Incomplete Network Alignment: Problem Definitions and Fast Solutions

SI ZHANG and HANGHANG TONG, University of Illinois at Urbana-Champaign
JIE TANG, Tsinghua University
JIEJUN XU, HRL Laboratories
WEI FAN, Tencent Medical AI Lab

Networks are prevalent in many areas and are often collected from multiple sources. However, due to the veracity characteristics, more often than not, networks are incomplete. Network alignment and network completion have become two fundamental cornerstones behind a wealth of high-impact graph mining applications. The state-of-the-art have been addressing these two tasks *in parallel*. That is, most of the existing network alignment methods have implicitly assumed that the topology of the input networks for alignment are perfectly known a priori, whereas the existing network completion methods admit either a single network (i.e., matrix completion) or multiple aligned networks (e.g., tensor completion). In this article, we argue that network alignment and completion are inherently complementary with each other, and hence propose to jointly address them so that the two tasks can mutually benefit from each other. We formulate the problem from the optimization perspective, and propose an effective algorithm (iNeAt) to solve it. The proposed method offers two distinctive advantages. First (*Alignment accuracy*), our method benefits from the higher-quality input networks while mitigates the effect of the incorrectly inferred links introduced by the completion task itself. Second (*Alignment efficiency*), thanks to the low-rank structure of the complete networks and the alignment matrix, the alignment process can be significantly accelerated. We perform extensive experiments which show that (1) the network completion can significantly improve the alignment accuracy, i.e., up to 30% over the baseline methods; (2) the network alignment can in turn help recover more missing edges than the baseline methods; and (3) our method achieves a good balance between the running time and the accuracy, and scales with a provable *linear* complexity in both time and space.

CCS Concepts: • **Information systems → Data mining**;

Additional Key Words and Phrases: Incomplete network alignment, network completion, low rank

ACM Reference format:
Si Zhang, Hanghang Tong, Jie Tang, Jiejun Xu, and Wei Fan. 2020. Incomplete Network Alignment: Problem Definitions and Fast Solutions. *ACM Trans. Knowl. Discov. Data* 14, 4, Article 38 (May 2020), 26 pages.
https://doi.org/10.1145/3384203

This work is supported by the National Science Foundation under grant No. 1947135 and 1715385, by the NSF Program on Fairness in AI in collaboration with Amazon under award No. 1939725, by the United States Air Force and DARPA under contract number FA8750-17-C-0153, and Department of Homeland Security under Grant Award Number 2017-ST-061-QA0001.

Authors' addresses: S. Zhang, 201 N Goodwin Ave, Urbana, IL, 61801; email: sizhang2@illinois.edu; H. Tong, 201 N Goodwin Ave, Urbana, IL 6180; email: htong@illinois.edu; J. Tang, office 1-308, FIT Building, Tsinghua University, Beijing, 100084, China; email: jietang@tsinghua.edu.cn; J. Xu, 3011 Malibu Canyon Rd #4797, Malibu, CA 90265; email: jxu@hrl.com; W. Fan, 2747 Park Blvd, Palo Alto, CA 94306; email: wei.fan@gmail.com.

1556-4681/2020/05-ART38 $15.00
https://doi.org/10.1145/3384203

1 INTRODUCTION

Networks are prevalent and naturally appear in many areas. More often than not, in the big data era, networks in many high-impact applications are collected from *multiple* sources (i.e., *variety*), such as social networks from different social platforms, protein–protein interaction (PPI) networks from multiple tissues, and transaction networks from multiple financial institutes. In order to integrate the considerable information associated with multiple networks, network alignment is of key importance to find the node correspondence across networks. For example, by aligning the same users in different transaction networks, the transaction patterns of users can be comprehended to enhance the financial fraud detection. However, real-world networks are often *incomplete* (i.e., *veracity*) due to, for instance, the difficulties in data collections. As such, network completion (e.g., to infer the missing links) has become another key task which benefits many graph mining applications by providing higher-quality networks if handled properly.

Although the multi-sourced and incomplete characteristics often *co-exist* in many real networks, the state-of-the-arts have been largely addressing network alignment and network completion problems *in parallel*. For example, most of the existing network alignment methods based on topological consistency have implicitly assumed that the topology of the input networks for alignment are perfectly known a priori [16, 42]. On the other hand, the existing network completion methods aim to infer the missing links in either a single network (e.g., by matrix completion [27]) or multiple networks that are aligned beforehand (e.g., by tensor completion [24]). How can we align two input incomplete networks when missing edges are unobserved in them?

A natural choice could be *completion-then-alignment*. That is, we first separately complete the missing edges in the input networks by some existing network completion methods, followed by the alignment across the resulting complete networks. However, there exist some fundamental limits of this strategy on the alignment performance. First (*Alignment accuracy*), the promise of this strategy lies in that by inferring the missing links of each input network, it would provide higher-quality input networks for the alignment task. However, the completion task itself might introduce noise (e.g., truly nonexistent edges), which might compromise, or even prevail the benefits of the correctly inferred missing links for the alignment task. Second (*Alignment efficiency*), the network alignment alone is already computationally costly. Most of the existing methods (even with approximation, such as [43]) have a time/space complexity at least $O(n^2)$, where n is the number of nodes of the input networks, mainly due to the computation/storage of the alignment matrix and the sparse matrix-matrix multiplication between the input adjacency matrices and the alignment matrix.[1] Yet, network completion would make each input network even denser by adding the missing edges. As a result, if we simply conduct the network alignment task on such densified networks, it might make the computation even more intensive.

To address these limitations, we hypothesize that network alignment and network completion can inherently complement each other due to the following reasons. First, *(H1) alignment helps completion*. Intuitively, when many nodes in one network share similar connectivity patterns with their corresponding aligned nodes (e.g., connecting to the similar sets of nodes) in another network, the knowledge about the existence or absence of links in one network could help inferring the missing links in another network via alignment if we can find such node correspondences across networks. Second, *(H2) completion helps alignment*. As introduced before, network completion could potentially improve the qualities of input networks, leading to the enhancement of the alignment accuracy. Moreover, network completion itself implicitly assumes a low-rank structure

[1] Although the *empirical* runtime of some existing methods (e.g., *BigAlign* [16]) is *near-linear*, the big-O time complexity of these methods is still quadratic.

on the input networks, which, if harnessed appropriately, will actually *accelerate* the alignment process as we will show in the article.

Armed with these hypotheses, we propose to *jointly* address network alignment and network completion problems so that the two tasks could mutually benefit from each other. To be specific, in order to leverage alignment for the completion task, we impose the low-rank structure on the underlying (true) network, which matches not only the observed links of the corresponding network, but also the *auxiliary observations* from the other network via the alignment matrix. Second, in order to leverage the network completion for the alignment, we recast the network alignment problem via the low-rank structures of the *complete* networks, which not only improves the alignment accuracy, but also speeds up the alignment process. We formulate them into a joint optimization problem and propose an effective algorithm to solve it.

The main contributions of the article are summarized as:

- **Problem Definition.** To our best knowledge, we are the first to jointly address the network alignment and network completion tasks in an optimization framework.
- **Algorithm and Analysis.** We propose an effective algorithm (iNeAt) based on the multiplicative update to solve the optimization. We also analyze its correctness, convergence, and complexity. In particular, we prove that the low-rank structure of the complete networks guarantees a low-rank structure of the alignment matrix, which in turn reduces the time complexity of each iterative update to be *linear*. To our best knowledge, this is the first known network alignment algorithm with a provable linear time complexity.
- **Experiments.** We evaluate the effectiveness and efficiency of the proposed algorithm by extensive experiments. The experimental results demonstrate that (1) network alignment and network completion can indeed benefit from each other in terms of alignment accuracy and missing edges recovery rate, (2) our algorithm iNeAt achieves a better alignment and completion quality, and meanwhile is faster than most of the baseline methods, and (3) our algorithm is only *linear* w.r.t. the number of nodes in the networks.

The rest of the article is organized as follows. Section 2 defines the incomplete network alignment problem and provides some preliminaries of the article. Section 3 presents the proposed optimization formulation of iNeAt and Section 4 gives an effective optimization algorithm, followed by some analyses. Section 5 presents the experimental results. Related work and conclusion are given in Sections 6 and 7.

2 PROBLEM DEFINITION

2.1 Problem Definition

Table 1 summarizes the main symbols and notations used throughout the article. We use bold uppercase letters for matrices (e.g., $\mathbf{A}$), bold lowercase letters for vectors (e.g., $\mathbf{s}$), and lowercase letters (e.g., α) for scalars. 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., $\mathbf{A}^T$ is the transpose of $\mathbf{A}$). The vectorization of a matrix (in the column order) is denoted by $\text{vec}(\cdot)$, and the result vector is denoted by the corresponding bold lowercase letter (e.g., $\mathbf{s} = \text{vec}(\mathbf{S})$). Equivalently, the transformation of a vector to its corresponding matrix is denoted by a de-vectorization operator $\text{mat}(\cdot)$ (e.g., $\mathbf{S} = \text{mat}(\mathbf{s})$). The trace of a matrix is denoted by $\text{Tr}(\cdot)$, and the diagonal matrix of a vector is denoted by $\text{diag}(\cdot)$.

Many real-world networks are incomplete with missing edges. Although some incompleteness scenarios may be possible (e.g., with the probabilities whether edges exist known a priori), in our article, we only consider the network incompleteness where we only have the knowledge

Table 1. Symbols and Notations

Symbols	Definition
$\mathcal{G}_1, \mathcal{G}_2$	incomplete networks
$\mathbf{A}_1, \mathbf{A}_2$	two adjacency matrices of $\mathcal{G}_1$ and $\mathcal{G}_2$
n_1, n_2	# of nodes in $\mathcal{G}_1$ and $\mathcal{G}_2$
m_1, m_2	# of nodes in $\mathcal{G}_1$ and $\mathcal{G}_2$
$\mathbf{S}$	an $n_2 \times n_1$ alignment matrix between $\mathcal{G}_2$ and $\mathcal{G}_1$
$P_\Omega(\cdot), P_{\bar{\Omega}}(\cdot)$	an operator to project only to observed (unobserved) entries
$\mathbf{U}_1, \mathbf{V}_1, \mathbf{U}_2, \mathbf{V}_2$	low rank factorizations of $\mathbf{A}_1$ and $\mathbf{A}_2$
$\mathbf{P}_{\Omega_1}, \mathbf{P}_{\Omega_2}$	projection matrix, all 1s at all observed entries
$\mathbf{1}_1, \mathbf{1}_2$	1s vectors of length n_1 and n_2 respectively
λ, γ, β	parameters
Tr[·]	trace operator
diag(·)	diagonal matrix of a vector
vec(·), mat(·)	vectorization and de-vectorization operator
rank(·)	the rank of a matrix
eig(·)	eigenvalues of a matrix

about the existence (i.e., a value of 1) or the absence (i.e., a value of 0) of certain entries (denoted by the set Ω) of its adjacency matrix. For the rest entries in the adjacency matrix, we do not know if the corresponding links exist or not, and hence are represented as the question mark ?. Figure 1 presents an illustrative example. All solid lines represent the observed existing edges. As we can see in Figure 1(a), the set of nodes (1,2,3,4) in the first incomplete network have similar topology to the nodes $(6', 7', 8', 9')$, possibly leading to a wrong alignment result that these two sets of nodes are aligned within each other. However, the complete networks in Figure 1(b) (by filling all the red lines) are identical, such as the cliques formed by nodes $(1, 2, 3, 4)$ and $(1', 2', 3', 4')$. Thus, the set of nodes $(1, 2, 3, 4)$ can be aligned to nodes $(1', 2', 3', 4')$, respectively, so can the rest of nodes. On the other hand, by completing two networks separately, noisy edges might be incorrectly added (e.g., edge $(4, 6)$) and the true network structure would fail to be recovered. The incorrectly recovered networks may further mislead the alignment results. Therefore, how to align the incomplete networks while completing them is the key challenge this article aims to address.

Problem 1. *Incomplete Network Alignment.*

Given: *(1) Incomplete adjacency matrices* $\mathbf{A}_1$, $\mathbf{A}_2$ *of two undirected networks* $\mathcal{G}_1, \mathcal{G}_2$, *and (2-optional) a prior node similarity matrix* $\mathbf{H}$ *across networks.*

Output: *(1) the* $n_2 \times n_1$ *alignment/similarity matrix* $\mathbf{S}$, *where* $\mathbf{S}(x, a)$ *represents to what extent node-a in* $\mathcal{G}_1$ *is aligned with node-x in* $\mathcal{G}_2$, *and (2) complete adjacency matrices* $\mathbf{A}_1^*$, *and* $\mathbf{A}_2^*$.

2.2 Preliminaries

A - Network Alignment. Most existing network alignment algorithms (such as *IsoRank* [34] and *FINAL* [43, 44]), explicitly or implicitly, are based on the *topology consistency* principle. Take *FINAL* as an example, the topology consistency principle can be stated as follows.[2] Given two pairs of nodes, say (1) node-a in $\mathcal{G}_1$ and node-x in $\mathcal{G}_2$ and (2) node-b in $\mathcal{G}_1$ and node-y in $\mathcal{G}_2$, if nodes a and

[2]In [43], the authors generalize the topology consistency principle to further accommodate the additional node/edge attribute information, which is outside the scope of this article.

