Multilevel Network Alignment

Si Zhang
Arizona State University
szhan172@asu.edu

Hanghang Tong
Arizona State University
hanghang.tong@asu.edu

Ross Maciejewski
Arizona State University
rmacieje@asu.edu

Tina Eliassi-Rad
Northeastern University
tina@eliassi.org

ABSTRACT

Network alignment, which aims to find the node correspondence across multiple networks, is a fundamental task in many areas, ranging from social network analysis to adversarial activity detection. The state-of-the-art in the data mining community often view the node correspondence as a probabilistic cross-network node similarity, and thus inevitably introduce an $\Omega(n^2)$ lower bound on the computational complexity. Moreover, they might ignore the rich patterns (e.g., clusters) accompanying the real networks. In this paper, we propose a multilevel network alignment algorithm (MOANA) which consists of three key steps. It first efficiently *coarsens* the input networks into their structured representations, and then *aligns* the coarsest representations of the input networks, followed by the *interpolations* to obtain the alignment at multiple levels including the node level at the finest granularity. The proposed *coarsen-align-interpolate* method bears two key advantages. First, it overcomes the $\Omega(n^2)$ lower bound, achieving a *linear* complexity. Second, it helps reveal the alignment between rich patterns of the input networks at multiple levels (e.g., node, clusters, super-clusters, etc.). Extensive experimental evaluations demonstrate the efficacy of the proposed algorithm on both the node-level alignment and the alignment among rich patterns (e.g., clusters) at different granularities.

KEYWORDS

Network alignment; multilevel alignment; multiresolution

ACM Reference Format:
Si Zhang, Hanghang Tong, Ross Maciejewski, and Tina Eliassi-Rad. 2019. Multilevel Network Alignment. In *Proceedings of the 2019 World Wide Web Conference (WWW '19), May 13–17, 2019, San Francisco, CA, USA.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3308558.3313484

1 INTRODUCTION

Networks are often *multi-sourced* (i.e., variety) and *large-scale* (i.e., volume), such as the financial networks built from the massive size of transactions in different institutes, the social networks with a substantial number of users on various platforms and so on. Whether or not subtle patterns associated with different networks (e.g., a complex fraud schema) can be distilled, profoundly depends on the successful integration of the multi-sourced network data. Network alignment that aims to uncover the node correspondence across different networks has drawn a lot of attention in many applications, including social network analysis [37], bioinformatics [10], adversarial activity detection [31], etc. For instance, financial criminals (e.g., money launderers [38, 39]) often camouflage themselves by frequently conducting less suspicious activities in different financial transaction networks, making it hard to detect them if we only focus on a single network. By finding the correspondence among the suspects across different networks, network alignment could help unveil such complex fraud transaction patterns.

Generally speaking, network alignment can be categorized into *local* alignment and *global* alignment. Among others, local network alignment essentially intends to find the similar motifs across different networks [3]. Such local alignment methods might be too restrictive to find the node correspondence, or to align the larger complex structures across different networks. On the other hand, a prevalent choice for global alignment in the data mining community is the soft alignment that views the node correspondence as a probabilistic cross-network node similarity (e.g., *IsoRank* [27], *FINAL* [35]). However, it suffers from the following drawbacks. First, the soft alignment inevitably introduces an $\Omega(n^2)$ lower bound on the computational complexity due to the dense alignment matrix, where n is the number of nodes. Second, most soft alignment methods (e.g., *BigAlign* [16]) assume one network is a noisy permutation of the other and perform the alignment at a single node level. Consequently, these approaches might ignore the rich patterns accompanying networks, e.g., clusters, etc.

Some recent efforts attempt to go beyond this assumption and perform the alignment at two levels [2, 20]. In addition to the node-level alignment, these methods aim to find the alignment at the coarser level, for instance, among clusters. Compared to the node-level alignment, these methods not only break the $\Omega(n^2)$ floor of the computational complexity by viewing the node-level alignment as the interpolation of the cluster-level alignment, but also reveal the more complicated alignment across the input networks (e.g., which cluster in network $\mathcal{G}_1$ links to which cluster in network $\mathcal{G}_2$).

Nonetheless, these two-level alignment methods still bear some fundamental limitations. First (*problem setting*), the existing methods are restricted to only two levels, and therefore might overlook some richer patterns in the underlying networks, e.g., the hierarchical *cluster-within-clusters* structure [25]. Second (*alignment accuracy*), it is not clear how the coarser-level alignment would impact the accuracy of the finer-level alignment. This is crucial especially if we want to perform the multilevel alignment, as the error at one level could be propagated or even diverged through different levels. Third (*computational efficiency*), most of the existing

WWW '19, May 13–17, 2019, San Francisco, CA, USA

ACM ISBN 978-1-4503-6674-8/19/05.
https://doi.org/10.1145/3308558.3313484

two-level approaches, with the only exception of [36], still requires a *sub-quadratic* computational complexity, e.g., [2].

In this paper, we aim to systematically address the above limitations and the main contributions are summarized as follows:

- **Problem Definition.** To our best knowledge, we are the first to address the multilevel network alignment problem.
- **Algorithm and Analysis.** We propose a multilevel network alignment algorithm (MOANA), which scales *linearly* w.r.t. the size of the input networks. In theory, we prove a condition on the perfect interpolation, which in principle allows us to perform the alignment at an arbitrary number of levels, without any interpolation error. In practice, we drive an error bound of alignment due to the coarsening step.
- **Experiments.** We perform extensive experiments, which demonstrate that (1) the proposed algorithm achieves a close node-level alignment accuracy to its single-level counterpart *FINAL* [35] with an up to 10× speedup, in the meanwhile outperforming all the other existing node-level alignment algorithms. Note that the proposed MOANA can be considered as an approximation of *FINAL* at the finest node level; (2) it scales *linearly* w.r.t. the size of the networks; and (3) it also reveals the alignment among the more complicated cluster structures at different granularities.

2 PROBLEM DEFINITION

Table 1 summarizes the main symbols and notations used throughout the paper. We use the bold uppercase letters to denote matrices (e.g., $\mathbf{A}$), bold lowercase letters for vectors (e.g., $\mathbf{s}$) and letters not in bold for scalars (e.g., α). We use $\mathbf{A}(i,j)$ to denote the entry at the intersection of the i-th row and j-th column of the matrix $\mathbf{A}$. We denote the transpose of a matrix by a superscript T (e.g., $\mathbf{A}^T$ as the transpose of $\mathbf{A}$). We use the subscripts to index the matrices at different levels. For example, $\mathbf{A}_1$ represents the symmetrically normalized matrix of the adjacency matrix $\bar{\mathbf{A}}_1$ of network $\mathcal{G}_1$ at the first level (i.e., $\mathbf{A}_1 = \mathbf{D}_1^{-\frac{1}{2}}\bar{\mathbf{A}}_1\mathbf{D}_1^{-\frac{1}{2}}$ where $\mathbf{D}_1$ is the diagonal degree matrix of $\bar{\mathbf{A}}_1$), and $\mathbf{A}_l$ is the corresponding coarsened adjacency matrix at the l-th level ($l \geq 2$). The identity matrix of size $n \times n$ is denoted by $\mathbf{I}_n$. Throughout this paper, we simplify $\mathbf{I}_n$ to $\mathbf{I}$ if the size is clear from the context. Without loss of generality, we assume that $\mathcal{G}_1$ and $\mathcal{G}_2$ share a comparable size, i.e., $O(n_1) = O(n_2) = O(n)$ and $O(m_1) = O(m_2) = O(m)$, to simplify the complexity analysis.

2.1 Multilevel Network Alignment Problem

Many large real-world networks often have the complex hierarchical *cluster-within-clusters* structures. For example, given the input network $\mathcal{G}_1$ in Figure 1 (a), nodes $\{1,3\}$ can be viewed as a two-node cluster and similarly nodes $\{2,12\}$ form another two-node cluster (in Figure 1 (b)). These two clusters are represented by node-1 and node-2 respectively at the second level, which further construct an even coarser two-node cluster (in Figure 1 (c)). This coarser cluster is represented by node-1 at the third level (in Figure 1 (d)). As one can see, each level reveals a unique pattern in the underlying network $\mathcal{G}_1$. This leads to the following question for the alignment task: *How can we align the clusters at different granularities across the input networks?* Such an alignment can not only uncover how the networks are aligned at the different cluster levels, but also might help the node-level alignment under the following hypothesis. If the cluster represented by node-1 in $\mathcal{G}_1$ is aligned with that represented by node-6′ in $\mathcal{G}_2$ at the coarsest level (in Figure 1 (e)), it might provide us with clues on how the two-node clusters represented by node-1 and node-2 are aligned with those represented by node-6′ and node-3′ at the next finer level (i.e., level-2). Furthermore, it might indicate how nodes $\{1,2,3,12\}$ are aligned with $\{6',3',11',13'\}$ at the original node level (in Figure 1 (f)). Formally, we define the multilevel network alignment problem.

Table 1: Symbols and Notations

Symbols	Definition
$\mathcal{G}_1$, $\mathcal{G}_2$	the input undirected networks
$\bar{\mathbf{A}}_1$, $\bar{\mathbf{B}}_1$	the adjacency matrices of $\mathcal{G}_1$ and $\mathcal{G}_2$
$\mathbf{A}_1$, $\mathbf{B}_1$	the symmetric normalization by degree matrix of $\bar{\mathbf{A}}_1$, $\bar{\mathbf{B}}_1$
$\mathbf{A}_l$, $\mathbf{B}_l$	the coarsened adjacency matrices at the l-th level
$\mathbf{P}_l$, $\mathbf{Q}_l$	the interpolation matrices at the l-th level
$\mathbf{H}_l$	the $n_2 \times n_1$ prior similarity matrix at the l-th level
$\mathbf{S}_l^*$	the output $n_2 \times n_1$ alignment matrix at the l-th level
$\mathcal{S}_{A_l}$, $\mathcal{S}_{B_l}$	the active indices of $\mathbf{A}_l$, $\mathbf{B}_l$ at the l-th level
n_1, n_2	# of nodes in $\mathcal{G}_1$, $\mathcal{G}_2$
m_1, m_2	# of edges in $\mathcal{G}_1$, $\mathcal{G}_2$
L	# of levels
$\mathrm{Tr}(\cdot)$, $\mathrm{nnz}(\cdot)$	the trace and the number of nonzero elements of a matrix
$\mathrm{vec}(\cdot)$, $\mathrm{mat}(\cdot)$	the vectorization and de-vectorization operators
$\otimes$	the Kronecker product operator

PROBLEM 1. *MULTILEVEL NETWORK ALIGNMENT PROBLEM.*

Given: *(1) adjacency matrices $\bar{\mathbf{A}}_1, \bar{\mathbf{B}}_1$ of two undirected (weighted) sparse networks $\mathcal{G}_1, \mathcal{G}_2$, (2) a sparse prior node similarity matrix $\mathbf{H}_1$ across two networks, and (3) the number of levels $L \geq 2$.*

Output: *a set of $n_2 \times n_1$ alignment matrices $\mathbf{S}_l^*$, $l = 1, \cdots, L$, where specifically $\mathbf{S}_1^*(x,a)$ represents the confidence that node-a in $\mathcal{G}_1$ is aligned with node-x in $\mathcal{G}_2$ at the input node level.*

2.2 Preliminaries #1: Node-Level Alignment

A classic node-level alignment algorithm is attributed to *IsoRank* [27] based on the *alignment consistency* by random walks on the product graph. With an additional normalization, it has the following optimization interpretation referred to as *FINAL* in [35].

$$\min_{\mathbf{s}_1} \ \alpha \mathbf{s}_1^T(\mathbf{I} - \mathbf{A}_1 \otimes \mathbf{B}_1)\mathbf{s}_1 + (1-\alpha)\|\mathbf{s}_1 - \mathbf{h}_1\|_2^2 \tag{1}$$

where $\mathbf{s}_1, \mathbf{h}_1$ are the vectorization of the alignment matrix $\mathbf{S}_1$ and the prior similarity matrix $\mathbf{H}_1$, respectively. $\mathbf{A}_1$ and $\mathbf{B}_1$ are the symmetrically normalized adjacency matrices by the degree matrices of $\bar{\mathbf{A}}_1$ and $\bar{\mathbf{B}}_1$ respectively. Besides, the prior node similarity $\mathbf{h}_1$ is to encode the prior alignment information (e.g., anchor links across two networks) and prevent trivial solutions (e.g., $\mathbf{s}_1 = \mathbf{0}$). When such anchor links are absent, we can construct it by some heuristics, e.g., degree similarity. Our proposed algorithm will be based on Eq. (1). The optimization in Eq. (1) is convex so that a global optimal solution can be obtained by the fixed-point algorithm:

$$\mathbf{S}_1 = \alpha \mathbf{B}_1 \mathbf{S}_1 \mathbf{A}_1 + (1-\alpha)\mathbf{H}_1 \tag{2}$$

with an $O(n^2)$ time complexity even with approximations [35].

