. 1957. A congruence theorem for trees. *Pacific J. Math.* 7, 1 (1957), 961–968.

[27] Aritra Konar and Nicholas D. Sidiropoulos. 2020. Graph Matching via the lens of Supermodularity. *IEEE Transactions on Knowledge and Data Engineering* (2020).

[28] Bernhard Korte and Dirk Hausmann. 1978. An analysis of the greedy heuristic for independence systems. In *Annals of Discrete Mathematics*. Vol. 2. Elsevier, 65–74.

[29] Jérôme Kunegis. 2013. Konect: the koblenz network collection. In *Proceedings of the 22nd International Conference on World Wide Web*. 1343–1350.

[30] Marius Leordeanu, Martial Hebert, and Rahul Sukthankar. 2009. An integer projected fixed point method for graph matching and map inference. In *Advances*

in neural information processing systems. 1114–1122.
[31] Jure Leskovec and Andrej Krevl. 2014. SNAP Datasets: Stanford large network dataset collection.
[32] Yongsub Lim, U Kang, and Christos Faloutsos. 2014. Slashburn: Graph compression and mining beyond caveman communities. *IEEE Transactions on Knowledge and Data Engineering* 26, 12 (2014), 3077–3089.
[33] Eliane Maria Loiola, Nair Maria Maia de Abreu, Paulo Oswaldo Boaventura-Netto, Peter Hahn, and Tania Querido. 2007. A survey for the quadratic assignment problem. *European journal of operational research* 176, 2 (2007), 657–690.
[34] Huda Nassar, Nate Veldt, Shahin Mohammadi, Ananth Grama, and David F Gleich. 2018. Low rank spectral network alignment. In *Proceedings of the 2018 World Wide Web Conference*. 619–628.
[35] Mingdong Ou, Peng Cui, Jian Pei, Ziwei Zhang, and Wenwu Zhu. 2016. Asymmetric transitivity preserving graph embedding. In *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*. 1105–1114.
[36] Christos H Papadimitriou and Kenneth Steiglitz. 1998. *Combinatorial optimization: algorithms and complexity*. Courier Corporation.
[37] Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. 2014. Deepwalk: Online learning of social representations. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*. 701–710.
[38] Jiezhong Qiu, Yuxiao Dong, Hao Ma, Jian Li, Kuansan Wang, and Jie Tang. 2018. Network embedding as matrix factorization: Unifying deepwalk, line, pte, and node2vec. In *Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining*. 459–467.
[39] Sartaj Sahni and Teofilo Gonzalez. 1976. P-complete approximation problems. *Journal of the ACM (JACM)* 23, 3 (1976), 555–565.
[40] Peter H Schönemann. 1966. A generalized solution of the orthogonal procrustes problem. *Psychometrika* 31, 1 (1966), 1–10.
[41] Mark R Sepanski. 2007. *Compact lie groups*. Vol. 235. Springer Science & Business Media.
[42] Rohit Singh, Jinbo Xu, and Bonnie Berger. 2007. Pairwise global alignment of protein interaction networks by matching neighborhood topology. In *Annual International Conference on Research in Computational Molecular Biology*. Springer, 16–31.
[43] Shinji Umeyama. 1988. An eigendecomposition approach to weighted graph matching problems. *IEEE transactions on pattern analysis and machine intelligence* 10, 5 (1988), 695–703.
[44] Hongteng Xu, Dixin Luo, Hongyuan Zha, and Lawrence Carin. 2019. Gromov-wasserstein learning for graph matching and node embedding. *arXiv preprint arXiv:1901.06003* (2019).
[45] Mikhail Zaslavskiy, Francis Bach, and Jean-Philippe Vert. 2008. A path following algorithm for the graph matching problem. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 31, 12 (2008), 2227–2242.
[46] Meng Zhang, Yang Liu, Huanbo Luan, and Maosong Sun. 2017. Earth mover's distance minimization for unsupervised bilingual lexicon induction. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*. 1934–1945.
[47] Si Zhang and Hanghang Tong. 2016. Final: Fast attributed network alignment. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 1345–1354.
[48] Feng Zhou and Fernando De la Torre. 2012. Factorized graph matching. In *2012 IEEE Conference on Computer Vision and Pattern Recognition*. IEEE, 127–134.