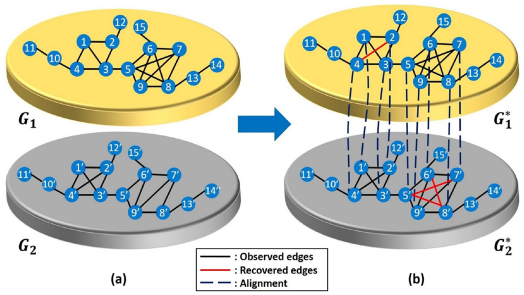


Fig. 1. An illustrative example. Figure 1(a) shows the input incomplete networks and Figure 1(b) shows part of the alignment across two complete networks.

b are close neighbors and nodes x and y are also close neighbors, the *topology consistency* principle assumes the similarity between a and x, and that between their respective close neighbors b and y to be consistent, i.e., small $[\hat{\mathbf{S}}(a,x) - \hat{\mathbf{S}}(b,y)]^2 \mathbf{A}_1(a,b)\mathbf{A}_2(x,y)$, where $\hat{\mathbf{S}}$ is the similarity matrix. Mathematically, this naturally leads to the following optimization problem.

$$\min_{\hat{\mathbf{s}}} \alpha \hat{\mathbf{s}}^T(\mathbf{D} - \mathbf{A}_1 \otimes \mathbf{A}_2)\hat{\mathbf{s}} + (1-\alpha)\|\mathbf{D}^{\frac{1}{2}}(\hat{\mathbf{s}} - \mathbf{h})\|_F^2 \tag{1}$$

where $\hat{\mathbf{s}}, \mathbf{h}$ are the vectorization of the similarity matrix $\hat{\mathbf{S}}$ and the prior similarity matrix $\mathbf{H}$, respectively. $\mathbf{D} = \mathbf{D}_1 \otimes \mathbf{D}_2$ and $\mathbf{D}_1, \mathbf{D}_2$ are the diagonal degree matrix of $\mathbf{A}_1, \mathbf{A}_2$, respectively. Note that instead of using $\hat{\mathbf{S}}$ to infer the alignment as in [43], we use the scaled similarity matrix $\mathbf{S}$ as the "soft" alignment matrix throughout our article, where $\mathbf{S}$ is the matrix form of $\mathbf{s} = \mathbf{D}\hat{\mathbf{s}}$ (i.e., $\mathbf{S} = \text{mat}(\mathbf{D}\hat{\mathbf{s}})$). In other words, the entries in the alignment matrix $\mathbf{S}$ measure to what extent the two corresponding nodes are aligned together. Besides, the second regularization term in Equation (1) is to avoid trivial solutions, such as a zero matrix $\hat{\mathbf{S}}$.

In order to solve the network alignment problem in Equation (1), we can either use an iterative algorithm with a time complexity of $O(nm)$ and a space complexity $O(n^2)$, or resort to its closed-form solution whose time complexity could be as high as $O(n^6)$ where we assume that the two networks have a comparable number of edges and nodes, i.e., $O(m) = O(m_1) = O(m_2)$ and $O(n) = O(n_1) = O(n_2)$. In [43], the authors proposed to approximate the closed-form solution via eigenvalue decomposition. But it is still *quadratic* in both time and space.

B - Network Completion. As mentioned earlier, incomplete networks might have many unobserved missing edges, which could significantly change the true network structure and hence mislead the topology-based network alignment. One straightforward way to address this issue is by using matrix completion. Most of the existing matrix completion methods are centered around minimizing the nuclear norm of the matrix [32]. However, since real-world networks are usually very large, it is very costly to directly minimize the nuclear norm of the adjacency matrices. In [33], the authors show that the nuclear norm $\|\mathbf{A}_1\|_* = \min_{\mathbf{U}_1,\mathbf{V}_1} \frac{1}{2}(\|\mathbf{U}_1\|_F^2 + \|\mathbf{V}_1\|_F^2)$ where $\mathbf{A}_1 = \mathbf{U}_1\mathbf{V}_1^T$, which allows the factorization-based completion methods. To be specific, we can recover the complete networks by minimizing the following objective function:

$$\begin{aligned} J_1(\mathbf{U}_1,\mathbf{V}_1,\mathbf{U}_2,\mathbf{V}_2) = & \underbrace{\frac{1}{2}\|P_{\Omega_1}(\mathbf{A}_1 - \mathbf{U}_1\mathbf{V}_1^T)\|_F^2 + \frac{\lambda}{2}(\|\mathbf{U}_1\|_F^2 + \|\mathbf{V}_1\|_F^2)}_{\text{network completion on } \mathbf{A}_1} \\ & + \underbrace{\frac{1}{2}\|P_{\Omega_2}(\mathbf{A}_2 - \mathbf{U}_2\mathbf{V}_2^T)\|_F^2 + \frac{\lambda}{2}(\|\mathbf{U}_2\|_F^2 + \|\mathbf{V}_2\|_F^2)}_{\text{network completion on } \mathbf{A}_2} \end{aligned} \tag{2}$$

where the operator P_{Ω_1} projects the value to the observed set Ω_1 of $\mathbf{A}_1$, e.g., $P_{\Omega_1}((\mathbf{U}_1\mathbf{V}_1^T)(i,j)) = (\mathbf{U}_1\mathbf{V}_1^T)(i,j)$ for any $(i,j) \in \Omega_1$, otherwise 0; and operator P_{Ω_2} is defined similarly.

3 PROPOSED OPTIMIZATION FORMULATION

In this section, we present the proposed optimization formulation to solve Problem 1. First, we present how to formulate the alignment task in the form of two complete networks. A key contribution here is that we prove that the low-rank structure of the complete networks guarantees a low-rank structure of the alignment matrix. Then we present how to leverage the alignment matrix to infer missing edges across networks, followed by the overall optimization formulation.

3.1 Network Completion Helps Network Alignment

By performing the network completion on both incomplete networks, the structure of the underlying networks could be recovered so that we can perform the alignment task across higher-quality networks. We use the factorization-based network completion (i.e., Equation (2)) and denote these two complete networks by $\mathbf{A}_1^* = \mathbf{U}_1\mathbf{V}_1^T$ and $\mathbf{A}_2^* = \mathbf{U}_2\mathbf{V}_2^T$, where $\mathbf{U}_i$ and $\mathbf{V}_i$ $(i = 1, 2)$ are the factorization matrices of rank-r. We adopt Equation (1) to perform the network alignment task. Note that in general, we cannot guarantee the recovered adjacency matrices ($\mathbf{A}_1^*$ and $\mathbf{A}_2^*$) to be symmetric because $\mathbf{V}_1$ ($\mathbf{V}_2$) may not be identical to $\mathbf{U}_1$ ($\mathbf{U}_2$). This leads to a slightly different objective function from Equation (1) to align directed networks. Specifically, based on the *topology consistency* (i.e., small $[\hat{\mathbf{S}}(a,x) - \hat{\mathbf{S}}(b,y)]^2\mathbf{A}_1(a,b)\mathbf{A}_2(x,y)$ in two directed networks), the optimization problem is formulated as follows:

$$\min_{\hat{\mathbf{s}}} \alpha\hat{\mathbf{s}}^T(\hat{\mathbf{D}} - \mathbf{A}_1^* \otimes \mathbf{A}_2^*)\hat{\mathbf{s}} + (1-\alpha)\|\hat{\mathbf{D}}^{\frac{1}{2}}(\hat{\mathbf{s}} - \mathbf{h})\|_F^2 \tag{3}$$

where $\hat{\mathbf{D}} = \frac{\mathbf{D}_1 \otimes \mathbf{D}_2 + \hat{\mathbf{D}}_1 \otimes \hat{\mathbf{D}}_2}{2}$, $\mathbf{D}_1 = \text{diag}(\mathbf{U}_1\mathbf{V}_1^T\mathbf{1}_1)$, and $\hat{\mathbf{D}}_1 = \text{diag}(\mathbf{1}_1^T\mathbf{U}_1\mathbf{V}_1^T)$ are the outdegree matrix and indegree matrix of $\mathbf{A}_1^*$, respectively. $\mathbf{D}_2$ and $\hat{\mathbf{D}}_2$ are defined in a similar way.

However, directly solving the above problem requires at least $O(n^2)$ time complexity, even with approximation. To address this issue, we give the following lemma, which states the alignment matrix $\mathbf{S}$ under the *topology consistency* (i.e., Equation (3)) intrinsically consists of a *low-rank* structure, thanks to the low-rank structure of two complete adjacency matrices.

Lemma 1 [Low-Rank Structure of the Alignment Matrix $\mathbf{S}$]. *Let $\hat{\mathbf{s}}$ be the solution of Equation (3) where $\mathbf{A}_1^* = \mathbf{U}_1\mathbf{V}_1^T$ and $\mathbf{A}_2^* = \mathbf{U}_2\mathbf{V}_2^T$ are two complete rank-r adjacency matrices. Let the alignment matrix $\mathbf{S}$ be the scaled similarity matrix $\mathbf{S} = mat(\hat{\mathbf{D}}\hat{\mathbf{s}})$ and $\mathbf{H}$ be the prior similarity matrix, then if $\alpha < 0.5$, the alignment matrix can be expressed as $\mathbf{S} = \alpha\mathbf{U}_2\mathbf{M}\mathbf{U}_1 + (1-\alpha)\mathbf{H}$ where $\mathbf{M}$ is an $r_2 \times r_1$ matrix and r_1, r_2 are the ranks of $\mathbf{A}_1^*$ and $\mathbf{A}_2^*$, respectively.*

Proof. Followed by Equation (3), the closed-form solution of similarity matrix $\hat{\mathbf{S}}$ can be computed by using Woodbury matrix identity [31] as below

$$\hat{\mathbf{s}} = (1-\alpha)\hat{\mathbf{D}}^{-1}\mathbf{h} + \alpha(1-\alpha)\hat{\mathbf{D}}^{-1}\mathbf{U}\Lambda^{-1}\mathbf{V}^T\hat{\mathbf{D}}^{-1}\mathbf{h} \tag{4}$$

where $\mathbf{U} = \mathbf{U}_1 \otimes \mathbf{U}_2$, $\mathbf{V} = \mathbf{V}_1 \otimes \mathbf{V}_2$, $\Lambda = \mathbf{I} - \alpha\mathbf{V}^T\hat{\mathbf{D}}^{-1}\mathbf{U}$.

First, we rewrite Λ^{-1} as follows. Since for any two matrices $\mathbf{X}, \mathbf{Y}$, the eigenvalues of their product satisfies $\text{eig}(\mathbf{XY}) = \text{eig}(\mathbf{YX})$ [31], we obtain

$$\begin{aligned}
|\text{eig}(\alpha\mathbf{V}^T\hat{\mathbf{D}}^{-1}\mathbf{U})| &= |\text{eig}(\alpha\mathbf{U}\mathbf{V}^T\hat{\mathbf{D}}^{-1})| \\
&\leq |\text{eig}(2\alpha\mathbf{U}\mathbf{V}^T(\mathbf{D}_1 \otimes \mathbf{D}_2)^{-1})| \\
&= 2\alpha|\text{eig}((\mathbf{U}_1\mathbf{V}_1^T\mathbf{D}_1^{-1}) \otimes (\mathbf{U}_2\mathbf{V}_2^T\mathbf{D}_2^{-1}))|
\end{aligned}$$

Here, the term $\mathbf{U}_1\mathbf{V}_1^T\mathbf{D}_1^{-1}$ represents a weighted directed network whose adjacency matrix has eigenvalues within $(-1, 1)$, so as the term $\mathbf{U}_2\mathbf{V}_2^T\mathbf{D}_2^{-1}$. Thus, if $\alpha < 0.5$, according to the spectrum property of Kronecker product, we have:

$$2\alpha|\text{eig}((\mathbf{U}_1\mathbf{V}_1^T\mathbf{D}_1^{-1}) \otimes (\mathbf{U}_2\mathbf{V}_2^T\mathbf{D}_2^{-1}))| < 1$$

Then, we can use Neumann expansion on Λ^{-1} as:

$$\Lambda^{-1} = \sum_{k=0}^{\infty}(2\alpha)^k[\mathbf{V}^T(2\hat{\mathbf{D}})^{-1}\mathbf{U}]^k \tag{5}$$

Next, we rewrite $(2\hat{\mathbf{D}})^{-1}$ as follows. Denote $\bar{\mathbf{D}}_1 = \mathbf{D}_1 + \hat{\mathbf{D}}_1$ and $\bar{\mathbf{D}}_2 = \mathbf{D}_2 + \hat{\mathbf{D}}_2$, we have:

$$\begin{aligned}
(2\hat{\mathbf{D}})^{-1} = (\mathbf{D}_1 \otimes \mathbf{D}_2 + \hat{\mathbf{D}}_1 \otimes \hat{\mathbf{D}}_2)^{-1} &= \{(\bar{\mathbf{D}}_1 \otimes \bar{\mathbf{D}}_2)[\mathbf{I} - (\bar{\mathbf{D}}_1^{-1} \otimes \bar{\mathbf{D}}_2^{-1})(\mathbf{D}_1 \otimes \hat{\mathbf{D}}_2 + \hat{\mathbf{D}}_1 \otimes \mathbf{D}_2)]\}^{-1} \\
&= [\mathbf{I} - (\bar{\mathbf{D}}_1^{-1} \otimes \bar{\mathbf{D}}_2^{-1})(\mathbf{D}_1 \otimes \hat{\mathbf{D}}_2 + \hat{\mathbf{D}}_1 \otimes \mathbf{D}_2)]^{-1}(\bar{\mathbf{D}}_1^{-1} \otimes \bar{\mathbf{D}}_2^{-1}) \\
&= \sum_{j=0}^{\infty}[(\bar{\mathbf{D}}_1^{-1}\mathbf{D}_1) \otimes (\bar{\mathbf{D}}_2^{-1}\hat{\mathbf{D}}_2) + (\bar{\mathbf{D}}_1^{-1}\hat{\mathbf{D}}_1) \otimes (\bar{\mathbf{D}}_2^{-1}\mathbf{D}_2)]^j(\bar{\mathbf{D}}_1^{-1} \otimes \bar{\mathbf{D}}_2^{-1}) \\
&= \sum_{j=0}^{\infty}\sum_{i=0}^{j}\binom{j}{i}[(\bar{\mathbf{D}}_1^{-1}\mathbf{D}_1)^i(\bar{\mathbf{D}}_1^{-1}\hat{\mathbf{D}}_1)^{j-i}\bar{\mathbf{D}}_1^{-1}] \otimes [(\bar{\mathbf{D}}_2^{-1}\hat{\mathbf{D}}_2)^i(\bar{\mathbf{D}}_2^{-1}\mathbf{D}_2)^{j-i}\bar{\mathbf{D}}_2^{-1}]
\end{aligned}$$