2.3 Preliminaries #2: MMF

Although many multilevel methods (e.g., hierarchical clustering, algebraic multigrid) exist for mining a single network, it turns out that multiresolution matrix factorization (MMF) [15, 29] is particularly suited for network alignment task for the reason that we will explain in Section 3.2. Here, we give a short review of MMF.

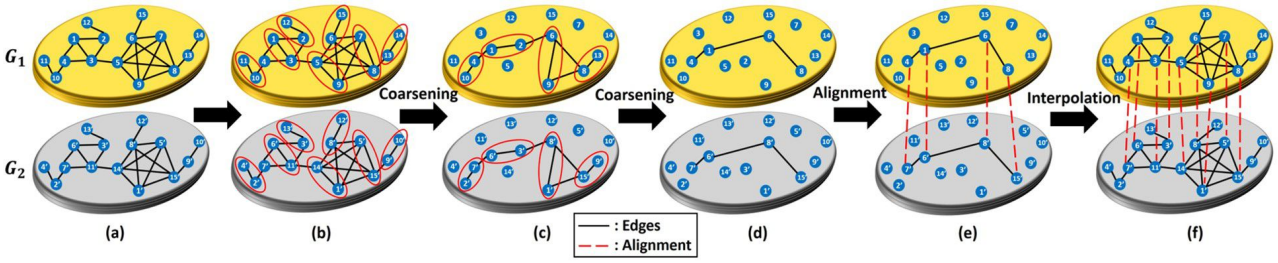


Figure 1: An illustrative example. (a) shows the input networks. (b)-(d) show the network coarsening process. (e) presents the alignment at the coarsest level and (f) shows parts of the node-level alignment across the input networks after interpolations.

Given a symmetric matrix $\mathbf{A}_1 \in R^{n_1 \times n_1}$, MMF explores the hierarchical cluster-within-clusters structure underlying the matrix $\mathbf{A}_1$ and factorizes $\mathbf{A}_1$ into several sparse orthogonal matrices and a core-diagonal matrix. This multilevel factorization is of the form

$$\mathbf{P}_{L-1} \cdots \mathbf{P}_1 \mathbf{A}_1 \mathbf{P}_1^T \cdots \mathbf{P}_{L-1}^T = \mathbf{A}_L \approx \tilde{\mathbf{A}}_L \tag{3}$$

where $\mathbf{P}_1, \cdots, \mathbf{P}_{L-1}$ are the sparse orthogonal matrices and the matrix $\tilde{\mathbf{A}}_L$ is the $\mathcal{S}_L$-core-diagonal matrix defined as below.

Definition 1. ***$\mathcal{S}_L$-core-diagonal matrix [15].*** *Denote a set of active indices as $\mathcal{S}_L = \{i_1, \cdots, i_k\} \subset \{1, \cdots, n_1\}$. Then a matrix $\tilde{\mathbf{A}}_L$ is defined as the $\mathcal{S}_L$-core-diagonal matrix under the condition that $\tilde{\mathbf{A}}_L(i,j) = 0$ unless $i = j$ or $i, j \in \mathcal{S}_L$.*

Throughout this paper, we refer to $\tilde{\mathbf{A}}_L(\mathcal{S}_L, \mathcal{S}_L)$ as the core matrix of $\tilde{\mathbf{A}}_L$. Note that $\tilde{\mathbf{A}}_L(\mathcal{S}_L, \mathcal{S}_L) = \mathbf{A}_L(\mathcal{S}_L, \mathcal{S}_L)$ according to how $\tilde{\mathbf{A}}_L$ is generated (see Appendix A). To compute these sparse orthogonal matrices, [15, 29] consider using the elementary rotation and compound rotation matrices. Among others, using the compound rotation matrices (see Appendix A) yields a more interpretable and compact hierarchical structure, which commonly resides in the underlying network. Therefore, in this paper, we use the parallel MMF with the compound rotations.

3 MULTILEVEL NETWORK ALIGNMENT

In this section, we present our solution to Problem 1. First, we present our multilevel network alignment framework from the optimization perspective. We analyze the theoretical condition on the perfect interpolation which allows no interpolation error across multiple levels. Then, we introduce the details of the proposed algorithm Moana, followed by further analysis of our algorithm.

3.1 Multilevel Optimization Formulation

Generally speaking, most of the existing multilevel approaches to graph mining problems (e.g., graph partitioning) follow similar strategies. They first hierarchically coarsen the network so that at the coarsest level, an approximation solution to the original problem can be obtained within an insignificant time. This approximation solution will then be sequentially projected to the finer levels by the *interpolation* matrices. Inspired by this generic strategy, our key idea is to use the interpolation matrices to efficiently estimate the alignment matrix at the finer level (e.g., $\mathbf{S}_l$ at the l-th level) from that at the next coarser level (e.g., $\mathbf{S}_{l+1}$). Since the alignment matrices hinge on two networks, different from the interpolations underlying a single network, the bilinear interpolations are required.

To derive the optimization formulation for our multilevel network alignment problem, without loss of generality, we focus on the first two levels for now. Denote two interpolation matrices $\mathbf{P}_1 \in R^{p_1 \times n_1}$ and $\mathbf{Q}_1 \in R^{q_1 \times n_2}$ where $p_1 \leq n_1, q_1 \leq n_2$ such that we can approximate the node-level alignment matrix $\mathbf{S}_1$ by $\mathbf{S}_1 = \mathbf{Q}_1^T \mathbf{S}_2 \mathbf{P}_1$ where $\mathbf{S}_2 \in R^{q_1 \times p_1}$ is the alignment matrix at the second level. By de-vectorization on $\mathbf{s}_1, \mathbf{h}_1$, Eq. (1) is equivalent to

$$\min_{\mathbf{S}_1} \ \alpha[\mathrm{Tr}(\mathbf{S}_1^T \mathbf{S}_1) - \mathrm{Tr}(\mathbf{S}_1^T \mathbf{B}_1 \mathbf{S}_1 \mathbf{A}_1)] + (1-\alpha)\|\mathbf{S}_1 - \mathbf{H}_1\|_F^2 \tag{4}$$

Plugging in $\mathbf{S}_1 = \mathbf{Q}_1^T \mathbf{S}_2 \mathbf{P}_1$, we have the objective function w.r.t. $\mathbf{S}_2$.

$$\begin{aligned} J(\mathbf{S}_2) =& \alpha[\mathrm{Tr}(\mathbf{P}_1^T \mathbf{S}_2^T \mathbf{Q}_1 \mathbf{Q}_1^T \mathbf{S}_2 \mathbf{P}_1) - \mathrm{Tr}(\mathbf{S}_2^T \mathbf{Q}_1 \mathbf{B}_1 \mathbf{Q}_1^T \mathbf{S}_2 \mathbf{P}_1 \mathbf{A}_1 \mathbf{P}_1^T)] \\ &+ (1-\alpha)\|\mathbf{Q}_1^T \mathbf{S}_2 \mathbf{P}_1 - \mathbf{H}_1\|_F^2 \end{aligned} \tag{5}$$

Notice that if the (semi-) orthogonality satisfies, i.e., $\mathbf{P}_1 \mathbf{P}_1^T = \mathbf{I}$ and $\mathbf{Q}_1 \mathbf{Q}_1^T = \mathbf{I}$, we can obtain the objective function at the second level which is of exactly the same form as Eq. (4), i.e.,

$$J(\mathbf{S}_2) = \alpha[\mathrm{Tr}(\mathbf{S}_2^T \mathbf{S}_2) - \mathrm{Tr}(\mathbf{S}_2^T \mathbf{B}_2 \mathbf{S}_2 \mathbf{A}_2)] + (1-\alpha)\|\mathbf{S}_2 - \mathbf{H}_2\|_F^2 \tag{6}$$

where $\mathbf{A}_2 = \mathbf{P}_1 \mathbf{A}_1 \mathbf{P}_1^T$, $\mathbf{B}_2 = \mathbf{Q}_1 \mathbf{B}_1 \mathbf{Q}_1^T$ and $\mathbf{H}_2 = \mathbf{Q}_1 \mathbf{H}_1 \mathbf{P}_1^T$. Equivalently, this can be viewed as coarsening $\mathbf{A}_1, \mathbf{B}_1$ into $\mathbf{A}_2, \mathbf{B}_2$ to be aligned at the second level by the interpolation matrices $\mathbf{P}_1$ and $\mathbf{Q}_1$, with the corresponding prior node similarity matrix $\mathbf{H}_2$.

In fact, the relationship between Eq. (4) and Eq. (6) applies to any two adjacent levels among all the L levels. This implies that as we coarsen the symmetrically normalized adjacency matrices of the input networks (i.e., $\mathbf{A}_1, \mathbf{B}_1$) into the coarsest level (i.e., $\mathbf{A}_L, \mathbf{B}_L$), we are always solving the same optimization problem but w.r.t. the different variables to find the alignment at different levels (e.g., $\mathbf{S}_L$). The benefits of this are three-fold. First, this ensures that the same theoretical properties and optimization algorithms at the finest node level (e.g., the convexity, the optimality, the convergence from [35]) will immediately apply to the coarse levels as well. Second, similar to most multilevel approaches, solving the alignment problem at the coarsest level is computationally more efficient. Third, provided that a set of 'good' (semi-) orthogonal interpolation matrices $\mathbf{P}_l, \mathbf{Q}_l$ can be found, we are able to obtain a set of well-represented adjacency matrices at the coarse levels that reveal the hierarchical cluster-within-clusters structure of the input networks.

3.2 Perfect Interpolation

Despite a large number of the proposed graph coarsening methods, to our best knowledge, few of them bear the (semi-) orthogonality of the interpolation matrices. For example, neither the heavy-edge matching [6, 13] nor the algebraic multigrid (AMG) coarsening [23] guarantees the semi-orthogonality. Moreover, since there are more variables at the finer levels than coarser levels, interpolation might introduce an interpolation error to the alignment matrix at the finer level. Even worse, in the multilevel setting, such an interpolation error could propagate or even diverge through different levels so that the node-level alignment might be sub-optimal or even misleading.

In this paper, instead of exploring the semi-orthogonal interpolation matrices, we seek to find a set of orthogonal matrices, i.e., $\mathbf{P}_l\mathbf{P}_l^T = \mathbf{I}$ and $\mathbf{P}_l^T\mathbf{P}_l = \mathbf{I}$. Indeed, by the following lemma, we show that the orthogonal interpolation matrices guarantee that the interpolation of the optimal alignment matrix from the coarser level is exactly the same as the optimal alignment matrix at the finer level.

Lemma 1. ***Perfect Interpolation.*** *The global optimal solution to the optimization problem at the finer level (e.g., Eq.* (4) *for level-1), denoted by* $\mathbf{S}_l$*, is exactly same as the interpolation of the optimal solution at the next coarser level (denoted by* $\mathbf{S}_{l+1}$*). That is,* $\mathbf{S}_l = \mathbf{Q}_l^T\mathbf{S}_{l+1}\mathbf{P}_l$ *if* $\mathbf{P}_l$ *and* $\mathbf{Q}_l$ *are orthogonal, where* $l = 1, \cdots, L-1$.