By substituting the above equation into Equation (5), the matrix Λ^{-1} can be derived as:

$$\Lambda^{-1} = \sum_{k=0}^{\infty}\sum_{j=0}^{\infty}\sum_{i=0}^{j}(2\alpha)^k\binom{j}{i}^k [\mathbf{V}_1^T(\bar{\mathbf{D}}_1^{-1}\mathbf{D}_1)^i(\bar{\mathbf{D}}_1^{-1}\hat{\mathbf{D}}_1)^{j-i}\bar{\mathbf{D}}_1^{-1}\mathbf{U}_1]^k \otimes [\mathbf{V}_2^T(\bar{\mathbf{D}}_2^{-1}\hat{\mathbf{D}}_2)^i(\bar{\mathbf{D}}_2^{-1}\mathbf{D}_2)^{j-i}\bar{\mathbf{D}}_2^{-1}\mathbf{U}_2]^k \tag{6}$$

Denote $\mathbf{s} = \hat{\mathbf{D}}\hat{\mathbf{s}}$ and $\hat{\mathbf{h}} = \hat{\mathbf{D}}^{-1}\mathbf{h}$. Armed with the Kronecker product property $\text{vec}(\mathbf{ABC}) = (\mathbf{C}^T \otimes \mathbf{A})\text{vec}(\mathbf{B})$, by substituting Equation (6) into Equation (4), we obtain the alignment matrix $\mathbf{S} = \text{mat}(\hat{\mathbf{D}}\hat{\mathbf{s}})$ as:

$$\mathbf{S} = \alpha\mathbf{U}_2\mathbf{M}\mathbf{U}_1^T + (1-\alpha)\mathbf{H} \tag{7}$$

where $\mathbf{M}$ is an $r_2 \times r_1$ matrix and is computed by:

$$\mathbf{M} = (1-\alpha)\sum_{k=0}^{\infty}\sum_{j=0}^{\infty}\sum_{i=0}^{j}2^k\alpha^k\binom{j}{i}^k [\mathbf{V}_2^T(\bar{\mathbf{D}}_2^{-1}\hat{\mathbf{D}}_2)^i(\bar{\mathbf{D}}_2^{-1}\mathbf{D}_2)^{j-i}\bar{\mathbf{D}}_2^{-1}\mathbf{U}_2]^k\mathbf{V}_2^T\hat{\mathbf{H}}\mathbf{V}_1[\mathbf{U}_1^T(\bar{\mathbf{D}}_1^{-1}\mathbf{D}_1)^i(\bar{\mathbf{D}}_1^{-1}\hat{\mathbf{D}}_1)^{j-i}\bar{\mathbf{D}}_1^{-1}\mathbf{V}_1]^k \tag{8}$$

□

Remarks. Equation (7) suggests that the alignment matrix $\mathbf{S}$ consists of two parts, including a low-rank structure and an additive term $\mathbf{H}$ to reflect the prior knowledge and is a convex combination of these two parts. Such a convex optimization follows Equation (3) where a regularization term is added to minimize the inconsistency between the alignment result and the prior information. Note that other types of regularization in Equation (3) can lead to more complex combinations with the prior knowledge which may utilize both the reliable and the unreliable prior information in a better way. However, we only consider Equations (3) and (7) in this article and leave the more complex combinations to future works.

In practice, the prior knowledge matrix $\mathbf{H}$ is either low-rank (e.g., a rank-one uniform matrix) or very sparse. Having this in mind, we will mainly focus on how to learn the true low-rank structure part of $\mathbf{S}$ (i.e., $\mathbf{U}_2\mathbf{M}\mathbf{U}_1$) from the input incomplete networks. This naturally leads to the following effective strategy. First, we temporarily treat the low-rank structure part as the alignment matrix to be solved in the optimization problems (i.e., $\mathbf{S} \approx \mathbf{U}_2\mathbf{M}\mathbf{U}_1$). After $\mathbf{U}_2, \mathbf{M}, \mathbf{U}_1$ are obtained, we can then calibrate the result by averaging between the learned $\mathbf{S}$ and the prior knowledge $\mathbf{H}$ to further

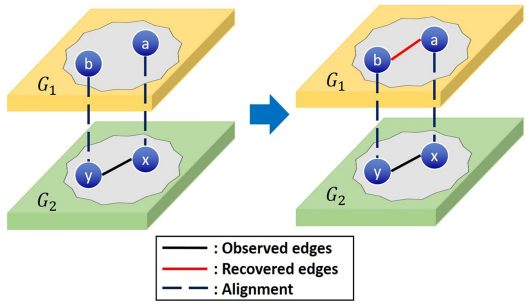


Fig. 2. Network completion via the alignment.

emphasize the importance of the prior knowledge, i.e., $\mathbf{S} \leftarrow (1-\alpha)\mathbf{H} + \alpha\mathbf{S}$. As we will show in the next section, a direct benefit of this strategy is that we can reduce the overall complexity (for both space and time cost) to be *linear*.

To take advantages of the low-rank structure of $\mathbf{S}$ under the above strategy, instead of minimizing Equation (3) regarding the similarity matrix $\hat{\mathbf{S}}$, we alternatively optimize the topology consistency on the low-rank structure of alignment matrix $\mathbf{S} = \mathbf{U}_2\mathbf{M}\mathbf{U}_1$ without the second regularization term, i.e., minimizing $\mathbf{s}^T(\hat{\mathbf{D}} - \mathbf{A}_1^* \otimes \mathbf{A}_2^*)\mathbf{s}$. Given $\mathbf{A}_1^* = \mathbf{U}_1\mathbf{V}_1^T$, $\mathbf{A}_2^* = \mathbf{U}_2\mathbf{V}_2^T$, by using the properties $\text{vec}(\mathbf{A})^T\text{vec}(\mathbf{B}) = \text{Tr}(\mathbf{A}^T\mathbf{B})$ and $\text{vec}(\mathbf{ABC}) = (\mathbf{C}^T \otimes \mathbf{A})\text{vec}(\mathbf{B})$, network alignment across the complete networks can be formulated as minimizing the following objective function:

$$\begin{aligned} J_2(\mathbf{U}_1, \mathbf{V}_1, \mathbf{U}_2, \mathbf{V}_2, \mathbf{M}) &= \frac{\gamma}{2}\mathbf{s}^T\text{vec}(\mathbf{D}_2\mathbf{S}\mathbf{D}_1 + \hat{\mathbf{D}}_2\mathbf{S}\hat{\mathbf{D}}_1) - \gamma\mathbf{s}^T\text{vec}(\mathbf{U}_2\mathbf{V}_2^T\mathbf{S}\mathbf{V}_1\mathbf{U}_1^T) \\ &= \underbrace{\frac{\gamma}{2}\text{Tr}(\mathbf{D}_2\mathbf{U}_2\mathbf{M}\mathbf{U}_1^T\mathbf{D}_1\mathbf{U}_1\mathbf{M}^T\mathbf{U}_2^T + \hat{\mathbf{D}}_2\mathbf{U}_2\mathbf{M}\mathbf{U}_1^T\hat{\mathbf{D}}_1\mathbf{U}_1\mathbf{M}^T\mathbf{U}_2^T)}_{\text{alignment across complete networks}} \\ &\quad - \underbrace{\gamma\text{Tr}(\mathbf{U}_2\mathbf{V}_2^T\mathbf{U}_2\mathbf{M}\mathbf{U}_1^T\mathbf{V}_1\mathbf{U}_1^T\mathbf{U}_1\mathbf{M}^T\mathbf{U}_2^T)}_{\text{alignment across complete networks}} \end{aligned} \tag{9}$$

3.2 Network Alignment Helps Network Completion

Despite the effectiveness of the factorization-based network completion methods (i.e., Equation (2)), in some applications, the information of a single network alone might be insufficient to correctly infer the missing edges. Meanwhile, the alignment across the two networks may provide extra hints of how to infer the missing edges. To be specific, since the aligned nodes are likely to share similar connectivity patterns, the observed existing edges in one network could potentially help recover the missing edges in the other network via the alignment matrix. Figure 2 presents an illustrative example. Here, node-a in $\mathcal{G}_1$ and node-x in $\mathcal{G}_2$ are aligned together, and the neighbor of x (say node-y) is aligned with the neighbor of a (e.g., node-b), which is not observed to connect with a. If we perform the completion solely based on the observed information of $\mathcal{G}_1$, we might probably conclude that the edge between a and b does not exist. However, the facts that (1) a and x are aligned, (2) b and y are aligned, and (3) there is an edge between x and y might provide an *auxiliary confidence* about the existence of the edge between a and b. In general, we can estimate such auxiliary confidence of the existence of the edge between a and b in $\mathcal{G}_1$ as:

$$\mathbf{A}_1^*(a,b) \approx \sum_{x,y}^{n_2} \mathbf{S}(a,x)\mathbf{S}(b,y)\mathbf{A}_2(x,y) = (\mathbf{S}^T\mathbf{A}_2\mathbf{S})(a,b) \tag{10}$$

where $\mathbf{S} = \mathbf{U}_2\mathbf{M}\mathbf{U}_1^T$ is the alignment matrix learned from the topology consistency.

In our experiments, we find that such auxiliary confidence is most powerful to estimate the existence/absence of an edge (a, b) when such an edge itself is not observed in $\mathcal{G}_1$ (i.e.,$(a, b) \in \bar{\Omega}_1$). Mathematically, this can be formulated as the following objective function:

$$J_3(\mathbf{U}_1, \mathbf{V}_1, \mathbf{U}_2, \mathbf{V}_2, \mathbf{M}) = \underbrace{\frac{\beta}{2}\|P_{\bar{\Omega}_1}(\mathbf{U}_1\mathbf{V}_1^T - \mathbf{U}_1\mathbf{M}^T\mathbf{U}_2^T\mathbf{A}_2\mathbf{U}_2\mathbf{M}\mathbf{U}_1^T)\|_F^2}_{\text{completion of } \mathcal{G}_1 \text{ based on the observed edges in } \mathcal{G}_2} + \underbrace{\frac{\beta}{2}\|P_{\bar{\Omega}_2}(\mathbf{U}_2\mathbf{V}_2^T - \mathbf{U}_2\mathbf{M}\mathbf{U}_1^T\mathbf{A}_1\mathbf{U}_1\mathbf{M}^T\mathbf{U}_2^T)\|_F^2}_{\text{completion of } \mathcal{G}_2 \text{ based on the observed edges in } \mathcal{G}_1} \tag{11}$$

where $\bar{\Omega}_1$ and $\bar{\Omega}_2$ are the unobserved set of $\mathbf{A}_1$ and $\mathbf{A}_2$.

3.3 Overall Objective Function

We impose the nonnegativity constraints on all the variables $\mathbf{U}_1, \mathbf{V}_1, \mathbf{U}_2, \mathbf{V}_2, \mathbf{M}$ to guarantee that all the entries in matrices $\mathbf{A}_1^*, \mathbf{A}_2^*, \mathbf{S}$ to be nonnegative. Combining Equations (2), (9), and (11) together, the overall optimization problem is formulated as:

$$\begin{aligned} \min_{\mathbf{U}_1, \mathbf{V}_1, \mathbf{U}_2, \mathbf{V}_2, \mathbf{M}} \quad & J(\mathbf{U}_1, \mathbf{V}_1, \mathbf{U}_2, \mathbf{V}_2, \mathbf{M}) = J_1 + J_2 + J_3 \\ \text{s.t} \quad & \mathbf{U}_1, \mathbf{U}_2, \mathbf{V}_1, \mathbf{V}_2, \mathbf{M} \geq \mathbf{0} \end{aligned} \tag{12}$$

4 PROPOSED OPTIMIZATION ALGORITHM

In this section, we first present the proposed algorithm to solve the optimization problem in Equation (12). Then, we analyze the proposed algorithm in terms of the correctness, the convergence and the complexity.

4.1 Optimization Algorithm

Since the overall objective function Equation (12) is not jointly convex, we optimize it by block coordinate descent. That is, the objective function is alternatively minimized with respect to one variable group (e.g., $\mathbf{U}_1$) while fixing the others once at a time. For the sake of conciseness, we only show the minimization procedures over $\mathbf{U}_1$ and $\mathbf{M}$ in this section. Other variables such as $\mathbf{V}_1, \mathbf{U}_2, \mathbf{V}_2$ can be solved in a similar way and we put the details in Appendix A.

First, we show the update algorithm over $\mathbf{U}_1$. The gradient of Equation (2) with respect to $\mathbf{U}_1$ is computed by:

$$\frac{\partial J_1}{\partial \mathbf{U}_1} = \mathbf{X}_1 - \mathbf{Y}_1 \tag{13}$$

where

$$\begin{aligned} \mathbf{X}_1 &= [\mathbf{P}_{\Omega_1} \odot (\mathbf{U}_1\mathbf{V}_1^T)]\mathbf{V}_1 + \lambda\mathbf{U}_1 \\ \mathbf{Y}_1 &= (\mathbf{P}_{\Omega_1} \odot \mathbf{A}_1)\mathbf{V}_1 \end{aligned}$$

and $\mathbf{P}_{\Omega_1}(i, j) = 1$ for $(i, j) \in \Omega_1$, otherwise $\mathbf{P}_{\Omega_1}(i, j) = 0$.

As for Equation (9), note that $\mathbf{D}_1 = \text{diag}(\mathbf{U}_1\mathbf{V}_1^T\mathbf{1}_1)$ and $\hat{\mathbf{D}}_1 = \text{diag}(\mathbf{1}_1^T\mathbf{U}_1\mathbf{V}_1^T)$ are also in terms of $\mathbf{U}_1$, thus the partial gradient is computed by:

$$\frac{\partial J_2}{\partial \mathbf{U}_1} = \mathbf{X}_2 - \mathbf{Y}_2 \tag{14}$$

where

$$\mathbf{X}_2 = \frac{\gamma}{2}[(\mathbf{U}_1\mathbf{M}^T\mathbf{U}_2^T\mathbf{D}_2\mathbf{U}_2\mathbf{M}) \odot \mathbf{U}_1]\mathbf{1}_{r_1}\mathbf{1}_1^T\mathbf{V}_1 + \frac{\gamma}{2}\mathbf{1}_1\mathbf{1}_{r_1}^T[(\mathbf{M}^T\mathbf{U}_2^T\hat{\mathbf{D}}_2\mathbf{U}_2\mathbf{M}\mathbf{U}_1^T) \odot \mathbf{U}_1^T]\mathbf{V}_1$$
$$+ \gamma(\mathbf{D}_1\mathbf{U}_1\mathbf{M}^T\mathbf{U}_2^T\mathbf{D}_2\mathbf{U}_2\mathbf{M} + \hat{\mathbf{D}}_1\mathbf{U}_1\mathbf{M}^T\mathbf{U}_2^T\hat{\mathbf{D}}_2\mathbf{U}_2\mathbf{M})$$
$$\mathbf{Y}_2 = \gamma\mathbf{V}_1\mathbf{U}_1^T\mathbf{U}_1\mathbf{M}^T\mathbf{U}_2^T\mathbf{U}_2\mathbf{V}_2^T\mathbf{U}_2\mathbf{M} + \gamma\mathbf{U}_1\mathbf{M}^T\mathbf{U}_2^T\mathbf{U}_2\mathbf{V}_2^T\mathbf{U}_2\mathbf{M}\mathbf{U}_1^T\mathbf{V}_1 + \gamma\mathbf{U}_1\mathbf{V}_1^T\mathbf{U}_1\mathbf{M}^T\mathbf{U}_2^T\mathbf{V}_2\mathbf{U}_2^T\mathbf{U}_2\mathbf{M}$$

And the gradient of Equation (11) over $\mathbf{U}_1$ is

$$\frac{\partial J_3}{\partial \mathbf{U}_1} = \mathbf{X}_3 - \mathbf{Y}_3 \tag{15}$$

where

$$\mathbf{X}_3 = 2\beta[\mathbf{P}_{\bar{\Omega}_1} \odot (\mathbf{U}_1\mathbf{M}^T\mathbf{U}_2^T\mathbf{A}_2\mathbf{U}_2\mathbf{M}\mathbf{U}_1^T)]\mathbf{U}_1\mathbf{M}^T\mathbf{U}_2^T\mathbf{A}_2\mathbf{U}_2\mathbf{M} + \beta[\mathbf{P}_{\bar{\Omega}_1} \odot (\mathbf{U}_1\mathbf{V}_1^T)]\mathbf{V}_1$$
$$+ 2\beta\mathbf{A}_1\mathbf{U}_1\mathbf{M}^T\mathbf{U}_2^T[\mathbf{P}_{\bar{\Omega}_2} \odot (\mathbf{U}_2\mathbf{M}\mathbf{U}_1^T\mathbf{A}_1\mathbf{U}_1\mathbf{M}^T\mathbf{U}_2^T)]\mathbf{U}_2\mathbf{M}$$
$$\mathbf{Y}_3 = \beta[\mathbf{P}_{\bar{\Omega}_1} \odot (\mathbf{U}_1\mathbf{M}^T\mathbf{U}_2^T\mathbf{A}_2\mathbf{U}_2\mathbf{M}\mathbf{U}_1^T)]\mathbf{V}_1 + \beta[\mathbf{P}_{\bar{\Omega}_1} \odot (\mathbf{U}_1\mathbf{V}_1^T + \mathbf{V}_1\mathbf{U}_1^T)]\mathbf{U}_1\mathbf{M}^T\mathbf{U}_2^T\mathbf{A}_2\mathbf{U}_2\mathbf{M}$$
$$+ \beta\mathbf{A}_1\mathbf{U}_1\mathbf{M}^T\mathbf{U}_2^T[\mathbf{P}_{\bar{\Omega}_2} \odot (\mathbf{U}_2\mathbf{V}_2^T + \mathbf{V}_2\mathbf{U}_2^T)]\mathbf{U}_2\mathbf{M}$$

and matrix $\mathbf{P}_{\bar{\Omega}_2}(i,j) = 1$ for any $(i,j) \notin \Omega_2$.

A fixed-point solution of $\frac{\partial J}{\partial \mathbf{U}_1} = \mathbf{0}$ under the non-negativity constraint of $\mathbf{U}_1$ leads to the following multiplicative update rule:

$$\mathbf{U}_1(p,q) \leftarrow \mathbf{U}_1(p,q)\sqrt[4]{\frac{\mathbf{Y}_1(p,q) + \mathbf{Y}_2(p,q) + \mathbf{Y}_3(p,q)}{\mathbf{X}_1(p,q) + \mathbf{X}_2(p,q) + \mathbf{X}_3(p,q)}} \tag{16}$$

Second, the optimization algorithm over $\mathbf{M}$ is given as below. The gradient of Equation (9) w.r.t $\mathbf{M}$ can be derived as:

$$\frac{\partial J_2}{\mathbf{M}} = \mathbf{X}_4 - \mathbf{Y}_4 \tag{17}$$

where

$$\mathbf{X}_4 = \gamma\mathbf{U}_2^T\mathbf{D}_2\mathbf{U}_2\mathbf{M}\mathbf{U}_1^T\mathbf{D}_1\mathbf{U}_1 + \gamma\mathbf{U}_2^T\hat{\mathbf{D}}_2\mathbf{U}_2\mathbf{M}\mathbf{U}_1^T\hat{\mathbf{D}}_1\mathbf{U}_1$$
$$\mathbf{Y}_4 = \gamma\mathbf{U}_2^T\mathbf{U}_2\mathbf{V}_2^T\mathbf{U}_2\mathbf{M}\mathbf{U}_1^T\mathbf{V}_1\mathbf{U}_1^T\mathbf{U}_1 + \gamma\mathbf{U}_2^T\mathbf{V}_2\mathbf{U}_2^T\mathbf{U}_2\mathbf{M}\mathbf{U}_1^T\mathbf{U}_1\mathbf{V}_1^T\mathbf{U}_1$$

And the gradient of Equation (11) w.r.t $\mathbf{M}$ is computed by:

$$\frac{\partial J_3}{\mathbf{M}} = \mathbf{X}_5 - \mathbf{Y}_5 \tag{18}$$

where

$$\mathbf{X}_5 = \beta\mathbf{U}_2^T\mathbf{A}_2\mathbf{U}_2\mathbf{M}\mathbf{U}_1^T[\mathbf{P}_{\bar{\Omega}_1} \odot (\mathbf{U}_1\mathbf{V}_1^T + \mathbf{V}_1\mathbf{U}_1^T)]\mathbf{U}_1$$
$$+ \beta\mathbf{U}_2^T[\mathbf{P}_{\bar{\Omega}_2} \odot (\mathbf{U}_2\mathbf{V}_2^T + \mathbf{V}_2\mathbf{U}_2^T)]\mathbf{U}_2\mathbf{M}\mathbf{U}_1^T\mathbf{U}_1^T\mathbf{A}_1\mathbf{U}_1$$
$$\mathbf{Y}_5 = 2\beta\mathbf{U}_2^T\mathbf{A}_2\mathbf{U}_2\mathbf{M}\mathbf{U}_1^T[\mathbf{P}_{\bar{\Omega}_1} \odot (\mathbf{U}_1\mathbf{M}^T\mathbf{U}_2^T\mathbf{A}_2\mathbf{U}_2\mathbf{M}\mathbf{U}_1^T)]\mathbf{U}_1$$
$$+ 2\beta\mathbf{U}_2^T[\mathbf{P}_{\bar{\Omega}_2} \odot (\mathbf{U}_2\mathbf{M}\mathbf{U}_1^T\mathbf{A}_1\mathbf{U}_1\mathbf{M}^T\mathbf{U}_2^T)]\mathbf{U}_2\mathbf{M}\mathbf{U}_1^T\mathbf{A}_1\mathbf{U}_1$$

Consequently, the fixed-point solution of $\frac{\partial J}{\partial \mathbf{M}} = \mathbf{0}$ under the nonnegative constraint leads to the following update rule:

$$\mathbf{M}(p,q) \leftarrow \mathbf{M}(p,q)\sqrt[4]{\frac{\mathbf{Y}_4(p,q) + \mathbf{Y}_5(p,q)}{\mathbf{X}_4(p,q) + \mathbf{X}_5(p,q)}} \tag{19}$$

Initialization. Since the optimization problem in Equation (12) is not a joint convex problem, a good initialization of each variable group could play an important role of obtaining a good final

ALGORITHM 1: iNeAt: Incomplete Network Alignment.

Input: (1) the adjacency matrices $\mathbf{A}_1$, $\mathbf{A}_2$ of the incomplete networks $\mathcal{G}_1$, $\mathcal{G}_2$,(2) the optional prior alignment preference $\mathbf{H}$, (3) the rank sizes r_1, r_2,(3) the parameters $\alpha, \lambda, \gamma, \beta$.and (4) the maximum iteration number $t_{\max}$.

Output: (1) the alignment matrix $\mathbf{S}$ between $\mathcal{G}_1$ and $\mathcal{G}_2$, and (2) the complete adjacency matrices $\mathbf{A}_1^*, \mathbf{A}_2^*$.

1: Initialize $\mathbf{U}_1, \mathbf{V}_1, \mathbf{U}_2, \mathbf{V}_2$ by Equation (20), $\mathbf{M}$ by Equation (21), $t = 1$;
2: **while** not converge and $t \le t_{\max}$ **do**
3: Update $\mathbf{U}_1$ by Equation (16);
4: Update $\mathbf{V}_1$ by Equation (27);
5: Update $\mathbf{U}_2$ by Equation (28);
6: Update $\mathbf{V}_2$ by Equation (29);
7: Update $\mathbf{M}$ by Equation (19);
8: Set $t \leftarrow t + 1$;
9: **end while**
10: $\mathbf{A}_1^* = \mathbf{U}_1\mathbf{V}_1^T$ and $\mathbf{A}_2^* = \mathbf{U}_2\mathbf{V}_2^T$.
11: $\mathbf{S} = \alpha\mathbf{U}_2\mathbf{M}\mathbf{U}_1^T + (1-\alpha)\mathbf{H}$.

solution. For $\mathbf{U}_1$ and $\mathbf{U}_2$, we initialize them by solving the symmetric nonnegative matrix factorization of $\mathbf{A}_1$ and $\mathbf{A}_2$, e.g., minimizing $\|\mathbf{A}_1 - \mathbf{U}_1\mathbf{U}_1^T\|_F^2$ over $\mathbf{U}_1 \ge \mathbf{0}$. Same as [9], we use the following multiplicative update rule to obtain the solution:

$$\mathbf{U}_1 \leftarrow \mathbf{U}_1 \odot \left[1 - \epsilon + \epsilon \frac{\mathbf{A}_1\mathbf{U}_1}{\mathbf{U}_1(\mathbf{U}_1^T\mathbf{U}_1)}\right] \tag{20}$$

where ϵ is suggested to be set to 0.5 in practice. Then we set $\mathbf{V}_1 = \mathbf{U}_1$ due to the symmetry of $\mathbf{A}_1$ and initialize $\mathbf{U}_2, \mathbf{V}_2$ similarly. As for the variable $\mathbf{M}$, given the initial $\mathbf{U}_1 = \mathbf{V}_1, \mathbf{U}_2 = \mathbf{V}_2$, we can simplify the computation of Equation (8) and initialize $\mathbf{M}$ as:

$$\mathbf{M} = (1-\alpha)\sum_{k=0}^{K} \alpha^{k+1}(\mathbf{U}_2^T\mathbf{D}_2^{-1}\mathbf{U}_2)^k\mathbf{U}_2^T\mathbf{D}_2^{-1}\mathbf{H}\mathbf{D}_1^{-1}\mathbf{U}_1(\mathbf{U}_1^T\mathbf{D}_1^{-1}\mathbf{U}_1)^k \tag{21}$$

where the constant K can be set to a relatively large number, e.g., 100.