Proof. Without loss of generality, we prove $\mathbf{S}_1 = \mathbf{Q}_1^T\mathbf{S}_2\mathbf{P}_1$. For all the other levels, it is straightforward to show the same result. The optimal closed-form solution to Eq. (1) (and equivalently Eq. (4)) is $\mathbf{s}_1 = (1-\alpha)(\mathbf{I} - \alpha\mathbf{A}_1 \otimes \mathbf{B}_1)^{-1}\mathbf{h}_1$. Similarly, the alignment matrix between $\mathbf{A}_2, \mathbf{B}_2$ at the second level is computed by

$$\mathbf{s}_2 = (1-\alpha)[\mathbf{I} - \alpha(\mathbf{P}_1\mathbf{A}_1\mathbf{P}_1^T) \otimes (\mathbf{Q}_1\mathbf{B}_1\mathbf{Q}_1^T)]^{-1}\mathbf{h}_2$$

The difference between $\mathbf{S}_1$ and the interpolated alignment from $\mathbf{S}_2$ measured in the Frobenius norm is

$$\begin{aligned}
&\|\mathbf{Q}_1^T\mathbf{S}_2\mathbf{P}_1 - \mathbf{S}_1\|_F = \|(\mathbf{P}_1^T \otimes \mathbf{Q}_1^T)\mathbf{s}_2 - \mathbf{s}_1\|_2 \\
=&(1-\alpha)\|(\mathbf{P}_1^T \otimes \mathbf{Q}_1^T)[\mathbf{I} - \alpha(\mathbf{P}_1\mathbf{A}_1\mathbf{P}_1^T) \otimes (\mathbf{Q}_1\mathbf{B}_1\mathbf{Q}_1^T)]^{-1}\mathbf{h}_2 \\
&-(\mathbf{I} - \alpha\mathbf{A}_1 \otimes \mathbf{B}_1)^{-1}\mathbf{h}_1\|_2 \\
=&(1-\alpha)\Big\|\sum_{k=0}^{\infty}\alpha^k(\mathbf{P}_1^T \otimes \mathbf{Q}_1^T)[(\mathbf{P}_1\mathbf{A}_1\mathbf{P}_1^T)^k \otimes (\mathbf{Q}_1\mathbf{B}_1\mathbf{Q}_1^T)^k](\mathbf{P}_1 \otimes \mathbf{Q}_1)\mathbf{h}_1 \\
&-\sum_{k=0}^{\infty}\alpha^k(\mathbf{A}_1^k \otimes \mathbf{B}_1^k)\mathbf{h}_1\Big\|_2
\end{aligned}$$

where the third equation is by Neumann series due to the fact that (1) the eigenvalues of $\mathbf{A}_1, \mathbf{B}_1$ are in the range of $[-1, 1]$, and (2) $\mathbf{P}_1\mathbf{A}_1\mathbf{P}_1^T, \mathbf{Q}_1\mathbf{B}_1\mathbf{Q}_1^T$ share the same eigenvalues as $\mathbf{A}_1, \mathbf{B}_1$ respectively given that $\mathbf{P}_1$ and $\mathbf{Q}_1$ are orthogonal [24].

Due to the orthogonality of $\mathbf{P}_1, \mathbf{Q}_1$, the following equations hold.

$$\begin{aligned}
\mathbf{A}_1^k =&\mathbf{P}_1^T(\mathbf{P}_1\mathbf{A}_1\mathbf{P}_1^T)^k\mathbf{P}_1 \\
\mathbf{B}_1^k =&\mathbf{Q}_1^T(\mathbf{Q}_1\mathbf{B}_1\mathbf{Q}_1^T)^k\mathbf{Q}_1
\end{aligned}$$

Thus, we have that

$$\|\mathbf{S}_1 - \mathbf{Q}_1^T\mathbf{S}_2\mathbf{P}_1\|_F^2 = (1-\alpha)\Big\|\sum_{k=0}^{\infty}\alpha^k(\mathbf{A}_1^k \otimes \mathbf{B}_1^k - \mathbf{A}_1^k \otimes \mathbf{B}_1^k)\mathbf{h}_1\Big\|_2 = 0$$

which completes the proof. □

3.3 Multilevel Alignment Algorithm

Our multilevel alignment algorithm Moana contains three phases, including *coarsening* the input networks, *aligning* across the coarsest networks, and then *interpolating* the alignment matrix from the coarser levels to the finer levels.

A - Network coarsening. Based on Lemma 1, we want to find a set of orthogonal matrices to coarsen the input networks. However, even though the input adjacency matrix, say $\mathbf{A}_1$, is sparse, the computational complexity of $\mathbf{P}_1\mathbf{A}_1\mathbf{P}_1^T$ may still be $O(n^3)$ at worst if $\mathbf{P}_1$ is not sufficiently sparse. With an arbitrarily orthogonal matrix, the resulting matrix $\mathbf{A}_2$ at the next coarser level could be less informative and thus could fail to reveal the underlying structural information of $\mathbf{A}_1$. Thus, we seek to find a set of orthogonal interpolation matrices $\mathbf{P}_l$ and $\mathbf{Q}_l$ such that (1) they are sufficiently sparse, and (2) they are able to uncover the *hierarchical cluster-within-clusters* structure of the input networks. In this paper, we leverage a recently proposed multiresolution matrix factorization (MMF) algorithm that satisfies these requirements [15]. Specifically, for each input network, we use the parallel second order MMF algorithm (i.e., $k_1, \cdots, k_p \le 2$ in Eq. (25)) to find a set of rotation matrices $\mathbf{P}_l, \mathbf{Q}_l$ such that at the l-th level ($l \ge 2$),

$$\mathbf{A}_l = \mathbf{P}_{l-1}\cdots\mathbf{P}_1\mathbf{A}_1\mathbf{P}_1^T\cdots\mathbf{P}_{l-1}^T \tag{7}$$

$$\mathbf{B}_l = \mathbf{Q}_{l-1}\cdots\mathbf{Q}_1\mathbf{B}_1\mathbf{Q}_1^T\cdots\mathbf{Q}_{l-1}^T \tag{8}$$

where the active indices of $\mathbf{A}_l, \mathbf{B}_l$ are denoted as $\mathcal{S}_{A_l}$ of size λ_l and $\mathcal{S}_{B_l}$ of size μ_l, respectively. Specifically, at the coarsest level, the rotated matrices are denoted as $\mathbf{A}_L, \mathbf{B}_L$. Then we form the core-diagonal matrices $\tilde{\mathbf{A}}_L$ and $\tilde{\mathbf{B}}_L$ as detailed in Appendix A.

Remarks. Different from the traditional coarsening methods, such as the AMG-based methods, the MMF algorithm can be viewed as coarsening the network without reducing the size of the adjacency matrix. Due to its root in multiresolution analysis on networks, MMF offers a natural way to explore the *cluster-within-clusters* structure, which we elaborate as follows. Generally speaking, the multiresolution analysis on networks (e.g., diffusion wavelet [4]) aims to find a sequence of smoother spaces of the network. Each space captures the corresponding structured representation of the network (e.g., clusters) at a certain level by its basis (i.e., scaling functions) and the scaling coefficients. Specifically, in MMF, the scaling functions of $\mathbf{A}_1$ at the l-th level are formed by the rows of $\mathbf{P}_{l-1}\cdots\mathbf{P}_1$ indexed by $\mathcal{S}_{A_l}$, and $\mathbf{A}_l(\mathcal{S}_{A_l}, \mathcal{S}_{A_l})$ acts as the scaling coefficients. In fact, each row/column of $\mathbf{A}_l(\mathcal{S}_{A_l}, \mathcal{S}_{A_l})$ can be viewed as a node represented in the corresponding smooth space (or equivalently, clusters at level l) and these coefficients measure the strength of the interactions among the clusters at level l. Similarly, at the $(l+1)$-th level, the clusters at level l tend to be further clustered and hence MMF can explore the *cluster-within-clusters* structure of the networks. Thus, $\tilde{\mathbf{A}}_L, \tilde{\mathbf{B}}_L$ capture the coarsest cluster structures of the input networks.

B - Alignment across the coarsest networks. After the coarsening step (e.g., Eq. (3) on $\mathbf{A}_1$), the symmetrically normalized adjacency matrices of the input networks are transformed into the corresponding core-diagonal matrices, i.e.,

$$\tilde{\mathbf{A}}_L = \mathbf{\Pi}_A\begin{bmatrix}\tilde{\mathbf{A}}_{L_1} & \mathbf{0}\\ \mathbf{0} & \tilde{\mathbf{A}}_{L_2}\end{bmatrix}\mathbf{\Pi}_A^T,\ \tilde{\mathbf{B}}_L = \mathbf{\Pi}_B\begin{bmatrix}\tilde{\mathbf{B}}_{L_1} & \mathbf{0}\\ \mathbf{0} & \tilde{\mathbf{B}}_{L_2}\end{bmatrix}\mathbf{\Pi}_B^T \tag{9}$$

where $\tilde{\mathbf{A}}_{L_1} = \tilde{\mathbf{A}}_L(\mathcal{S}_{A_L}, \mathcal{S}_{A_L})$ and $\tilde{\mathbf{B}}_{L_1} = \tilde{\mathbf{B}}_L(\mathcal{S}_{B_L}, \mathcal{S}_{B_L})$ are the core matrices of $\tilde{\mathbf{A}}_L$ and $\tilde{\mathbf{B}}_L$ respectively. $\mathbf{\Pi}_A, \mathbf{\Pi}_B$ are the orthogonal permutation matrices to reorder the active indices of the matrices $\tilde{\mathbf{A}}_L, \tilde{\mathbf{B}}_L$ to be in the upper left part for the illustration purpose. Denote the inactive indices as $\overline{\mathcal{S}}_{A_L} = \{1, \cdots, n_1\} \setminus \mathcal{S}_{A_L}$ and $\overline{\mathcal{S}}_{B_L} = \{1, \cdots, n_2\} \setminus \mathcal{S}_{B_L}$. Accordingly, $\tilde{\mathbf{A}}_{L_2} = \tilde{\mathbf{A}}_L(\overline{\mathcal{S}}_{A_L}, \overline{\mathcal{S}}_{A_L})$ and $\tilde{\mathbf{B}}_{L_2} = \tilde{\mathbf{B}}_L(\overline{\mathcal{S}}_{B_L}, \overline{\mathcal{S}}_{B_L})$. Note that our algorithm does not need to explicitly compute such permutation matrices.

To solve the alignment between $\tilde{\mathbf{A}}_L$ and $\tilde{\mathbf{B}}_L$ at the coarsest level, we use the iterative fixed-point algorithm similar to Eq. (2).

$$\mathbf{S}_L = \alpha\tilde{\mathbf{B}}_L\mathbf{S}_L\tilde{\mathbf{A}}_L + (1-\alpha)\mathbf{H}_L \tag{10}$$

where $\mathbf{H}_L = \mathbf{Q}_{L-1} \cdots \mathbf{Q}_1 \mathbf{H}_1 \mathbf{P}_1^T \cdots \mathbf{P}_{L-1}^T$ is the corresponding prior similarity matrix at the coarsest level. By using the permutation matrices $\mathbf{\Pi}_A, \mathbf{\Pi}_B$, Eq. (10) can be rewritten as

$$\mathbf{\Pi}_B^T \mathbf{S}_L \mathbf{\Pi}_A = \alpha(\mathbf{\Pi}_B^T \tilde{\mathbf{B}}_L \mathbf{\Pi}_B)(\mathbf{\Pi}_B^T \mathbf{S}_L \mathbf{\Pi}_A)(\mathbf{\Pi}_A^T \tilde{\mathbf{A}}_L \mathbf{\Pi}_A^T) + (1-\alpha)\mathbf{\Pi}_B^T \mathbf{H}_L \mathbf{\Pi}_A$$