Overall, the proposed algorithm is summarized in Algorithm 1. First, it initializes each variable as line 1. Then, the algorithm alternatively updates each variable group one by one (line 3–7) until it converges or the maximum iteration number t_{max} is reached. The algorithm finally outputs the complete networks $\mathbf{A}_1^*, \mathbf{A}_2^*$ (line 10), and the alignment matrix $\mathbf{S}$ as line 11.

4.2 Proof and Analysis

In this subsection, we provide the theoretical analysis of the updating rule of $\mathbf{U}_1$. We first prove that the fixed-point solution of Equation (16) satisfies the Karush–Kuhn–Tucker (KKT) condition. Then we analyze its convergence, as well as its time and space complexity. The analyses and proofs for other variables are similar and are omitted in the article for brevity.

Theorem 1 [Correctness of Equation (16)]. *At convergence, the fixed-point solution of Equation (16) satisfies the KKT condition.*

Proof. Let $\Sigma \in \mathbb{R}^{n_1 \times r_1}$ be the Lagrangian multiplier and the Lagrangian function of Equation (12) be:

$$L(\mathbf{U}_1) = J(\mathbf{U}_1) - \mathrm{Tr}(\Sigma^T\mathbf{U}_1)$$

By setting the gradient of L w.r.t $\mathbf{U}_1$ to 0, we obtain:

$$\Sigma = \mathbf{X}_1 + \mathbf{X}_2 + \mathbf{X}_3 - \mathbf{Y}_1 - \mathbf{Y}_2 - \mathbf{Y}_3 \tag{22}$$

The KKT complementary condition for the nonnegativity of $\mathbf{U}_1$ gives:

$$(\mathbf{X}_1 + \mathbf{X}_2 + \mathbf{X}_3 - \mathbf{Y}_1 - \mathbf{Y}_2 - \mathbf{Y}_3) \odot \mathbf{U}_1 = \mathbf{0} \tag{23}$$

According to the updating rule Equation (16), at convergence, we have for $\forall p, q$,

$$\mathbf{U}_1(p,q) = \mathbf{U}_1(p,q)\sqrt[4]{\frac{\mathbf{Y}_1 + \mathbf{Y}_2 + \mathbf{Y}_3}{\mathbf{X}_1 + \mathbf{X}_2 + \mathbf{X}_3}} \tag{24}$$

which is equivalent to:

$$(\mathbf{X}_1 + \mathbf{X}_2 + \mathbf{X}_3 - \mathbf{Y}_1 - \mathbf{Y}_2 - \mathbf{Y}_3) \odot (\mathbf{U}_1)^4 = \mathbf{0} \tag{25}$$

Equations (23) and (25) are equivalent, so at convergence, Equation (16) satisfies the KKT condition. □

Then, we show the convergence of updating $\mathbf{U}_1$ under Equation (16). First, the following lemma gives the auxiliary function for the objective function Equation (12) w.r.t $\mathbf{U}_1$.

Lemma 2 [Auxiliary function of $J(\mathbf{U}_1)$]. *Let $J(\mathbf{U}_1)$ denote all the terms in Equation (12) that contains $\mathbf{U}_1$, then the following function $Z(\mathbf{U}_1, \tilde{\mathbf{U}}_1)$*

$$\begin{aligned}
Z(\mathbf{U}_1, \tilde{\mathbf{U}}_1) = {} & \underbrace{\frac{1}{4}\sum_{p,q}[(\mathbf{P}_{\Omega_1} \odot (\tilde{\mathbf{U}}_1\mathbf{V}_1^T))\mathbf{V}_1](p,q)\frac{\mathbf{U}_1^4(p,q) + \tilde{\mathbf{U}}_1^4(p,q)}{\tilde{\mathbf{U}}_1^3(p,q)}}_{T_1'} \\
& \underbrace{- \sum_{p,q}[(\mathbf{P}_{\Omega_1} \odot \mathbf{A}_1)\mathbf{V}_1](p,q)\tilde{\mathbf{U}}_1(p,q)(1 + \log\frac{\mathbf{U}_1(p,q)}{\tilde{\mathbf{U}}_1(p,q)})}_{T_2'} \\
& + \underbrace{\frac{\lambda}{4}\sum_{p,q}\tilde{\mathbf{U}}_1(p,q)\frac{\mathbf{U}_1^4(p,q) + \tilde{\mathbf{U}}_1^4(p,q)}{\tilde{\mathbf{U}}_1^3(p,q)}}_{T_3'} + \underbrace{\frac{\gamma}{12}\sum_{p,q}\mathbf{Z}_1(p,q)\frac{3\mathbf{U}_1^4(p,q) + \tilde{\mathbf{U}}_1^4(p,q)}{\tilde{\mathbf{U}}_1^3(p,q)}}_{T_4'} \\
& + \gamma T_5' + \underbrace{\frac{\beta}{4}\sum_{p,q}[(\mathbf{P}_{\tilde{\Omega}_1} \odot (\tilde{\mathbf{U}}_1\mathbf{V}_1^T))\mathbf{V}_1](p,q)\frac{\mathbf{U}_1^4(p,q) + \tilde{\mathbf{U}}_1^4(p,q)}{\tilde{\mathbf{U}}_1^3(p,q)}}_{T_6'} \\
& + \underbrace{\frac{\beta}{2}\sum_{p,q}\mathbf{Z}_2(p,q)\frac{\mathbf{U}_1^4(p,q)}{\tilde{\mathbf{U}}_1^3(p,q)}}_{T_7'} + \underbrace{\frac{\beta}{2}\sum_{p,q}\mathbf{Z}_3(p,q)\frac{\mathbf{U}_1^4(p,q)}{\tilde{\mathbf{U}}_1^3(p,q)}}_{T_8'} + \beta T_9' + \beta T_{10}'
\end{aligned}$$

where

$$\begin{aligned}
\mathbf{Z}_1 &= \frac{1}{2}[(\tilde{\mathbf{U}}_1\mathbf{M}^T\mathbf{U}_2^T\mathbf{D}_2\mathbf{U}_2\mathbf{M}) \odot \tilde{\mathbf{U}}_1]\mathbf{1}_{r_1}\mathbf{1}_1^T\mathbf{V}_1 + \frac{1}{2}\mathbf{1}_1\mathbf{1}_{r_1}^T[(\mathbf{M}^T\mathbf{U}_2^T\hat{\mathbf{D}}_2\mathbf{U}_2\mathbf{M}\tilde{\mathbf{U}}_1) \odot \tilde{\mathbf{U}}_1]\mathbf{V}_1 \\
&\quad + diag(\tilde{\mathbf{U}}_1\mathbf{V}_1^T\mathbf{1}_1)\tilde{\mathbf{U}}_1\mathbf{M}^T\mathbf{U}_2^T\mathbf{D}_2\mathbf{U}_2\mathbf{M} + diag(\mathbf{V}_1\mathbf{U}_1^T\mathbf{1}_1)\tilde{\mathbf{U}}_1\mathbf{M}^T\mathbf{U}_2^T\hat{\mathbf{D}}_2\mathbf{U}_2\mathbf{M} \\
\mathbf{Z}_2 &= [\mathbf{P}_{\tilde{\Omega}_1} \odot (\tilde{\mathbf{U}}_1\mathbf{M}^T\mathbf{U}_2^T\mathbf{A}_2\mathbf{U}_2\mathbf{M}\tilde{\mathbf{U}}_1^T)]\tilde{\mathbf{U}}_1\mathbf{M}^T\mathbf{U}_2^T\mathbf{A}_2\mathbf{U}_2\mathbf{M} \\
\mathbf{Z}_3 &= \mathbf{A}_1\tilde{\mathbf{U}}_1\mathbf{M}^T\mathbf{U}_2^T[\mathbf{P}_{\tilde{\Omega}_2} \odot (\mathbf{U}_2\mathbf{M}\tilde{\mathbf{U}}_1^T\mathbf{A}_1\tilde{\mathbf{U}}_1\mathbf{M}^T\mathbf{U}_2^T]\mathbf{U}_2\mathbf{M} \\
T_5' &= -\sum_{o,p,q,r,s}(\mathbf{M}^T\mathbf{U}_2^T\mathbf{U}_2\mathbf{V}_2^T\mathbf{U}_2\mathbf{M})(o,q)\mathbf{V}_1(p,r)\tilde{\mathbf{U}}_1(s,r)\tilde{\mathbf{U}}_1(s,o) \times \tilde{\mathbf{U}}_1(p,q) \\
&\quad (1 + \log\frac{\mathbf{U}_1(p,q)\mathbf{U}_1(s,r)\mathbf{U}_1(s,o)}{\tilde{\mathbf{U}}_1(p,q)\tilde{\mathbf{U}}_1(s,r)\tilde{\mathbf{U}}_1(s,o)}) \\
T_9' &= -\sum_{o,p,q,r,s}\mathbf{P}_{\tilde{\Omega}_1}(p,o)\mathbf{V}_1(o,q)(\mathbf{M}^T\mathbf{U}_2^T\mathbf{A}_2\mathbf{U}_2\mathbf{M})(s,r)\tilde{\mathbf{U}}_1(o,s) \times \tilde{\mathbf{U}}_1(p,r)\tilde{\mathbf{U}}_1(p,q) \\
&\quad (1 + \log\frac{\mathbf{U}_1(p,q)\mathbf{U}_1(o,s)\mathbf{U}_1(p,r)}{\tilde{\mathbf{U}}_1(p,q)\tilde{\mathbf{U}}_1(o,s)\tilde{\mathbf{U}}_1(p,r)}) \\
T_{10}' &= -\sum_{o,p,q,r,s,t}\mathbf{P}_{\tilde{\Omega}_2}(p,q)(\mathbf{U}_2\mathbf{V}_2^T)(p,q)(\mathbf{U}_2\mathbf{M})(p,r)\tilde{\mathbf{U}}_1(s,r)\mathbf{A}_1(s,t)\tilde{\mathbf{U}}_1(t,o)(\mathbf{M}^T\mathbf{U}_2^T)(o,q) \\
&\quad \times (1 + \log\frac{\mathbf{U}_1(s,r)\mathbf{U}_1(t,o)}{\tilde{\mathbf{U}}_1(s,r)\tilde{\mathbf{U}}_1(t,o)})
\end{aligned}$$

is an auxiliary function of $J(\mathbf{U}_1)$ *for any* $\mathbf{U}_1, \tilde{\mathbf{U}}_1 \geq \mathbf{0}$ *after removing some constant terms such that* $Z(\mathbf{U}_1, \tilde{\mathbf{U}}_1) \geq J(\mathbf{U}_1)$ *and* $Z(\mathbf{U}_1, \mathbf{U}_1) = J(\mathbf{U}_1)$. *And it is also a convex function w.r.t* $\mathbf{U}_1$ *and its global minima is*

$$\mathbf{U}_1(p,q) = \tilde{\mathbf{U}}_1(p,q)\sqrt[4]{\frac{\tilde{\mathbf{Y}}_1(p,q) + \tilde{\mathbf{Y}}_2(p,q) + \tilde{\mathbf{Y}}_3(p,q)}{\tilde{\mathbf{X}}_1(p,q) + \tilde{\mathbf{X}}_2(p,q) + \tilde{\mathbf{X}}_3(p,q)}} \tag{26}$$

where $\tilde{\mathbf{X}}_i, \tilde{\mathbf{Y}}_i, i = 1,2,3$ *are all in terms of* $\tilde{\mathbf{U}}_1$ *while sharing the same formulas with* $\mathbf{X}_i, \mathbf{Y}_i, i = 1,2,3$.

Proof. Refer the proof to Appendix B. □

Next, we show the convergence of updating $\mathbf{U}_1$ by Equation (16) in the following theorem.

Theorem 2 [Convergence of Equation (16)]. *When other variables are fixed, under the updating rule Equation (16), the objective function w.r.t* $\mathbf{U}_1$ *monotonically non-increases.*

Proof. Denote $\mathbf{U}_1$ at iteration t as $\mathbf{U}_1^{(t)}$. According to Lemma 2, the global minima $\mathbf{U}_1^{(t+1)}$ of the auxiliary function is achieved by minimizing the function $Z(\mathbf{U}_1, \mathbf{U}_1^{(t)})$ over $\mathbf{U}_1$, which leads to:

$$Z(\mathbf{U}_1^{(t+1)}, \mathbf{U}_1^{(t)}) \leq Z(\mathbf{U}_1^{(t)}, \mathbf{U}_1^{(t)}) = J(\mathbf{U}_1^{(t)})$$

Besides, based on Lemma 2, $J(\mathbf{U}_1^{(t+1)}) \leq Z(\mathbf{U}_1^{(t+1)}, \mathbf{U}_1^{(t)})$ and therefore $J(\mathbf{U}_1^{(t+1)}) \leq J(\mathbf{U}_1^{(t)})$ which means the objective function w.r.t. $\mathbf{U}_1$ is monotonically non-increasing. □

The time and space complexity of each updating iteration in Algorithm 1 are summarized in Lemma 3. Note that by exploring the low-rank structure of the alignment matrix, the time complexity is reduced to *linear*.

Lemma 3 [Complexity of iNeAt]. *The time complexity of each update iteration in Algorithm 1 is* $O(nr^2 + min\{|\bar{\Omega}|, |\Omega|\}r)$, *and the space complexity is* $O(nr + min\{|\bar{\Omega}|, |\Omega|\})$ *where* $n, |\Omega|, |\bar{\Omega}|$ *are*

Table 2. Statistics of Datasets

Category	Network	# of Nodes	# of Edges
Collaboration	Gr-Qc	5,241	14,484
Infrastructure	Oregon	7,352	15,665
Social	Google+	23,628	39,194
Social	Youtube	1,134,890	2,987,624
Communication	Gordian Channel 1	1,000	41,191
Communication	Gordian Channel 2	1,003	4,627

the number of nodes, the number of observed and unobserved entries in two incomplete networks respectively. And r denotes the rank of networks.

Proof. For time complexity, calculating the term $\mathbf{X}_1, \mathbf{Y}_1, \mathbf{X}_3$, and $\mathbf{Y}_3$ in each iteration has $O(nr^2 + mr + \min\{|\bar{\Omega}|, |\Omega|\}r)$ time complexity and $O(m + nr)$ space complexity. Note that $\mathbf{P}_{\Omega_1} + \mathbf{P}_{\bar{\Omega}_1} = \mathbf{1}_{n_1 \times n_1}$. In this way, for example, $\mathbf{X}_1$ can be computed from either $\mathbf{P}_{\Omega_1}$ or $\mathbf{P}_{\bar{\Omega}_1}$. Thus, we use $\min\{|\bar{\Omega}|, |\Omega|\}$ for the complexity analysis. For terms $\mathbf{X}_2$ and $\mathbf{Y}_2$, it takes $O(nr^2)$ time complexity and $O(nr)$ space complexity. For other variables $\mathbf{V}_1, \mathbf{U}_2, \mathbf{V}_2$, and $\mathbf{M}$, since the analyses are similar, we omit the analyses for brevity. Overall, we can obtain the time and space complexity in the above lemma. □

We remark that the linear complexity is obtained in each updating iteration of Algorithm 1. If we carry out line 10–11 in a straightforward way, it will incur an additional $O(n^2)$ cost due to the multiplications between low-rank matrices (e.g., $\mathbf{U}_1\mathbf{V}_1^T$ and $\mathbf{U}_2\mathbf{M}\mathbf{U}_1$) as well as the need to store the potentially dense matrices (e.g., $\mathbf{A}_1^*$, $\mathbf{S}$). To address this issue, we can store the resulting $\mathbf{A}_1^*$, $\mathbf{A}_2^*$, and $\mathbf{S}$ in a compact way by the corresponding low-rank matrices. Then when we access a certain entry of the matrix (e.g., $\mathbf{A}_1^*$), we perform the vector-vector inner product between the corresponding rows of $\mathbf{U}_1$ and $\mathbf{V}_1$.

5 EXPERIMENTAL RESULTS

In this section, we present the experimental results of the proposed algorithm iNeAt. We evaluate our algorithm in the following two aspects:

— *Effectiveness*: How accurate is our algorithm for aligning incomplete networks? How effective is our algorithm to recover missing edges by leveraging the alignment result?
— *Efficiency*: How fast and scalable is our algorithm?

5.1 Experimental Setup

Datasets. We evaluate the proposed algorithm on three types of real-world networks, including the collaboration network, infrastructure network and social networks. The statistics of all the datasets are summarized in Table 2.

— *Collaboration Network*: We use the collaboration network in the general relativity and quantum cosmology (*Gr-Qc*) area from the e-print arXiv [18]. In the network, each node represents an author and there exists an edge if two authors have co-authored at least one paper.
— *Infrastructure Network*: This dataset is a network of Autonomous Systems (AS) inferred from Oregon route-view [18]. In the network, nodes are the routers, and edges represent the peering information among routers.

— *Social Network*: We use the social network collected from Google+ [19]. In the network, nodes are the users and an edge denotes that one user has the other user in his/her circles. We also use the Youtube network [39] where nodes are the Youtube users and edges represent the friendship among users.
— *Gordian Networks*: This dataset contains communication networks via different channels. In particular, we aim to align the communication networks via phone (Channel 1) and e-mails (Channel 2). Each node in both networks represents a person. An edge in Channel 1 network indicates two people contact each other through phone whereas each edge in Channel 2 network represents two people send an email. There are 1,000 common nodes in both networks that are used as the alignment ground-truth.

Based on these datasets except Gordian dataset, we construct four pairs of incomplete networks for alignment evaluations by the following steps. For each dataset, we first generate a random permutation matrix and use it to construct the second (permuted) network. Then, in each of these two networks, we remove 0.1%, 0.5%, 1%, 5%, 10%, 15%, 20% of the total number of edges uniformly at random to generate the unobserved edges. For the Gordian dataset (Channel 1 and Channel 2), we first compute the edge betweenness score for each edge, which is sum of the fraction of all pairs of shortest paths through the edge [6]. Then we normalize the edge betweenness scores such that they sum to 1, i.e., $\sum_{(u,v)\in\mathcal{G}_1} s_b((u,v)) = 1$ where $s_b((u,v))$ is the edge betweenness score of edge (u,v) and then we use the normalized scores as the probabilities to remove the corresponding edge as an unobserved edge. We run our algorithm and other comparison methods in all the pairs of incomplete networks.

Comparison Methods.

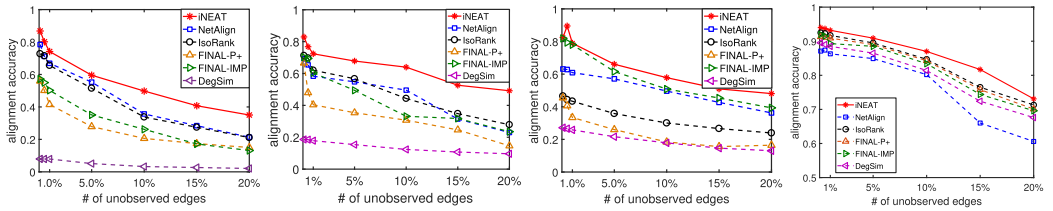
— *Alignment.* To evaluate the alignment performance of our proposed algorithm INEAT[3], we compare it with the following existing network alignment algorithms, including (1) *NetAlign* [3], (2) *IsoRank* [34], and (3) *FINAL-P+* [43]. Besides, in order to validate whether alignment and imputation are mutually beneficial from each other, we use the low-rank networks completed solely by Equation (2) as the input networks for *FINAL-P+*. We name this method as *FINAL-IMP*. We also show the alignment results by the degree similarity (*DegSim*), which is also used as the prior knowledge matrix $\mathbf{H}$ of INEAT.
— *Completion.* To evaluate the completion performance, we compare our algorithm with the existing matrix completion methods which are for the single network completion task, including (1) a matrix factorization method based on Equation (2) (*NMF-IMP*), (2) an accelerated proximal gradient based nuclear norm minimization method (*NNLS*) [36], and (3) a Riemannian trust-region based matrix completion method (*RTRMC*) [5].

Machines and Repeatability. All experiments are performed on a Windows machine with four 3.6 GHz Intel Cores and 32 GB RAM. The algorithms are programmed with MATLAB using a single thread.

5.2 Effectiveness Analysis

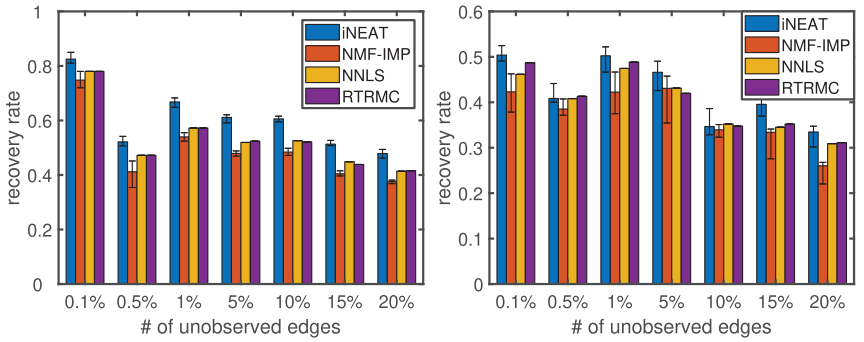
We first evaluate the alignment accuracy with different numbers of unobserved edges in the incomplete networks. We use a heuristic greedy matching algorithm as the post processing step on the alignment matrix to obtain the one-to-one mapping matrix between two input networks, then we compute the alignment accuracy with respect to the ground-truth (i.e., the permutation matrix). The results are summarized in Figure 3. We have the following observations. First, we observe that INEAT outperforms the baseline methods with different numbers of unobserved

[3]The code can be found in http://www.public.asu.edu/~szhan172/ineat.zip.



(a) Gr-Qc collaboration networks. (b) Oregon infrastructure networks. (c) Google+ social networks. (d) Gordian channel 1 vs. channel 2.

Fig. 3. (Higher is better.) Alignment accuracy vs. the number of unobserved edges in the networks.

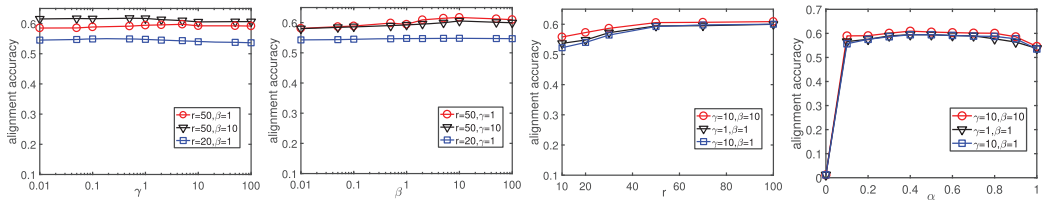


(a) Gr-Qc collaboration network. (b) Google+ social network.

Fig. 4. (Higher is better.) Recovery rate vs. the number of unobserved edges in the networks.

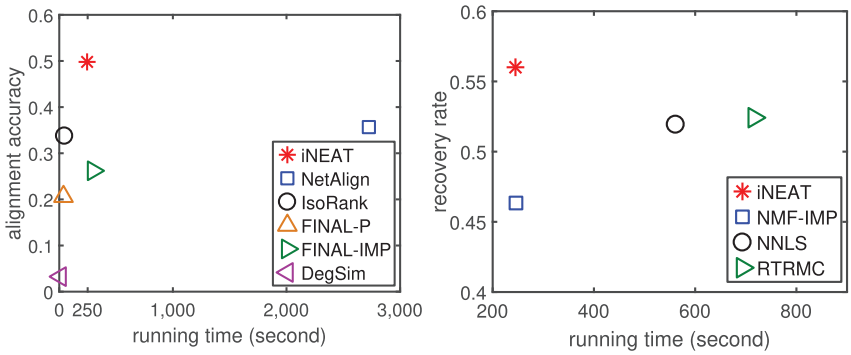
edges. To be specific, our method achieves an up to 30% alignment accuracy improvement, compared with the baseline methods that directly align across two incomplete networks (i.e., *NetAlign*, *IsoRank*, and *FINAL-P+*). Second, the degree similarity (i.e., $\mathbf{H}$) alone gives a very poor performance on the alignment accuracy, whereas by averaging $\mathbf{H}$ and $\mathbf{U}_2\mathbf{M}\mathbf{U}_1$, the alignment matrix (i.e., results of iNeAt) provides a much better accuracy. This verifies the effectiveness of our strategy combining the low-rank structure of alignment matrix and prior knowledge $\mathbf{H}$. Third, the accuracy of iNeAt is higher than that of *FINAL-IMP*, which indicates that solving the alignment and imputation tasks simultaneously indeed achieves a better performance than the *completion-then-alignment* strategy. Specifically, as Figure 3(a) and 3(b) show, in some cases, the pure completion may introduce too much noise in the incomplete networks and hence lead to an even worse alignment result than that of other alignment baseline methods (those without performing network completion at all).

Second, to evaluate the effectiveness of iNeAt for network completion, we assume the missing edges are recovered if the corresponding entries of the completed adjacency matrix are larger than a certain threshold (e.g., set to be 0.3 in our article). Then, we calculate the recovery rate over the total number of missing edges. In addition, as the algorithms of network completion may achieve different performance with different initializations, we repeat the algorithms for 20 runs and present the mean edge recovery rates and variances. The results are shown in Figure 4. As we can see, iNeAt has a higher recovery rate than other baseline methods, indicating that the completion performance is indeed improved by leveraging the alignment across two networks. Besides, the network completion performance of our algorithms are not sensitive to the algorithm initializations. For *NNLS* and *RTRMC* baseline methods, we did not observe any variances.



(a) Alignment accuracy vs. γ. (b) Alignment accuracy vs. β. (c) Alignment accuracy vs. rank r. (d) Alignment accuracy vs. α.

Fig. 5. Parameter study on collaboration networks with 5% unobserved edges: study the effect of the parameters γ, β, rank r and α in terms of alignment accuracy.



(a) Balance between running time and alignment accuracy. (b) Balance between running time and recovery rate.

Fig. 6. Quality-speed results on the collaboration network with 10% unobserved edges.