By denoting $\tilde{\mathbf{S}}_L = \mathbf{\Pi}_B^T \mathbf{S}_L \mathbf{\Pi}_A$ and $\tilde{\mathbf{H}}_L = \mathbf{\Pi}_B^T \mathbf{H}_L \mathbf{\Pi}_A$, the computation can be simplified to

$$\tilde{\mathbf{S}}_L = \alpha \begin{bmatrix} \tilde{\mathbf{B}}_{L_1} & \mathbf{0} \\ \mathbf{0} & \tilde{\mathbf{B}}_{L_2} \end{bmatrix} \begin{bmatrix} \tilde{\mathbf{S}}_{L_1} & \tilde{\mathbf{S}}_{L_2} \\ \tilde{\mathbf{S}}_{L_3} & \tilde{\mathbf{S}}_{L_4} \end{bmatrix} \begin{bmatrix} \tilde{\mathbf{A}}_{L_1} & \mathbf{0} \\ \mathbf{0} & \tilde{\mathbf{A}}_{L_2} \end{bmatrix} + (1-\alpha) \begin{bmatrix} \tilde{\mathbf{H}}_{L_1} & \tilde{\mathbf{H}}_{L_2} \\ \tilde{\mathbf{H}}_{L_3} & \tilde{\mathbf{H}}_{L_4} \end{bmatrix}$$

which allows the computation to be block-wise as follows:

$$\tilde{\mathbf{S}}_{L_1} = \alpha \tilde{\mathbf{B}}_{L_1} \tilde{\mathbf{S}}_{L_1} \tilde{\mathbf{A}}_{L_1} + (1-\alpha)\tilde{\mathbf{H}}_{L_1} \tag{11}$$

$$\tilde{\mathbf{S}}_{L_2} = \alpha \tilde{\mathbf{B}}_{L_1} \tilde{\mathbf{S}}_{L_2} \tilde{\mathbf{A}}_{L_2} + (1-\alpha)\tilde{\mathbf{H}}_{L_2} \tag{12}$$

$$\tilde{\mathbf{S}}_{L_3} = \alpha \tilde{\mathbf{B}}_{L_2} \tilde{\mathbf{S}}_{L_3} \tilde{\mathbf{A}}_{L_1} + (1-\alpha)\tilde{\mathbf{H}}_{L_3} \tag{13}$$

$$\tilde{\mathbf{S}}_{L_4} = \alpha \tilde{\mathbf{B}}_{L_2} \tilde{\mathbf{S}}_{L_4} \tilde{\mathbf{A}}_{L_2} + (1-\alpha)\tilde{\mathbf{H}}_{L_4} \tag{14}$$

Armed with the iterative fixed-point algorithm, the global optimal solutions to Eq. (11), Eq. (12) and Eq. (13) can be achieved and are denoted as $\tilde{\mathbf{S}}^*_{L_1}, \tilde{\mathbf{S}}^*_{L_2}, \tilde{\mathbf{S}}^*_{L_3}$ respectively. Furthermore, since both $\tilde{\mathbf{A}}_{L_2}$ and $\tilde{\mathbf{B}}_{L_2}$ are sparse diagonal matrices, the closed-form optimal solution of Eq. (14) can be easily computed by

$$\tilde{\mathbf{s}}^*_{L_4} = (1-\alpha)(\mathbf{I} - \alpha \tilde{\mathbf{A}}_{L_2} \otimes \tilde{\mathbf{B}}_{L_2})^{-1} \tilde{\mathbf{h}}_{L_4} \tag{15}$$

where $\tilde{\mathbf{s}}^*_{L_4} = \text{vec}(\tilde{\mathbf{S}}^*_{L_4}), \tilde{\mathbf{h}}_{L_4} = \text{vec}(\tilde{\mathbf{H}}_{L_4})$ and the operator $\otimes$ represents the Kronecker product. In this way, by the permutation matrices $\mathbf{\Pi}_A, \mathbf{\Pi}_B$, the optimal solution to the alignment problem at the coarsest level $\mathbf{S}^*_L$ is composed of $\mathbf{S}^*_L(\mathcal{S}_{B_L}, \mathcal{S}_{B_L})$, $\mathbf{S}^*_L(\mathcal{S}_{B_L}, \overline{\mathcal{S}}_{A_L})$, $\mathbf{S}^*_L(\overline{\mathcal{S}}_{B_L}, \mathcal{S}_{A_L})$ and $\mathbf{S}^*_L(\overline{\mathcal{S}}_{B_L}, \overline{\mathcal{S}}_{A_L})$ where

$$\begin{aligned} \mathbf{S}^*_L(\mathcal{S}_{B_L}, \mathcal{S}_{A_L}) &= \tilde{\mathbf{S}}^*_{L_1}, & \mathbf{S}^*_L(\mathcal{S}_{B_L}, \overline{\mathcal{S}}_{A_L}) &= \tilde{\mathbf{S}}^*_{L_2} \\ \mathbf{S}^*_L(\overline{\mathcal{S}}_{B_L}, \mathcal{S}_{A_L}) &= \tilde{\mathbf{S}}^*_{L_3}, & \mathbf{S}^*_L(\overline{\mathcal{S}}_{B_L}, \overline{\mathcal{S}}_{A_L}) &= \tilde{\mathbf{S}}^*_{L_4} \end{aligned} \tag{16}$$

Note that though the core matrices $\tilde{\mathbf{A}}_{L_1}, \tilde{\mathbf{B}}_{L_1}$ could be dense, their sizes $\lambda_L \times \lambda_L$ and $\mu_L \times \mu_L$ are very small (i.e., $\lambda_L \ll n_1, \mu_L \ll n_2$), which can be controlled by the parameter L. This is due to the fact of parallel MMF that the size of the core matrix decreases exponentially (e.g., $\lambda_L \approx 2^{-L} n_1$). Empirically, we find that λ_L, μ_L in the order of hundreds often achieves a good balance between effectiveness and efficiency, as we will show in Section 4.. Moreover, thanks to the fact that both $\tilde{\mathbf{A}}_{L_2}, \tilde{\mathbf{B}}_{L_2}$ are diagonal, the above computations are very efficient as we will show in the next subsection.

C - Interpolation. After the optimal alignment matrix $\mathbf{S}^*_L$ at the coarsest level is achieved, the alignment at each level $\mathbf{S}^*_l$ can be computed by the interpolation, i.e., $\mathbf{S}^*_l = \mathbf{Q}_l^T \mathbf{S}^*_{l+1} \mathbf{P}_l$, $l = 1, \cdots, L-1$.

As shown in the next subsection, the number of nonzero elements of $\mathbf{S}^*_L$ is $O(L^2 m_H + n d_L)$, making the computations of interpolations possibly intense. To address this issue, we only preserve the top-K elements of each row/column based on their absolute values in $\mathbf{S}^*_L$. The entire algorithm is summarized in Algorithm 1. We use two criteria to terminate the iteration (line 7-9): (1) the change of $\tilde{\mathbf{S}}_{L_1}, \tilde{\mathbf{S}}_{L_2}, \tilde{\mathbf{S}}_{L_3}$ of two successive iterations is less than a threshold ξ, or (2) the maximum number of iterations $t_{\max}$ is reached.

Recall that the coarse-scale structures of the networks are captured by the core matrices (e.g., $\mathbf{A}_2(\mathcal{S}_{A_2}, \mathcal{S}_{A_2})$ and $\mathbf{B}_2(\mathcal{S}_{B_2}, \mathcal{S}_{B_2})$), the corresponding submatrix of the alignment matrix indicates how network clusters are aligned. For example, $\mathbf{S}^*_2(\mathcal{S}_{B_2}, \mathcal{S}_{A_2})$ indicates how clusters of network $\mathcal{G}_1$ and $\mathcal{G}_2$ are aligned at the second level.

Algorithm 1 Multilevel Network Alignment (Moana).

Input: (1) the adjacency matrices $\bar{\mathbf{A}}_1, \bar{\mathbf{B}}_1$ of two undirected networks $\mathcal{G}_1$, $\mathcal{G}_2$, (2) the sparse prior alignment preference $\mathbf{H}_1$, (3) the number of levels L, (4) the parameters α, K.

Output: the alignment matrices $\mathbf{S}^*_l$, $l = 1, \cdots, L$ between $\mathcal{G}_1, \mathcal{G}_2$.

1: Compute $\mathbf{A}_1, \mathbf{B}_1$ by symmetrically normalizing $\bar{\mathbf{A}}_1, \bar{\mathbf{B}}_1$;

▷ **Network coarsening.**

3: $\mathbf{P}_1, \cdots, \mathbf{P}_{L-1}$ and $\tilde{\mathbf{A}}_L \leftarrow \text{MMF}(\mathbf{A}_1)$;

4: $\mathbf{Q}_1, \cdots, \mathbf{Q}_{L-1}$ and $\tilde{\mathbf{B}}_L \leftarrow \text{MMF}(\mathbf{B}_1)$;

5: Compute the coarsest level $\mathbf{H}_L = \mathbf{Q}_{L-1} \cdots \mathbf{Q}_1 \mathbf{H}_1 \mathbf{P}_1^T \cdots \mathbf{P}_{L-1}^T$;

▷ **Alignment at the coarsest level.**

7: **while** not converged **do**

8: Update $\tilde{\mathbf{S}}_{L_1}, \tilde{\mathbf{S}}_{L_2}, \tilde{\mathbf{S}}_{L_3}$ by Eq. (11)-(13);

9: Compute $\tilde{\mathbf{s}}^*_{L_4}$ by Eq. (15) and $\tilde{\mathbf{S}}^*_{L_4} = \text{mat}(\tilde{\mathbf{s}}^*_{L_4})$;

10: Compose $\mathbf{S}^*_L$ by Eq. (16);

▷ **Alignment interpolation.**

12: Preserve top-K elements in each row/column of $\mathbf{S}^*_L$;

13: **for** $l = L-1 \rightarrow 1$ **do**

14: Compute $\mathbf{S}^*_l = \mathbf{Q}_l^T \mathbf{S}^*_{l+1} \mathbf{P}_l$;

3.4 Algorithm Analysis

We analyze the effectiveness and efficiency of our proposed algorithm. The *perfect interpolation* (Lemma 1) indicates that in theory, by using orthogonal interpolation matrices, the alignment matrix at the finer level can be obtained from that at the coarser level without any interpolation error. Nonetheless, in practice, we might still introduce some alignment error at the coarsest level due to the fact that many matrices are not *fully multiresolution factorizable* (i.e., $\tilde{\mathbf{A}}_L \neq \mathbf{A}_L$) [15]. Having this in mind, we seek to quantify the alignment error due to the fact that $\mathbf{A}_L \neq \tilde{\mathbf{A}}_L, \mathbf{B}_L \neq \tilde{\mathbf{B}}_L$.

Theorem 1. ***Alignment error bound.*** *Denote $\delta_1 = \|\mathbf{A}_L - \tilde{\mathbf{A}}_L\|_F$, $\delta_2 = \|\mathbf{B}_L - \tilde{\mathbf{B}}_L\|_F$. Suppose the ranks of $\mathbf{A}_1, \mathbf{B}_1$ are r_1, r_2 whose values are often small due to the low-rank characteristics of many real-world networks. Then, the error of the output alignment matrix at level $l, l = 1, \cdots, L$ is bounded as*

$$\frac{\|\mathbf{S}^*_l - \mathbf{S}_l\|_F}{\|\mathbf{S}_l\|_F} \leq \frac{2\epsilon\kappa}{1-\epsilon\kappa} \tag{17}$$

where $\mathbf{S}^*_l$ is the alignment matrix at level l obtained by Algorithm 1, $\mathbf{S}_l$ is the ideally interpolated alignment from $\mathbf{S}_L$ which is computed by $\mathbf{A}_L$. The parameter κ is the condition number under Frobenius norm of the matrix $(1-\alpha)(\mathbf{I} - \alpha \mathbf{A}_1 \otimes \mathbf{B}_1)$, and $\epsilon = \sqrt{\frac{\alpha}{2n}}(\delta_1 r_2 + \delta_2 r_1 + \delta_1 \delta_2)$.

Proof. See Appendix B. □

For the efficiency analysis of Moana, we start with the Lemma 2 on the sparsity of the matrix $\tilde{\mathbf{H}}_{L_4}$ which paves the way to the complexity analysis.