Third, we study how different parameters affect the alignment accuracy. In our experiments, we mainly study three parameters, including (1) γ which controls the importance of alignment task, (2) β which controls the importance of cross-network completion task, and (3) r which is the rank of the complete network. The results are shown in Figure 5. As we can see, the alignment accuracy is stable within a wide range of parameter settings. Besides, Figure 5(c) suggests that a relatively small rank might be sufficient to achieve a satisfactory alignment performance. We also observe in Figure 5(d) that (1) by leveraging the combination of both $\mathbf{U}_2\mathbf{M}\mathbf{U}_1^T$ and the prior information $\mathbf{H}$ can significantly improve the alignment performance, and (2) $\alpha = 0.5$ leads to better results in most cases.

5.3 Efficiency Analysis

Quality-Speed Trade-off. In order to evaluate the trade-off between the effectiveness and efficiency of our method, we measure the quality from two aspects, including the quality of alignment and that of network completion. Here, we show the trade-off results on the collaboration network with 10% unobserved edges in Figure 6. As we can see in Figure 6(a), the running time of our method iNeAt is slightly higher than *IsoRank* and *FINAL-P+*, but it achieves a 15%–25% alignment accuracy improvement across the incomplete networks. Meanwhile, our method is much faster than *NetAlign*.

On the other hand, to evaluate the quality of network completion, note that the running time is the time for completing two incomplete networks. As Figure 6(b) shows, iNeAt obtains a better

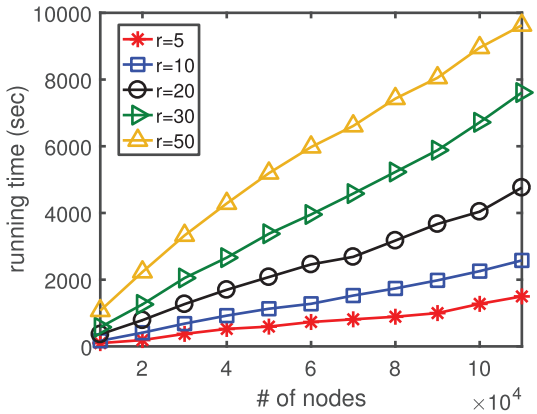


Fig. 7. Running time vs. the number of nodes in the networks.

recovery rate and less running time. To be specific, compared with *NMF-IMP*, iNeAt can recover 10% more missing edges with a similar running time. Besides, iNeAt achieves a slightly better recovery rate and a much faster speed than *NNLS* and *RTRMC*.

Scalability. We use the largest dataset (Youtube) to study the scalability of our proposed method iNeAt (i.e., running time vs. size of the network). Here, we use the same method to extract and construct several pairs of incomplete subgraphs with different sizes from the entire network. As we can see from Figure 7, the running time of the algorithm is *linear* w.r.t the number of nodes in the networks which is consistent with our time complexity analysis.

6 RELATED WORK

Network Alignment. In general, network alignment has two categories, i.e., local alignment and global alignment. Among others, local network alignment aims to uncover the alignment among small regions across multiple networks, such as motifs and small subgraphs. Some recent works include [4, 26, 30]. Nevertheless, local network alignment might be too restrictive to effectively find the node correspondence. On the other side, many global network alignment algorithms that targets to find node alignment, are based on the *topology consistency*. For example, one early well-known approach *IsoRank* computes the cross-network pairwise topology similarities by propagating the similarities of their neighboring node pairs and it is shown that this can be formulated as a random-walk propagation procedure in the Kronecker product graph [34]. In addition, *IsoRankN* [20] extends the original *IsoRank* algorithm by using PageRank-Nibble [1] to align multiple networks. *BigAlign* [16] and *UMA* [42] both assume that one network is a noisy permutation of the other network, whereas [42] is generalized to align multiple networks by adding the *transitivity* constraints. *NetAlign* formulates the network alignment problem as an optimization problem to maximize the number of aligned neighboring node pairs [3] and solve it based on a belief-propagation heuristic.

More recently, Liu et al. propose an algorithm that learns the embedding of the nodes while making the aligned nodes closed to each other in the embedding space [23]. Then the distances among the embedding vectors of nodes are used to measure the probabilities that nodes are aligned. Another recent embedding-based alignment method for attributed networks is proposed in [14]. However, these embedding-based methods implicitly suffer from the space disparity. To address this issue, Du et al. propose an embedding-based method *MrMine* that forces the embedding vectors of different objects of the networks at different resolutions to lie in the same space [11]. In

addition, Zhang et al. propose to mitigate this issue by postprocessing the node embedding vectors by a non-rigid point-set registration [45]. Moreover, Vijayan et al. proposes *MAGNA++* [38] that simultaneously maximizes the node conservation and edge conservation which are widely used in bioinformatics. Other network alignment algorithms proposed in the bioinformatics community include *HubAlign* [13], *Natelie* [12], *L-GRAAL* [25], and the like. However, most of these methods only assume the *global consistency* in the network topology and possibly leverage the attribute information by calculating the attribute similarity matrix as the prior alignment matrix. One potential drawback of these methods is not considering the consistency in the attributes of the networks.

Some early works in another relevant topic (i.e., anchor link prediction), which use both structural information and attribute information to map users across networks, include [15, 40]. A recent work *COSNET* formulates the local consistency among the attributes of each node and the global topology consistency, as well as the transitivity property into an energy-based model to find the alignment across multiple attributed networks [46]. However, these methods are all supervised and require the exact alignment relationships as the training data. On the other side, Zhang et al. propose an attributed network alignment algorithm by adopting both the topological and attribute consistency principles [43]. This work formulates these consistency principles into a convex quadratic problem and propose a fixed-point solution to solve it. Du et al. propose an efficient solver-based on Krylov subspace to accelerate the algorithm [10]. Chen et al. [8] propose a community-based alignment method that can not only leverage the node attributes, but also find both the node-level alignment and the community-level alignment. However, most, if not all, of the existing methods implicitly assume that the input networks are complete without missing edges.

Network Completion. On the other side, since the real-world networks are often incomplete, the network completion task is often the very first step prior to many applications in order to gain a better performance. Kim et al. propose a network completion method based on expectation maximization that can add the missing nodes and edges under the assumption that networks follow the Kronecker graph model [17]. Another work proposed by Masrour et al. decouples the network completion from transduction so that the node similarity matrix can be efficiently leveraged as the side information [27]. Soundarajan et al. study to reduce the incompleteness of the input network by a careful node selection to probe nodes [35]. In addition, inferring the missing edges in the incomplete network can be considered as an adjacency matrix completion problem, and hence the network completion problem can be naturally solved by many matrix completion approaches. Among them, one well-known method is based on singular value thresholding, which iteratively shrinks the singular values to minimize the nuclear norm [7]. In order to speed up the completion process, Toh and Yun propose an accelerated proximal gradient algorithm to solve the nuclear norm regularized linear least squares problem [36]. Besides, by exploiting the geometry of the low-rank structure constraint, a first-order and second-order Riemannian trust-region approach is proposed to solve the formulated optimization problem on the Grassmann manifold [5]. [37] utilizes graph auto-encoder for matrix completion, or specifically for bipartite network completion. However, all these methods aim to complete a single network once at a time. On the other hand, there are some recent work to complete multiple *aligned* networks by tensor completion [22, 24]. Nonetheless, how these input networks are aligned at the first place was not answered in these work.

Another hot topic related to network completion is the link prediction problem [21]. Miller et al. propose a Beysian nonparametric latent feature model to predict the links in the relational data [29]. [28] proposes to use the supervised matrix factorization to obtain the latent features and then predict links by combining latent features with the explicit node and edge features. Barbieri et al.

propose a stochastic topic model which can not only predict the links to be recommended, but also explain why the links are predicted [2]. In [41], Zhang et al. study the link prediction problem across multiple aligned networks and formulate it as a sparse low-rank matrix completion problem. This is different from iNeAt where no alignment are known apriori.

7 CONCLUSION

In the era of big data, the multi-sourced and incomplete characteristics often *co-exist* in many real networks. Nonetheless, the state-of-the-arts has been largely addressing them *in parallel*. In this article, we propose to jointly address network alignment and network completion so that the two tasks can mutually benefit from each other. We formulate incomplete network alignment problem as an optimization problem and propose a multiplicative update algorithm (iNeAt) to solve it. The proposed algorithm is proved to converge to the KKT fixed point with a linear complexity in both time and space. To our best knowledge, the proposed iNeAt algorithm is the first network alignment algorithm with a provable linear complexity. The empirical evaluations demonstrate the effectiveness and efficiency of the proposed iNeAt algorithm. Specially, it (1) improves the alignment accuracy by up to 30% over the existing network alignment methods, in the meanwhile it leads a better imputation outcome; and (2) achieves a good quality-speed balance and scales *linearly* w.r.t the number of nodes in the networks. Future work includes extending our algorithm to handle attributed networks and other ways to leverage the prior knowledge.

APPENDIX

A UPDATE RULE OF $\mathbf{V_1}, \mathbf{U_2}, \mathbf{V_2}$

Similarly, the gradient of Equation (2) in terms of $\mathbf{V}_1$ is computed by $\frac{\partial J_1}{\partial \mathbf{V}_1} = \mathbf{X}_6 - \mathbf{Y}_6$ where:

$$\mathbf{X}_6 = [\mathbf{P}_{\Omega_1} \odot (\mathbf{V}_1\mathbf{U}_1^T)]\mathbf{U}_1 + \lambda\mathbf{V}_1$$
$$\mathbf{Y}_6 = (\mathbf{P}_{\Omega_1} \odot \mathbf{A}_1)\mathbf{U}_1$$

The partial gradient of Equation (9) w.r.t. $\mathbf{V}_1$ is computed by $\frac{\partial J_2}{\partial \mathbf{V}_1} = \mathbf{X}_7 - \mathbf{Y}_7$ where:

$$\mathbf{X}_7 = \frac{\gamma}{2}\mathbf{1}_1\mathbf{1}_{r_1}^T[(\mathbf{M}^T\mathbf{U}_2^T\mathbf{D}_2\mathbf{U}_2\mathbf{M}\mathbf{U}_1^T) \odot \mathbf{U}_1^T]\mathbf{U}_1 + \frac{\gamma}{2}[(\mathbf{U}_1\mathbf{M}^T\mathbf{U}_2^T\hat{\mathbf{D}}_2\mathbf{U}_2\mathbf{M}) \odot \mathbf{U}_1]\mathbf{1}_{r_1}\mathbf{1}_1^T\mathbf{U}_1$$
$$\mathbf{Y}_7 = \gamma\mathbf{U}_1\mathbf{M}^T\mathbf{U}_2^T\mathbf{V}_2\mathbf{U}_2^T\mathbf{U}_2\mathbf{M}$$

And $\frac{\partial J_3}{\partial \mathbf{V}_1}$ is computed by $\frac{\partial J_3}{\partial \mathbf{V}_1} = \mathbf{X}_8 - \mathbf{Y}_8$ where:

$$\mathbf{X}_8 = \beta[\mathbf{P}_{\bar{\Omega}_1} \odot (\mathbf{V}_1\mathbf{U}_1^T)]\mathbf{U}_1$$
$$\mathbf{Y}_8 = \beta[\mathbf{P}_{\bar{\Omega}_1} \odot (\mathbf{U}_1\mathbf{M}^T\mathbf{U}_2^T\mathbf{A}_2\mathbf{U}_2\mathbf{M}\mathbf{U}_1^T)]\mathbf{U}_1$$

The fixed-point solution of $\frac{\partial J}{\mathbf{V}_1} = \mathbf{0}$ under the nonnegativity constraint of $\mathbf{V}_1$ leads to the following multiplicative update rule:

$$\mathbf{V}_1(p,q) \leftarrow \mathbf{V}_1(p,q)\sqrt[4]{\frac{\mathbf{Y}_6(p,q) + \mathbf{Y}_7(p,q) + \mathbf{Y}_8(p,q)}{\mathbf{X}_6(p,q) + \mathbf{X}_7(p,q) + \mathbf{X}_8(p,q)}} \tag{27}$$

The gradient of Equation (2) w.r.t. $\mathbf{U}_2$ is computed by $\frac{\partial J_1}{\partial \mathbf{U}_2} = \mathbf{X}_9 - \mathbf{Y}_9$ where:

$$\mathbf{X}_9 = [\mathbf{P}_{\Omega_1} \odot (\mathbf{U}_2\mathbf{V}_2^T)]\mathbf{V}_2 + \lambda\mathbf{U}_2$$
$$\mathbf{Y}_9 = (\mathbf{P}_{\Omega_1} \odot \mathbf{A}_2)\mathbf{V}_2$$