Lemma 2. ***Sparsity of $\tilde{\mathbf{H}}_{L_4}$.*** *Suppose there are m_H nonzero elements in the sparse prior similarity matrix $\mathbf{H}_1$ and $m_H \ll n^2$. At the coarsest level, the number of nonzero elements of $\tilde{\mathbf{H}}_{L_4}$ is $O(L^2 m_H)$.*

Proof. Denote the density of the nonzero elements in $\mathbf{H}_1$ as $\rho_1 = \frac{m_H}{n^2}$. Starting with the coarsening at the first level, the computation

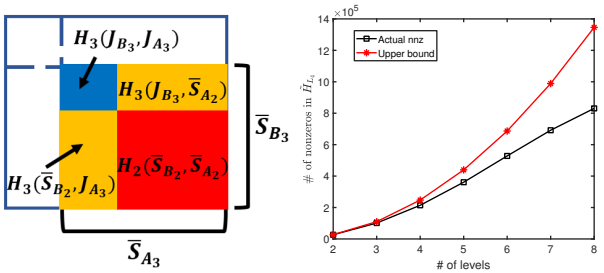


Figure 2: An illustration example of Lemma 2. (a) explains its proof where the indices have been reordered. (b) shows a case on Gr-Qc networks where $\rho_1 = 0.001, n = 5,241$.

of the matrix $\mathbf{H}_2$ is as follows.

$$\mathbf{H}_2(i,j) = (\mathbf{Q}_1\mathbf{H}_1\mathbf{P}_1^T)(i,j) = \sum_{p,q} \mathbf{Q}_1(i,q)\mathbf{H}_1(q,p)\mathbf{P}_1(j,p) \quad (18)$$

$$= \mathbf{Q}_1(i,i)\mathbf{H}_1(i,j)\mathbf{P}_1(j,j) + \mathbf{Q}_1(i,i)\mathbf{H}_1(i,p)\mathbf{P}_1(j,p) \quad (19)$$
$$+ \mathbf{Q}_1(i,q)\mathbf{H}_1(q,j)\mathbf{P}_1(j,j) + \mathbf{Q}_1(i,q)\mathbf{H}_1(q,p)\mathbf{P}_1(j,p)$$

The second equation is due to Eq. (25) where $k = 2$. That is, $\mathbf{Q}_1(i,q) \neq 0$ if $i_1^u = i$ and $i_2^u = q$ for some u, and similarly to $\mathbf{P}_1(j,p)$. Thus, $\mathbf{H}_2(i,j)$ can be nonzero if at least one of the $\mathbf{H}_1(i,j)$, $\mathbf{H}_1(i,p)$, $\mathbf{H}_1(q,j)$, $\mathbf{H}_1(q,p)$ are nonzero. In other words, each nonzero element in $\mathbf{H}_1$ can contribute to generating at most four nonzero elements in $\mathbf{H}_2$. Thus, the density of $\mathbf{H}_2$ is $\rho_2 \leq 4\rho_1$. Denote $\mathcal{S}_{A_l}, \mathcal{S}_{B_l}$ as the active indices of $\mathbf{A}_l, \mathbf{B}_l$ at the l-th level, and $\overline{\mathcal{S}}_{A_l}, \overline{\mathcal{S}}_{B_l}$ as the inactive indices. Since the second order MMF inactivates half of the current active indices [15], i.e., $|\overline{\mathcal{S}}_{A_2}| \approx |\overline{\mathcal{S}}_{B_2}| \approx \frac{n}{2}$, we have the number of nonzero elements $\text{nnz}(\mathbf{H}_2(\overline{\mathcal{S}}_{B_2}, \overline{\mathcal{S}}_{A_2})) \leq 4\rho_1(\frac{n}{2})^2 = \rho_1 n^2$. At the second level, thanks to the property of MMF that $\mathbf{P}_2(\overline{\mathcal{S}}_{A_2}, \overline{\mathcal{S}}_{A_2}) = \mathbf{I}$ and $\mathbf{Q}_2(\overline{\mathcal{S}}_{B_2}, \overline{\mathcal{S}}_{B_2}) = \mathbf{I}$, the computation of $\mathbf{H}_3 = \mathbf{Q}_2\mathbf{H}_2\mathbf{P}_2^T$ can be decomposed into

$$\mathbf{H}_3(\mathcal{S}_{B_2}, \mathcal{S}_{A_2}) = \mathbf{Q}_2(\mathcal{S}_{B_2}, \mathcal{S}_{B_2})\mathbf{H}_2(\mathcal{S}_{B_2}, \mathcal{S}_{A_2})\mathbf{P}_2^T(\mathcal{S}_{A_2}, \mathcal{S}_{A_2}) \quad (20)$$

$$\mathbf{H}_3(\mathcal{S}_{B_2}, \overline{\mathcal{S}}_{A_2}) = \mathbf{Q}_2(\mathcal{S}_{B_2}, \mathcal{S}_{B_2})\mathbf{H}_2(\mathcal{S}_{B_2}, \overline{\mathcal{S}}_{A_2}) \quad (21)$$

$$\mathbf{H}_3(\overline{\mathcal{S}}_{B_2}, \mathcal{S}_{A_2}) = \mathbf{H}_2(\overline{\mathcal{S}}_{B_2}, \mathcal{S}_{A_2})\mathbf{P}_2^T(\mathcal{S}_{A_2}, \mathcal{S}_{A_2}) \quad (22)$$

$$\mathbf{H}_3(\overline{\mathcal{S}}_{B_2}, \overline{\mathcal{S}}_{A_2}) = \mathbf{H}_2(\overline{\mathcal{S}}_{B_2}, \overline{\mathcal{S}}_{A_2}) \quad (23)$$

Similar to the analysis of Eq. (18), the density of the submatrix $\mathbf{H}_3(\mathcal{S}_{B_2}, \mathcal{S}_{A_2})$ is further increased by at most four times, and that of $\mathbf{H}_3(\mathcal{S}_{B_2}, \overline{\mathcal{S}}_{A_2}), \mathbf{H}_3(\overline{\mathcal{S}}_{B_2}, \mathcal{S}_{A_2})$ will be increased by at most twice. Note that the indices $\mathcal{J}_{A_3} = \mathcal{S}_{A_2} \setminus \mathcal{S}_{A_3}$ and $\mathcal{J}_{B_3} = \mathcal{S}_{B_2} \setminus \mathcal{S}_{B_3}$ are added into the inactive indices at the third level where $|\mathcal{J}_{A_3}| \approx |\mathcal{J}_{B_3}| \approx \frac{n}{4}$. And since the matrix $\mathbf{H}_3(\overline{\mathcal{S}}_{B_3}, \overline{\mathcal{S}}_{A_3})$ consists of four small blocks, including $\mathbf{H}_3(\mathcal{J}_{B_3}, \mathcal{J}_{A_3}), \mathbf{H}_3(\mathcal{J}_{B_3}, \overline{\mathcal{S}}_{A_2}), \mathbf{H}_3(\overline{\mathcal{S}}_{B_2}, \mathcal{J}_{A_3})$ and $\mathbf{H}_2(\overline{\mathcal{S}}_{B_2}, \overline{\mathcal{S}}_{A_2})$, the number of its nonzero elements is

$$\text{nnz}(\mathbf{H}_3(\overline{\mathcal{S}}_{B_3}, \overline{\mathcal{S}}_{A_3})) \leq 16\rho_1(\frac{n}{4})^2 + 2(\frac{n^2}{8})8\rho_1 + \rho_1 n^2 = 4\rho_1 n^2$$

By induction, at the L-th level, the total number of nonzero elements of $\mathbf{H}_L(\overline{\mathcal{S}}_{B_L}, \overline{\mathcal{S}}_{A_L})$ is

$$\text{nnz}(\mathbf{H}_L(\overline{\mathcal{S}}_{B_L}, \overline{\mathcal{S}}_{A_L})) \leq \sum_{l=2}^{L}(2l-3)\rho_1 n^2 \leq (L-1)^2 m_H \quad (24)$$

Thus, $\text{nnz}(\tilde{\mathbf{H}}_{L_4}) = \text{nnz}(\mathbf{H}_L(\overline{\mathcal{S}}_{B_L}, \overline{\mathcal{S}}_{A_L})) = O(L^2 m_H)$. □

We present an illustrative example of Lemma 2 in Figure 2, where Figure 2 (a) shows different parts of matrix $\mathbf{H}_L$ after some index reordering for the sake of illustration, and Figure 2 (b) gives a real case on the Gr-Qc networks where $\rho_1 = 0.001$, $n = 5,241$. We can observe that the actual number of nonzero elements in $\tilde{H}_{L_4}$ is much less than the analyzed upper bound, i.e., $(L-1)^2 m_H$, as L increases. Next, based on the Lemma 2, we give the complexity of the proposed Moana algorithm in Theorem 2, which states that the computational complexity of our proposed algorithm is *linear*.

Theorem 2. ***Complexity analysis.*** *The time complexity of Algorithm 1 is $O(mL + nd_L^2 t_{max} + L^2 m_H + LKn)$ and its space complexity is $O(L^2 m_H + L^2 Kn + nd_L)$. Here, m, n are the number of edges and nodes in the networks, $d_L = max(\lambda_L, \mu_L)$ is the size of core matrix. t_{max} is the number of iterations until convergence in the alignment phase and K is used for top-K preservation of $\mathbf{S}_L^*$. m_H is the number of nonzero elements in the matrix $\mathbf{H}_1$ and L is the number of levels.*

Proof. In Algorithm 1, the symmetric normalization of $\bar{\mathbf{A}}_1, \bar{\mathbf{B}}_1$ in Line 1 requires an $O(m)$ time complexity. In the network coarsening phase, the MMF on $\mathbf{A}_1, \mathbf{B}_1$ has an $O(mL)$ time and space complexity[1] where by a careful implementation, we use Graclus [6] for an extremely efficient clustering and avoid explicitly computing the Gram matrix in MMF. Note that we can compute $\mathbf{H}_l$ ($l = 2, \cdots, L$) in a same way as Eq. (20)-(23). Thus, by Lemma 2, the computation of the matrix $\mathbf{H}_L$ takes $O(Lm_H)$ as time complexity and $O(L^2 m_H + nd_L)$ as space complexity.

In the alignment phase, Line 8 has an $O(d_L^3)$ time complexity for Eq. (11), as well as $O(nd_L^2)$ for both Eq. (12) and Eq. (13). To compute Line 10, since there are $O(L^2 m_H)$ nonzero elements in $\tilde{h}_L$, the time complexity is $O(L^2 m_H)$. The space complexity to store the alignment matrix $\mathbf{S}_L^*$ is $O(L^2 m_H + nd_L)$ due to the nonzero elements.

In the interpolation phase, Line 13 costs $O(Kn)$ in both time and space. The analysis of Line 15 is analogous to that of Line 5 so Line 15 has an $O(LKn)$ time complexity and an $O(L^2 Kn)$ space complexity. Thus, the total time complexity is $O(m + nd_L^2 t_{\max} + L^2 m_H + LKn)$ and the space complexity is $O(L^2 m_H + L^2 Kn + nd_L)$. □

4 EXPERIMENTS

In this section, we present the experimental results of the proposed algorithm Moana. We evaluate it in the following aspects:

- *Effectiveness*: How accurate is our algorithm to align networks and how robust is our algorithm to the parameters?
- *Efficiency*: How fast and scalable is our algorithm?

4.1 Experimental Setup

Datasets. We evaluate the proposed algorithm on six real-world networks. The statistics of all datasets are summarized as follows.

- *Zachary's Karate Club:* This dataset contains 34 nodes and 78 edges. Each node is a member of the karate club and each edge represents the friendship between two members [33].
- *Gr-Qc network:* This collaboration network contains 5,241 nodes and 11,923 edges. Each node represents an author, and

[1] In [29], the complexity for a sparse matrix is given in the form of $O(\gamma c L n^2)$ where c is the cluster size, γ is the fraction of nonzero elements and hence $O(\gamma n^2) = O(m)$.

there exists an edge if two authors have coauthored together [17].

- *Google+ network:* This dataset contains 23,628 nodes and 39,194 edges. Nodes are the users of Google+ and an edge denotes that one user has the other user in his/her circles [18].
- *Amazon product co-purchasing network:* This network was collected from the Amazon website. Different nodes represent different products. If a product is frequently co-purchased with another product, there exists an edge between them. In total, there are 334,863 nodes and 925,872 edges in the network [32].
- *ACM coauthor network:* The ACM dataset is collected up to 2016 and it contains 2,381,688 papers. Each paper has a list of authors as well as the venue of the paper [28]. We construct a coauthorship network from the dataset where each node represents an author and each edge represents the coauthorship.
- *DBLP coauthor network:* The DBLP dataset is collected up to 2016 and it contains 3,272,991 papers [28]. Same as the ACM dataset, we construct an coauthorship network from the dataset.

Based on the above datasets, we construct the following five alignment scenarios for evaluations.

- *S1. Zachary vs. Zachary networks.* Given the adjacency matrix $\bar{\mathbf{A}}_1$ of the original network, we generate a random permutation matrix $\mathbf{P}$ and treat the permuted matrix $\bar{\mathbf{B}}_1 = \mathbf{P}\bar{\mathbf{A}}_1\mathbf{P}^T$ as the adjacency matrix of the second network. The permutation matrix $\mathbf{P}$ is used as the ground-truth alignment. We construct the prior similarity matrix $\mathbf{H}_1$ based on the node degree similarity.
- *S2. Gr-Qc vs. Gr-Qc networks.* The scenario is built same as *S1*.
- *S3. Google+ vs. Google+ networks.* This is built same as *S1*.
- *S4. Amazon co-purchasing vs. Amazon co-purchasing networks.* We randomly extract two subgraphs to be aligned from the entire network. The first subgraph has 74,596 nodes and 196,534 edges while the second subgraph has 66,951 nodes and 174,467 edges. There are 62,406 common nodes which are used as the ground-truth. We use node degree similarity as $\mathbf{H}_1$.
- *S5. ACM vs. DBLP coauthor networks.* We extract a subgraph from each coauthorship network. Specifically, the extracted ACM subgraph has 9,872 nodes and 39,561 edges, while the DBLP subgraph has 9,916 nodes and 44,808 edges. There are 6,325 common nodes as the ground-truth. We use the number of papers in different venues published by each author as the node attributes. We calculate $\mathbf{H}_1$ by the cosine similarity of the node attributes.

Implementation Details. As analyzed in Theorem 1, the zero-out operation in MMF implementation could introduce a certain alignment error at the coarsest level. To mitigate this issue, we run one iteration of the fixed-point update as the post-processing step: $\mathbf{S}_l^* = \alpha\mathbf{B}_l\mathbf{S}_l^*\mathbf{A}_l + (1-\alpha)\mathbf{H}_l$. This computation can be done in near-linear time with a locality sensitive hashing based implementation, thanks to the sparsity of the matrices. To evaluate the effectiveness of our proposed algorithm, we use the greedy match algorithm [27] as a post-processing to obtain the one-to-one mapping from the alignment matrix, followed by calculating the percentage of the ground-truth that can be correctly aligned as the accuracy.

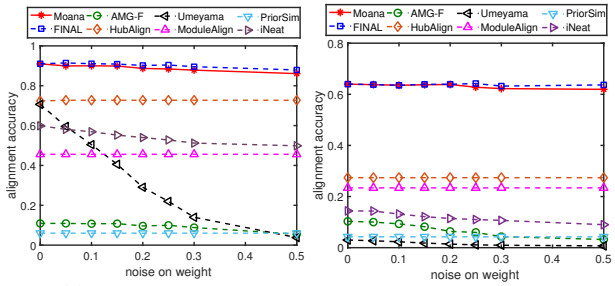
Comparison Methods. We compare our algorithm Moana with the following existing network alignment algorithms, including (1) *FINAL* that uses the fixed-point algorithm on Eq. (4) [35], (2) *AMG-F* that first uses *AMG* [23] to coarsen the input networks at L levels and then aligns based on the coarsest networks, followed by interpolating the alignment matrix with the interpolation matrices generated by *AMG*, (3) *iNeat* [36], (4) *HubAlign* [11], (5) *Umeyama* [30], (6) *ModuleAlign* [10] that leverages the hierarchical clustering on the networks and (7) *PriorSim* which aligns based on the prior similarity matrix $\mathbf{H}_1$. Note that *iNeat* was originally designed to learn the nonnegative matrix factorization of the incomplete adjacency matrices. In our experiments, we use the low-rank eigen-decomposition of the input adjacency matrices since there are no missing entries in our setting.

Machines and Repeatability. All experiments are performed with four 3.6GHz Intel Cores and 256G RAM. Note that a large size of memory is only needed for running some comparison methods on large datasets (e.g., *Umayama* and *ModuleAlign* on *S4*). Our proposed algorithm is programmed in MATLAB. We will release the source code and the datasets after the paper is published.

4.2 Effectiveness Results

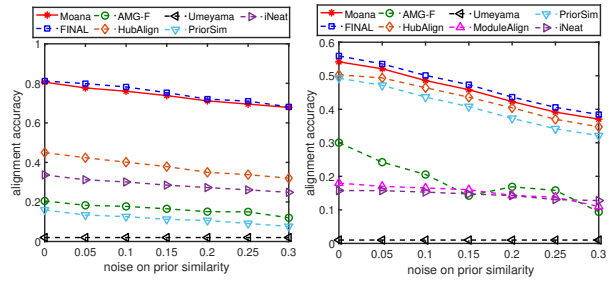
We first evaluate how the uniform noise on the edge weights influences the node-level alignment accuracy in scenarios *S2* and *S3*. The results are summarized in Figure 3. We have the following observations. First, our proposed algorithm Moana is very close to its single node-level alignment counterpart *FINAL*. Note that our algorithm can be viewed as an approximation of *FINAL* in terms of the node-level alignment. Specifically, our algorithm is only about 1.5% lower than *FINAL* in the worst case. In the meanwhile, the proposed Moana outperforms all the remaining methods that we compared against with respect to the node-level alignment accuracy. Second, our proposed algorithm, as well as *FINAL* are quite robust to the noise. This is due to the prior similarity matrix that works as the regularization and can mitigate the effects of noise on the networks. For *HubAlign* and *ModuleAlign*, since they are originally designed solely for unweighted networks, we replicate their results without any noise and therefore their curves are flat. On the other side, *Umeyama* is more sensitive to the noise. Third, our multilevel alignment algorithm achieves an accuracy improvement by up to 30%, compared with *HubAlign*, *ModuleAlign* and *iNeat* that are designed to leverage some special structural characteristics (e.g., hub nodes, hierarchical clustering, low rank) of the underlying networks. This improvement demonstrates the potential advantages in finding node-level alignment of our multilevel alignment method by exploring the richer patterns of networks.

Furthermore, we evaluate the impact on the alignment accuracy by the noise in the input prior similarity matrix $\mathbf{H}_1$ in the scenarios *S4* and *S5*. As one can see in the Figure 4, although most of the comparison methods have an accuracy drop due to the noise, our proposed algorithm still remains very close to *FINAL*. The results of *ModuleAlign* in the Amazon Product co-purchasing networks are not included because the algorithm cannot finish in a reasonable



(a) Cr-Qc networks. (b) Google+ networks.

Figure 3: (Higher is better.) Alignment accuracy vs. the edge weight noise. (a) L=5, K=500, α=0.5. (b) L=7, K=1500, α=0.5.



(a) Amazon co-purchasing networks (L=8, K=2000, α=0.5). (b) ACM vs. DBLP coauthor networks (L=5, K=1000, α=0.5).

Figure 4: (Higher is better.) Alignment accuracy vs. the noise in the prior similarity matrix.

time on large networks (i.e., about 70k nodes) given its high time complexity $O(n^3 \log n)$ [10].

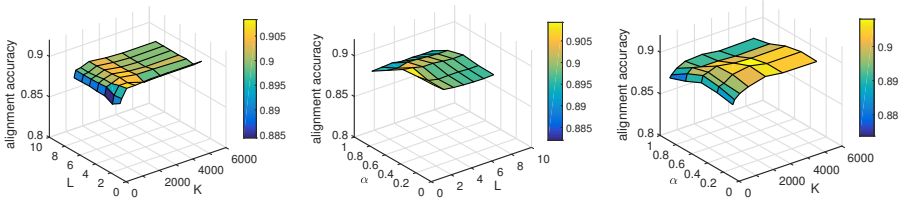
Parametric study on node-level alignment. There are three parameters K, L, α in the proposed algorithm. To be specific, α is the regularization parameter that controls the importance of the topology consistency, K is used to preserve top-K elements in $\mathbf{S}_L^*$, and L is the number of levels. We fix one of these parameters and study the impact of the other two parameters on the node-level alignment accuracy. The results are summarized in Figure 5. We have two main observations. First, the alignment accuracy is stable over a wide range of α, K and L. In particular, the performance first increases with respect to K, and it quickly becomes flat for $K \geq 500$. Notice that a small K can also accelerate the algorithm, and hence $K = 500$ is recommended in practice. On the other hand, recall that increasing L reduces the size of core matrices (i.e., λ_L, μ_L) and hence improves the efficiency of the alignment at the coarsest level, at a small expense in the alignment accuracy and the running time of MMF itself. We find that an L that leads to the size of core matrices (i.e., λ_L, μ_L) in the order of hundreds achieves a good balance between effectiveness and efficiency. Second, compared with the other two parameters (K, L), the regularization parameter α has a relatively larger impact on the alignment accuracy. This is because α balances the importance of the topology consistency and the prior alignment information in Eq. (1), and it directly influences the exact solution of its single node-level alignment.

The multilevel nature of Moana brings the potential to align clusters across different networks. To verify this, we further conduct an experiment to evaluate the cluster alignment accuracy on Gr-Qc networks. In particular, we view the nodes that are integrated into a single 'supernode' up to the l-th level (e.g., based on $\mathbf{P}_1, \cdots, \mathbf{P}_{l-1}$) as a cluster at level l. If a cluster at level l in network $\mathcal{G}_1$ share the most overlapped nodes with some cluster in $\mathcal{G}_2$, we treat the alignment between these two clusters as a ground-truth. With the output $\mathbf{S}_l^*(\mathcal{S}_{B_l}, \mathcal{S}_{A_l})$ matrices by Algorithm 1, we define the top-15% cluster alignment accuracy as follows. Given a ground-truth alignment between cluster i in $\mathcal{G}_1$ and cluster j in $\mathcal{G}_2$, if $\mathbf{S}_l^*(j, i)$ is among the highest top-15% entries within the j-th row and i-th column, we say there is a *hit*, i.e., the two clusters are correctly aligned. As Figure 6 shows, our algorithm can achieve a good cluster alignment accuracy at different levels.

A case study on the multilevel alignment. In addition to the quantitative comparisons, we also conduct a case study on the Zachary's Karate Club networks (*S1*) to further demonstrate our algorithm's capability to find the alignment among clusters at different granularities. We only show sample results in Figure 7 for clarity. Among others, Figure 7 (a) visualizes the input networks. Figure 7 (b) and (c) show the network structures at the two coarsest levels, respectively. To be specific, the nodes with a larger marker size represent the active indices of the coarsened matrices. Each of them represents a certain pattern (e.g., cluster) that is differentiated by the shape/color of the nodes. For example, in Figure 7 (b), the blue circle node with a larger size is one of the active nodes whereas the small blue nodes are inactivated during the MMF coarsening. As one can see, all these blue circle nodes form a cluster represented by the corresponding active node, and hence each network has three clusters in total in Figure 7 (b). The weighted interactions among the clusters are computed as the submatrix indexed by the active indices, and are represented by the thick solid lines. At the coarsest level (Figure 7 (c)), the clusters and their interactions are visualized similarly. In Figure 7 (d), the orange dashed lines show the alignment results among the active nodes at the coarsest level. In fact, since the active nodes represent different clusters, the correct alignment among the active nodes indicates that the clusters at the coarsest level are aligned correctly. Once the alignment at the coarsest level is done, we interpolate the results to the next finer level shown in Figure 7 (e). Likewise, we observe that the alignment after the first interpolation can unveil the alignment among the clusters at the finer level. Figure 7 (f) only shows parts of the interpolated node-level alignment for the sake of clarity. Overall, our algorithm can correctly align 32 out of 34 node pairs. This case study demonstrates that our algorithm can not only find the correct node-level alignment, but also unveil the meaningful alignment of the clusters in the networks at different granularities.

4.3 Efficiency Results

We first evaluate the efficiency of our algorithm (Moana) in terms of the balance between the running time and the node-level alignment accuracy. The results are summarized in Figure 8. As one can see, our algorithm obtains an up to 10× speedup compared with its single-level counterpart *FINAL* with a little loss in terms of the node-level alignment accuracy. For other alignment algorithms that explore the special characteristics of the networks (*HubAlign*, *ModuleAlign* and *iNeat*), our algorithm consistently outperforms them in terms of both efficiency and effectiveness, with the only exception of *iNeat* in Figure 8 (b) which runs faster at the cost of a much lower accuracy. Overall, we conclude from these results that



(a) Impact of K and L (fixing $\alpha = 0.5$). (b) Impact of α and L (fixing $K = 500$). (c) Impact of α and K (fixing $L = 5$).

Figure 5: Parametric studies on Cr-Qc networks.

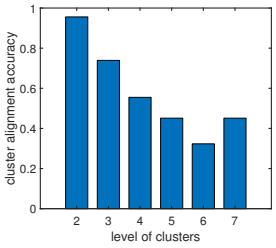


Figure 6: Cluster alignment accuracy at multiple levels.

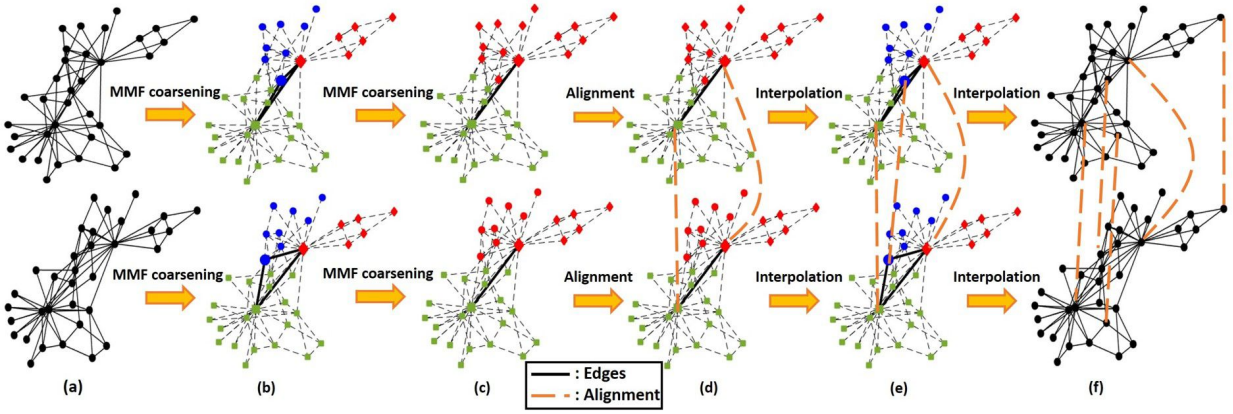
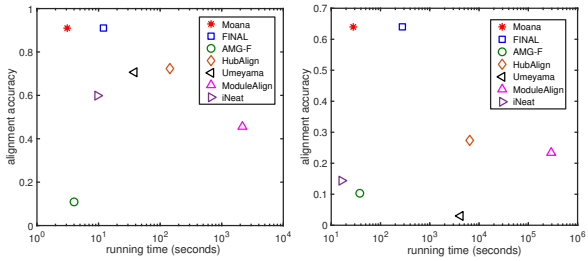
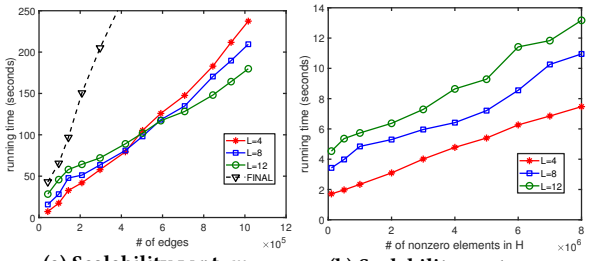


Figure 7: A case study on the Zachary's Karate Club networks (Best viewed in color).



(a) Gr-Qc networks. (b) Google+ networks.

Figure 8: Balance between the accuracy and running time.



(a) Scalability w.r.t. m. (b) Scalability w.r.t. m_H.

Figure 9: Scalability of Moana.

the proposed Moana algorithm achieves a good balance between the running time and the alignment accuracy.

Scalability. We evaluate the scalability of our algorithm (Moana) based on the Amazon product co-purchasing networks w.r.t. (1) the number of edges while fixing $m_H = 3 \times 10^5$, and (2) m_H while fixing the size of networks $n = 5,000$. The results are summarized in Figure 9. We observe that the algorithm scales *linearly* w.r.t. both the number of edges in the networks and the number of nonzero elements in the prior similarity matrix $\mathbf{H}_1$, which is consistent with Theorem 2. *FINAL* which can be viewed as a special case when $L = 1$ (i.e., without coarsening), on the other hand, scales quadratically and is slower than the proposed algorithm, especially when the networks are large. We also observe from Figure 9 (a) that large L is beneficial in terms of efficiency as the size of networks becomes large. This is because coarsening the networks at more levels can further reduce the size of the core diagonal matrices.

5 RELATED WORK

Network Alignment. Multiple networks naturally appear in many real-world applications, including network alignment [35], subgraph matching [8], multi-network ranking [12] and so on. Among others, network alignment has attracted extensive research interests. Network alignment can be generally categorized into local network alignment and global network alignment. Local network alignment aims to align the small regions (e.g., motifs, small subgraphs, etc.) across multiple networks. Some recent works in this category include [3, 22]. However, the local alignment methods might be too restrictive to find the one-to-one node mappings and the alignment among more complicated large patterns of the networks. On the other side, many existing global network alignment algorithms explicitly or implicitly assume the *topology consistency*. That is, if two nodes are aligned together, their corresponding neighbors are likely to be aligned. For example, an early work *IsoRank* conducts a random walk in the Kronecker product graph and propagates the node pair similarities to the neighboring node pairs so that the similarities are smoothed [27]. *NetAlign* formulates the network alignment problem as an optimization problem and maximizes the number of neighboring node pairs that are aligned [1]. *BigAlign*

(as well as its variant *UniAlign*) [16] and *UMA* [34] assumes one network is a noisy permutation of the other network. Moreover, *IONE* attempts to leverage the network embedding for alignment [19] in a semi-supervised manner, under the assumption that the aligned nodes are close to each other in the embedding space. To further incorporate the attribute information, *COSNET* formulates the local consistency among the node attributes and the global topological consistency into a joint optimization problem [37]. *FINAL* [35] considers both node and edge attributes to calibrate the topology-based alignment. [7] further improves the efficiency by accelerating solving the Sylvester equation.

More recently, several network alignment methods attempt to exploit some special characteristics of real-world networks. For example, *HubAlign* aligns the hub nodes first and then all the remaining nodes [11]. In a different thread, *ModuleAlign* [10] and *CAlign* [2] leverage the cluster/community structures in the networks. However, none of these methods allows to find the alignment at more than two levels. Thanks to the low rank characteristics of many real networks, a recent work *iNeat* exploits the low rank structure of the alignment matrix, leading to a provable linear algorithm [36]. The alignment matrices in *iNeat* [36] and another early work *Umeyama* [30] are represented by the multiplication of three matrices, which seems to resemble the interpolation formula in our method. However, neither *iNeat* [36] or *Umeyama* [30] is able to find the alignment at the coarse levels, since they are both essentially single-level alignment methods.

Multilevel Approaches. The multilevel approaches have shown a strong performance in many graph mining problems. For example, *METIS* [13] and *Graclus* [6] are two well-known multilevel approaches to solve the graph partitioning problem. These two methods use a heavy-edge matching based graph coarsening approach, making themselves extremely efficient. On the other side, the algebraic multigrid methods which are originally designed for solving the linear systems, have been widely used to many graph related problems, such as [21, 23]. Most of the multilevel approaches follow the *coarsen-solve-interpolate* strategy. Indeed, our proposed network alignment algorithm follows this generic strategy as well.

Another closely related concept is the multiresolution analysis on graph that has played an important role in many research areas. For example, it has been widely used to design the pooling operations in the graph convolutional neural networks [5], an emerging area in machine learning and data mining. Besides, it has been applied to design the multiscale graph kernel as well [14]. In graph signal processing, the multiresolution analysis is extensively studied by many different methods, such as the graph diffusion wavelet [4], multiscale pyramid transform [26]. Moreover, the multiresolution matrix factorization, as an alternate to low rank characteristics [40], is able to factorize the input matrix at different scales [15].

6 CONCLUSION

Large multi-sourced networks from many application domains have greatly galvanized network alignment research. Most of the existing single-level alignment methods might overlook the rich patterns (e.g., *hierarchical cluster-within-clusters* underlying the networks) and/or bear a super-linear computational complexity. In this paper, we study the multilevel network alignment problem. We first drive a theoretical condition on perfect interpolation (Lemma 1). Based on that, we propose a carefully designed *coarsening-alignment-interpolation* multilevel algorithm Moana, which has a *linear* complexity in both time and space. We also drive an error bound of the alignment due to the coarsening step. We perform extensive experiments that demonstrate the efficacy of our algorithm on the node-level alignment and its capability of finding the alignment of the rich patterns (e.g., clusters) across the input networks.

7 ACKNOWLEDGEMENT

This material is supported by the National Science Foundation under Grant No. IIS-1651203, IIS-1715385, IIS-1741197 and CNS-1639227, by DTRA under the grant number HDTRA1-16-0017, by the United States Air Force and DARPA under contract number FA8750-17-C-0153 [2], by Department of Homeland Security under Grant Award Number 2017-ST-061-QA0001, and by Army Research Office under the contract number W911NF-16-1-0168. The content of the information in this document does not necessarily reflect the position or the policy of the Government, and no official endorsement should be inferred. The U.S. Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation here on.

A COMPOUND ROTATION MATRIX

The compound rotation matrix is formally defined as follows [15].

Definition 2. ***Compound rotation matrix.*** *The compound rotation matrix* $\mathbf{P}$ *of order* k *is a block diagonal orthogonal matrix as*

$$\mathbf{P} = \oplus_{(i_1^1,\cdots,i_{k_1}^1)}\mathbf{O}^1 \oplus_{(i_1^2,\cdots,i_{k_2}^2)} \mathbf{O}^2 \oplus \cdots \oplus_{(i_1^p,\cdots,i_{k_p}^p)} \mathbf{O}^p. \quad (25)$$

where $\{i_1^1,\cdots,i_{k_1}^1\},\cdots,\{i_1^p,\cdots,i_{k_p}^p\}$ are the subsets of $\{1,\cdots,n\}$ with $k_1,\cdots,k_p \le k$ and the matrices $\mathbf{O}^1,\cdots,\mathbf{O}^p$ are some orthogonal *Givens* rotation matrices. The operator $\oplus$ is defined to form the block diagonal matrix, i.e.,

$$\mathbf{P}(a,b) = \begin{cases} \mathbf{O}^u(q,r) & \text{if } i_q^u = a \text{ and } i_r^u = b \text{ for some } u,q,r \\ 0 & \text{otherwise} \end{cases}$$

By sequentially applying the compound rotation matrices, we can obtain a series of coarsened matrices as below

$$\mathbf{A}_1 \overset{\mathbf{P}_1}{\to} \mathbf{A}_2 \overset{\mathbf{P}_2}{\to} \cdots \overset{\mathbf{P}_{L-1}}{\to} \mathbf{A}_L \to \tilde{\mathbf{A}}_L$$

where $\tilde{\mathbf{A}}_L$ is formed by zeroing out all the nonzero elements that are neither on the diagonal of $\mathbf{A}_L$ nor inside the $\mathcal{S}_L$-core matrix.

B PROOF OF THEOREM 1

Denote $\Delta\mathbf{A} = \tilde{\mathbf{A}}_L - \mathbf{A}_L$ and $\Delta\mathbf{B} = \tilde{\mathbf{B}}_L - \mathbf{B}_L$. Consider two linear systems $\mathbf{W}_L\mathbf{s}_L = \mathbf{h}_L$ and $\tilde{\mathbf{W}}_L\mathbf{s}_L^* = \mathbf{h}_L$ where $\mathbf{W}_L = (1-\alpha)(\mathbf{I}-\alpha\mathbf{A}_L\otimes\mathbf{B}_L)$, $\tilde{\mathbf{W}}_L = (1-\alpha)(\mathbf{I}-\alpha\tilde{\mathbf{A}}_L\otimes\tilde{\mathbf{B}}_L)$ and $\mathbf{s}_L = \mathrm{vec}(\mathbf{S}_L)$, $\mathbf{s}_L^* = \mathrm{vec}(\mathbf{S}_L^*)$. Clearly, the solutions to these linear systems are the closed-form solutions of alignment matrices, e.g., $\mathbf{s}_L^*$ corresponding to $\mathbf{S}_L^*$ in line 11 of Algorithm 1. By denoting $\Delta\mathbf{W}_L = \tilde{\mathbf{W}}_L - \mathbf{W}_L$, we have

$$\begin{aligned}
\|\Delta\mathbf{W}_L\|_F &= \alpha(1-\alpha)\|\mathbf{A}_L\otimes\mathbf{B}_L - \tilde{\mathbf{A}}_L\otimes\tilde{\mathbf{B}}_L\|_F \\
&= \alpha(1-\alpha)\|\Delta\mathbf{A}\otimes\mathbf{B}_L + \mathbf{A}_L\otimes\Delta\mathbf{B} + \Delta\mathbf{A}\otimes\Delta\mathbf{B}\|_F \\
&\le \alpha(1-\alpha)(\|\Delta\mathbf{A}\|_F\|\mathbf{B}_L\|_F + \|\mathbf{A}_L\|_F\|\Delta\mathbf{B}\|_F + \|\Delta\mathbf{A}\|_F\|\Delta\mathbf{B}\|_F) \\
&\le \alpha(1-\alpha)(\delta_1 r_2\|\mathbf{B}_L\|_2 + \delta_2 r_1\|\mathbf{A}_L\|_2 + \delta_1\delta_2) \\
&= \alpha(1-\alpha)(\delta_1 r_2 + \delta_2 r_1 + \delta_1\delta_2)
\end{aligned}$$

[2] Distribution Statement "A" (Approved for Public Release, Distribution Unlimited)

where the fourth line is due to $\|\mathbf{A}\|_F \leq \text{rank}(\mathbf{A})\|\mathbf{A}\|_2$ [24] for any matrix and $\text{rank}(\mathbf{A}_L) = \text{rank}(\mathbf{A}_1)$. The last line is due to the fact that $\mathbf{A}_L, \mathbf{B}_L$ have the same eigenvalues of $\mathbf{A}_1, \mathbf{B}_1$ respectively and hence the largest eigenvalues are equal to 1. Besides, we have

$$\begin{aligned}\|\mathbf{W}_L\|_F &= (1-\alpha)\sqrt{n_1 n_2 + \alpha^2 \|\mathbf{A}_L \otimes \mathbf{B}_L\|_F^2} \\ &\geq (1-\alpha)\sqrt{n^2+\alpha^2} \geq (1-\alpha)\sqrt{2\alpha n}\end{aligned}$$

where $n = \min(n_1, n_2)$. Thus, by picking $\epsilon = \sqrt{\alpha}\frac{\delta_1 r_2 + \delta_2 r_1 + \delta_1\delta_2}{\sqrt{2n}}$, we can guarantee that $\|\Delta\mathbf{W}\|_F \leq \epsilon\|\mathbf{W}_L\|_F$.

According to the well-known sensitivity analysis of linear systems [9], we have

$$\frac{\|\mathbf{S}_L - \mathbf{S}_L^*\|_F}{\|\mathbf{S}_L\|_F} = \frac{\|\mathbf{s}_L - \mathbf{s}_L^*\|_2}{\|\mathbf{s}_L\|_2} \leq \frac{2\epsilon\kappa}{1-\epsilon\kappa} \tag{26}$$

where κ is the condition number under Frobenius norm of matrix $\mathbf{W}_L$, i.e., $\kappa = \|\mathbf{W}_L\|_F\|\mathbf{W}_L^{-1}\|_F$. By interpolating $\mathbf{S}_L$, at level l

$$\begin{aligned}\frac{\|\mathbf{S}_l^* - \mathbf{S}_l\|_F}{\|\mathbf{S}_l\|_F} &= \frac{\|\mathbf{Q}_l^T \cdots \mathbf{Q}_{L-1}^T(\mathbf{S}_l^* - \mathbf{S}_l)\mathbf{P}_{L-1}\cdots\mathbf{P}_l\|_F}{\|\mathbf{Q}_l^T \cdots \mathbf{Q}_{L-1}^T\mathbf{S}_l\mathbf{P}_{L-1}\cdots\mathbf{P}_l\|_F} \\ &= \frac{\|\mathbf{S}_L - \mathbf{S}_L^*\|_F}{\|\mathbf{S}_L\|_F} \leq \frac{2\epsilon\kappa}{1-\epsilon\kappa}\end{aligned} \tag{27}$$

REFERENCES

[1] Mohsen Bayati, Margot Gerritsen, David F Gleich, Amin Saberi, and Ying Wang. 2009. Algorithms for large, sparse network alignment problems. In *2009 Ninth IEEE International Conference on Data Mining*. IEEE, 705–710.

[2] Zheng Chen, Xinli Yu, Bo Song, Jianliang Gao, Xiaohua Hu, and Wei-Shih Yang. 2017. Community-based network alignment for large attributed network. In *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*. ACM, 587–596.

[3] Giovanni Ciriello, Marco Mina, Pietro H Guzzi, Mario Cannataro, and Concettina Guerra. 2012. AlignNemo: a local network alignment method to integrate homology and topology. *PloS one* 7, 6 (2012), e38107.

[4] Ronald R Coifman and Mauro Maggioni. 2006. Diffusion wavelets. *Applied and Computational Harmonic Analysis* 21, 1 (2006), 53–94.

[5] Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. 2016. Convolutional neural networks on graphs with fast localized spectral filtering. In *Advances in neural information processing systems*. 3844–3852.

[6] Inderjit S Dhillon, Yuqiang Guan, and Brian Kulis. 2007. Weighted graph cuts without eigenvectors a multilevel approach. *IEEE transactions on pattern analysis and machine intelligence* 29, 11 (2007), 1944–1957.

[7] Boxin Du and Hanghang Tong. 2018. FASTEN: Fast Sylvester Equation Solver for Graph Mining. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. ACM, 1339–1347.

[8] Boxin Du, Si Zhang, Nan Cao, and Hanghang Tong. 2017. First: Fast interactive attributed subgraph matching. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 1447–1456.

[9] Gene H Golub and Charles F Van Loan. 2012. *Matrix computations*. Vol. 3. JHU press.

[10] Somaye Hashemifar, Jianzhu Ma, Hammad Naveed, Stefan Canzar, and Jinbo Xu. 2016. ModuleAlign: module-based global alignment of protein–protein interaction networks. *Bioinformatics* 32, 17 (2016), i658–i664.

[11] Somaye Hashemifar and Jinbo Xu. 2014. HubAlign: an accurate and efficient method for global alignment of protein–protein interaction networks. *Bioinformatics* 30, 17 (2014), i438–i444.

[12] Jian Kang, Scott Freitas, Haichao Yu, Yinglong Xia, Nan Cao, and Hanghang Tong. 2018. X-rank: Explainable ranking in complex multi-layered networks. In *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*. ACM, 1959–1962.

[13] George Karypis and Vipin Kumar. 1998. Multilevelk-way partitioning scheme for irregular graphs. *Journal of Parallel and Distributed computing* 48, 1 (1998), 96–129.

[14] Risi Kondor and Horace Pan. 2016. The multiscale laplacian graph kernel. In *Advances in Neural Information Processing Systems*. 2990–2998.

[15] Risi Kondor, Nedelina Teneva, and Vikas Garg. 2014. Multiresolution matrix factorization. In *International Conference on Machine Learning*. 1620–1628.

[16] Danai Koutra, Hanghang Tong, and David Lubensky. 2013. Big-align: Fast bipartite graph alignment. In *2013 IEEE 13th International Conference on Data Mining*. IEEE, 389–398.

[17] Jure Leskovec, Jon Kleinberg, and Christos Faloutsos. 2007. Graph evolution: Densification and shrinking diameters. *ACM Transactions on Knowledge Discovery from Data (TKDD)* 1, 1 (2007), 2.

[18] Jure Leskovec and Julian J Mcauley. 2012. Learning to discover social circles in ego networks. In *Advances in neural information processing systems*. 539–547.

[19] Li Liu, William K Cheung, Xin Li, and Lejian Liao. 2016. Aligning Users across Social Networks Using Network Embedding. In *IJCAI*. 1774–1780.

[20] Rui Liu, Wei Cheng, Hanghang Tong, Wei Wang, and Xiang Zhang. 2015. Robust multi-network clustering via joint cross-domain cluster alignment. In *2015 IEEE International Conference on Data Mining*. IEEE, 291–300.

[21] Oren E Livne and Achi Brandt. 2012. Lean algebraic multigrid (LAMG): Fast graph Laplacian linear solver. *SIAM Journal on Scientific Computing* 34, 4 (2012), B499–B522.

[22] Hazel N Manners, Ahed Elmsallati, Pietro H Guzzi, Swarup Roy, and Jugal K Kalita. 2017. Performing local network alignment by ensembling global aligners. In *2017 IEEE International Conference on Bioinformatics and Biomedicine (BIBM)*. IEEE, 1316–1323.

[23] Yvan Notay. 2010. An aggregation-based algebraic multigrid method. *Electronic transactions on numerical analysis* 37, 6 (2010), 123–146.

[24] Kaare Brandt Petersen, Michael Syskind Pedersen, et al. 2008. The matrix cookbook. *Technical University of Denmark* 7, 15 (2008), 510.

[25] Erzsébet Ravasz and Albert-László Barabási. 2003. Hierarchical organization in complex networks. *Physical review E* 67, 2 (2003), 026112.

[26] David I Shuman, Mohammad Javad Faraji, and Pierre Vandergheynst. 2016. A multiscale pyramid transform for graph signals. *IEEE Transactions on Signal Processing* 64, 8 (2016), 2119–2134.

[27] Rohit Singh, Jinbo Xu, and Bonnie Berger. 2008. Global alignment of multiple protein interaction networks with application to functional orthology detection. *Proceedings of the National Academy of Sciences* 105, 35 (2008), 12763–12768.

[28] Jie Tang, Jing Zhang, Limin Yao, Juanzi Li, Li Zhang, and Zhong Su. 2008. Arnetminer: extraction and mining of academic social networks. In *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 990–998.

[29] Nedelina Teneva, Pramod Kaushik Mudrakarta, and Risi Kondor. 2016. Multiresolution matrix compression. In *Artificial Intelligence and Statistics*. 1441–1449.

[30] Shinji Umeyama. 1988. An eigendecomposition approach to weighted graph matching problems. *IEEE transactions on pattern analysis and machine intelligence* 10, 5 (1988), 695–703.

[31] Jiejun Xu, Hanghang Tong, Tsai-Ching Lu, Jingrui He, and Nadya Bliss. 2018. GTA 3 2018: Workshop on Graph Techniques for Adversarial Activity Analytics. In *Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining*. ACM, 803–803.

[32] Jaewon Yang and Jure Leskovec. 2015. Defining and evaluating network communities based on ground-truth. *Knowledge and Information Systems* 42, 1 (2015), 181–213.

[33] Wayne W Zachary. 1977. An information flow model for conflict and fission in small groups. *Journal of anthropological research* 33, 4 (1977), 452–473.

[34] Jiawei Zhang and S Yu Philip. 2015. Multiple anonymized social networks alignment. In *2015 IEEE International Conference on Data Mining*. IEEE, 599–608.

[35] 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*. ACM, 1345–1354.

[36] Si Zhang, Hanghang Tong, Jie Tang, Jiejun Xu, and Wei Fan. 2017. ineat: Incomplete network alignment. In *2017 IEEE International Conference on Data Mining (ICDM)*. IEEE, 1189–1194.

[37] Yutao Zhang, Jie Tang, Zhilin Yang, Jian Pei, and Philip S Yu. 2015. Cosnet: Connecting heterogeneous social networks with local and global consistency. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 1485–1494.

[38] Dawei Zhou, Jingrui He, Hongxia Yang, and Wei Fan. 2018. Sparc: Self-paced network representation for few-shot rare category characterization. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. ACM, 2807–2816.

[39] Dawei Zhou, Si Zhang, Mehmet Yigit Yildirim, Scott Alcorn, Hanghang Tong, Hasan Davulcu, and Jingrui He. 2017. A local algorithm for structure-preserving graph cut. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 655–664.

[40] Yao Zhou and Jingrui He. 2016. Crowdsourcing via Tensor Augmentation and Completion.. In *IJCAI*. 2435–2441.