For Equation (9), its gradient over $\mathbf{U}_2$ can be derived as $\frac{\partial J_2}{\partial \mathbf{U}_2} = \mathbf{X}_{10} - \mathbf{Y}_{10}$ where:

$$\begin{aligned}\mathbf{X}_{10} = {} & \frac{\gamma}{2}[\mathbf{U}_2\mathbf{M}\mathbf{U}_1^T\mathbf{D}_1\mathbf{U}_1\mathbf{M}^T) \odot \mathbf{U}_2]\mathbf{1}_{r_2}\mathbf{1}_2^T\mathbf{V}_2 + \frac{\gamma}{2}\mathbf{1}_2\mathbf{1}_{r_2}^T[(\mathbf{M}\mathbf{U}_1^T\hat{\mathbf{D}}_1\mathbf{U}_1\mathbf{M}^T\mathbf{U}_2^T) \odot \mathbf{U}_2^T] \\ & + \gamma(\gamma\mathbf{D}_2\mathbf{U}_2\mathbf{M}\mathbf{U}_1^T\mathbf{D}_1\mathbf{U}_1\mathbf{M}^T + \hat{\mathbf{D}}_2\mathbf{U}_2\mathbf{M}\mathbf{U}_1^T\hat{\mathbf{D}}_1\mathbf{U}_1\mathbf{M}^T) \\ \mathbf{Y}_{10} = {} & \gamma\mathbf{U}_2\mathbf{M}\mathbf{U}_1^T\mathbf{U}_1\mathbf{V}_1^T\mathbf{U}_1\mathbf{M}^T\mathbf{U}_2^T\mathbf{V}_2 + \gamma\mathbf{V}_2\mathbf{U}_2^T\mathbf{U}_2\mathbf{M}\mathbf{U}_1^T\mathbf{U}_1\mathbf{V}_1^T\mathbf{U}_1\mathbf{M}^T + \gamma\mathbf{U}_2\mathbf{V}_2^T\mathbf{U}_2\mathbf{M}\mathbf{U}_1^T\mathbf{V}_1\mathbf{U}_1^T\mathbf{U}_1\mathbf{M}^T\end{aligned}$$

The partial gradient of Equation (11) w.r.t. $\mathbf{U}_2$ is computed by $\frac{\partial J_3}{\partial \mathbf{U}_2} = \mathbf{X}_{11} - \mathbf{Y}_{11}$ where:

$$\begin{aligned}\mathbf{X}_{11} = {} & 2\beta[\mathbf{P}_{\bar{\Omega}_2} \odot (\mathbf{U}_2\mathbf{M}\mathbf{U}_1^T\mathbf{A}_1\mathbf{U}_1\mathbf{M}^T\mathbf{U}_2^T)]\mathbf{U}_2\mathbf{M}\mathbf{U}_1^T\mathbf{A}_1\mathbf{U}_1\mathbf{M}^T + \beta[\mathbf{P}_{\bar{\Omega}_2} \odot (\mathbf{U}_2\mathbf{V}_2^T)]\mathbf{V}_2 \\ & + 2\beta\mathbf{A}_2\mathbf{U}_2\mathbf{M}\mathbf{U}_1^T[\mathbf{P}_{\bar{\Omega}_1} \odot (\mathbf{U}_1\mathbf{M}^T\mathbf{U}_2^T\mathbf{A}_2\mathbf{U}_2\mathbf{M}\mathbf{U}_1^T)]\mathbf{U}_1\mathbf{M}^T \\ \mathbf{Y}_{11} = {} & \beta[\mathbf{P}_{\bar{\Omega}_2} \odot (\mathbf{U}_2\mathbf{M}\mathbf{U}_1^T\mathbf{A}_1\mathbf{U}_1\mathbf{M}^T\mathbf{U}_2^T)]\mathbf{V}_2 + \beta[\mathbf{P}_{\bar{\Omega}_2} \odot (\mathbf{U}_2\mathbf{V}_2^T + \mathbf{V}_2\mathbf{U}_2^T)]\mathbf{U}_2\mathbf{M}\mathbf{U}_1^T\mathbf{A}_1\mathbf{U}_1\mathbf{M}^T \\ & + \beta\mathbf{A}_2\mathbf{U}_2\mathbf{M}\mathbf{U}_1^T[\mathbf{P}_{\bar{\Omega}_1} \odot (\mathbf{U}_1\mathbf{V}_1^T + \mathbf{V}_1\mathbf{U}_1^T)]\mathbf{U}_1\mathbf{M}^T\end{aligned}$$

In this way, the fixed-point solution of $\frac{\partial J}{\partial \mathbf{U}_2} = \mathbf{0}$ under the nonnegativity constraint of $\mathbf{U}_2$ gives the following multiplicative update rule:

$$\mathbf{U}_2(p,q) \leftarrow \mathbf{U}_2(p,q)\sqrt[4]{\frac{\mathbf{Y}_9(p,q) + \mathbf{Y}_{10}(p,q) + \mathbf{Y}_{11}(p,q)}{\mathbf{X}_9(p,q) + \mathbf{X}_{10}(p,q) + \mathbf{X}_{11}(p,q)}} \tag{28}$$

The gradient of Equation (2) w.r.t. $\mathbf{V}_2$ is computed by $\frac{\partial J_1}{\partial \mathbf{V}_2} = \mathbf{X}_{12} - \mathbf{Y}_{12}$ where:

$$\mathbf{X}_{12} = [\mathbf{P}_{\Omega_2} \odot (\mathbf{U}_2\mathbf{V}_2^T)]\mathbf{U}_2 + \lambda\mathbf{V}_2$$
$$\mathbf{Y}_{12} = (\mathbf{P}_{\Omega_2} \odot \mathbf{A}_2)\mathbf{U}_2$$

The gradient of Equation (9) over $\mathbf{V}_2$ is computed by $\frac{\partial J_2}{\partial \mathbf{V}_2} = \mathbf{X}_{13} - \mathbf{Y}_{13}$ where:

$$\begin{aligned}\mathbf{X}_{13} = {} & \frac{\gamma}{2}\mathbf{1}_2\mathbf{1}_{r_2}^T[(\mathbf{M}\mathbf{U}_1^T\mathbf{D}_1\mathbf{U}_1\mathbf{M}^T\mathbf{U}_2^T) \odot \mathbf{U}_2^T]\mathbf{U}_2 + \frac{\gamma}{2}[(\mathbf{U}_2\mathbf{M}\mathbf{U}_1^T\hat{\mathbf{D}}_1\mathbf{U}_1\mathbf{M}^T) \odot \mathbf{U}_2]\mathbf{1}_{r_2}\mathbf{1}_2^T\mathbf{U}_2 \\ \mathbf{Y}_{13} = {} & \gamma\mathbf{U}_2\mathbf{M}\mathbf{U}_1^T\mathbf{V}_1\mathbf{U}_1^T\mathbf{U}_1\mathbf{M}^T\mathbf{U}_2^T\mathbf{U}_2\end{aligned}$$

And the gradient of Equation (11) w.r.t. $\mathbf{V}_2$ is $\frac{\partial J_3}{\partial \mathbf{V}_2} = \mathbf{X}_{14} - \mathbf{Y}_{14}$ where:

$$\mathbf{X}_{14} = \beta[\mathbf{P}_{\bar{\Omega}_2} \odot (\mathbf{V}_2\mathbf{U}_2^T)]\mathbf{U}_2$$
$$\mathbf{Y}_{14} = \beta[\mathbf{P}_{\bar{\Omega}_2} \odot (\mathbf{U}_2\mathbf{M}\mathbf{U}_1^T\mathbf{A}_1\mathbf{U}_1\mathbf{M}^T\mathbf{U}_2^T)]\mathbf{U}_2$$

In this way, the fixed-point solution of $\frac{\partial J}{\partial \mathbf{V}_2} = \mathbf{0}$ under the nonnegativity constraint of $\mathbf{V}_2$ leads to the multiplicative update rule as:

$$\mathbf{V}_2(p,q) \leftarrow \mathbf{V}_2(p,q)\sqrt[4]{\frac{\mathbf{Y}_{12}(p,q) + \mathbf{Y}_{13}(p,q) + \mathbf{Y}_{14}(p,q)}{\mathbf{X}_{12}(p,q) + \mathbf{X}_{13}(p,q) + \mathbf{X}_{14}(p,q)}} \tag{29}$$

B PROOF OF LEMMA 2

Proof. First, we prove that for any nonnegative $\mathbf{U}_1, \tilde{\mathbf{U}}_1$, we have $Z(\mathbf{U}_1, \tilde{\mathbf{U}}_1) \geq J(\mathbf{U}_1)$. Recall that the objective function w.r.t. $\mathbf{U}_1$ contains three parts, i.e., $J(\mathbf{U}_1) = J_1(\mathbf{U}_1) + J_2(\mathbf{U}_1) + J_3(\mathbf{U}_1)$. For $J_1(\mathbf{U}_1)$, after removing the constant terms w.r.t. $\mathbf{U}_1$, we have:

$$J_1(\mathbf{U}_1) = \underbrace{\frac{1}{2}\|\mathbf{P}_{\Omega_1} \odot (\mathbf{U}_1\mathbf{V}_1^T)\|_F^2}_{T_1} \underbrace{- \mathrm{Tr}(\mathbf{A}_1[\mathbf{P}_{\Omega_1} \odot (\mathbf{V}_1\mathbf{U}_1^T)])}_{T_2} + \underbrace{\frac{\lambda}{2}\|\mathbf{U}_1\|_F^2}_{T_3}$$

Similarly, we have the following for $J_2(\mathbf{U}_1)$ and $J_3(\mathbf{U}_1)$:

$$J_2(\mathbf{U}_1) = \underbrace{\frac{\gamma}{2}\mathrm{Tr}(\mathbf{U}_1^T\mathbf{D}_1\mathbf{U}_1\mathbf{M}^T\mathbf{U}_2^T\mathbf{D}_2\mathbf{U}_2\mathbf{M} + \mathbf{U}_1^T\hat{\mathbf{D}}_1\mathbf{U}_1\mathbf{M}^T\mathbf{U}_2^T\hat{\mathbf{D}}_2\mathbf{U}_2\mathbf{M})}_{T_4}$$
$$\underbrace{-\gamma\mathrm{Tr}(\mathbf{U}_2\mathbf{V}_2^T\mathbf{U}_2\mathbf{M}\mathbf{U}_1^T\mathbf{V}_1\mathbf{U}_1^T\mathbf{V}_1\mathbf{U}_1^T\mathbf{U}_1\mathbf{M}\mathbf{U}_2^T)}_{T_5}$$

$$J_3(\mathbf{U}_1) = \underbrace{\frac{\beta}{2}\|\mathbf{P}_{\bar{\Omega}_1} \odot (\mathbf{U}_1\mathbf{V}_1^T)\|_F^2}_{T_6} + \underbrace{\frac{\beta}{2}\|\mathbf{P}_{\bar{\Omega}_1} \odot (\mathbf{U}_1\mathbf{M}^T\mathbf{U}_2^T\mathbf{A}_2\mathbf{U}_2\mathbf{M}\mathbf{U}_1^T)\|_F^2}_{T_7} + \underbrace{\frac{\beta}{2}\|\mathbf{P}_{\bar{\Omega}_2} \odot (\mathbf{U}_2\mathbf{M}\mathbf{U}_1^T\mathbf{A}_1\mathbf{M}^T\mathbf{U}_2^T)\|_F^2}_{T_8}$$
$$\underbrace{-\beta\mathrm{Tr}([\mathbf{P}_{\bar{\Omega}_1} \odot (\mathbf{U}_1\mathbf{V}_1^T)][\mathbf{P}_{\bar{\Omega}_1} \odot (\mathbf{U}_1\mathbf{M}^T\mathbf{U}_2^T\mathbf{A}_2\mathbf{U}_2\mathbf{M}\mathbf{U}_1^T)])}_{T_9}$$
$$\underbrace{-\beta\mathrm{Tr}([\mathbf{P}_{\bar{\Omega}_2} \odot (\mathbf{U}_2\mathbf{V}_2^T)][\mathbf{P}_{\bar{\Omega}_2} \odot (\mathbf{U}_2\mathbf{M}\mathbf{U}_1^T\mathbf{A}_1\mathbf{U}_1\mathbf{M}^T\mathbf{U}_2^T)])}_{T_{10}}$$

Now we prove that $T_i' \geq T_i$ for $i = 1, \ldots, 10$ term by term. By the definition of $\mathbf{P}_{\Omega_1}$ and its symmetry, $\mathbf{P}_{\Omega_1} = \mathbf{P}_{\Omega_1}^T = \mathbf{P}_{\Omega_1} \odot \mathbf{P}_{\Omega_1}$. Let $\mathbf{U}_1(p,q) = \tilde{\mathbf{U}}_1(p,q)\mathbf{Q}(p,q),\ \forall p, q$, we have:

$$\begin{aligned}
T_1' &\geq \frac{1}{2}\sum_{p,q}[(\mathbf{P}_{\Omega_1} \odot \mathbf{P}_{\Omega_1}(\tilde{\mathbf{U}}_1\mathbf{V}_1^T))\mathbf{V}_1](p,q)\frac{\mathbf{U}_1^2(p,q)}{\tilde{\mathbf{U}}_1(p,q)} \\
&= \frac{1}{2}\sum_{p,q,r,s}\mathbf{P}_{\Omega_1}^2(p,r)\mathbf{V}_1(r,t)\mathbf{V}_1(r,q)\tilde{\mathbf{U}}_1(p,t)\tilde{\mathbf{U}}_1(p,q)\mathbf{Q}^2(p,q) \\
&= \frac{1}{2}\sum_{p,q,r,s}\mathbf{P}_{\Omega_1}^2(p,r)\mathbf{V}_1(r,t)\mathbf{V}_1(r,q)\tilde{\mathbf{U}}_1(p,t)\tilde{\mathbf{U}}_1(p,q)(\frac{\mathbf{Q}^2(p,t) + \mathbf{Q}^2(p,q)}{2}) \\
&\geq \frac{1}{2}\sum_{p,q,r,s}\mathbf{P}_{\Omega_1}^2(p,r)\mathbf{V}_1(r,t)\mathbf{V}_1(r,q)\tilde{\mathbf{U}}_1(p,t)\tilde{\mathbf{U}}_1(p,q)\mathbf{Q}(p,t)\mathbf{Q}(p,q) \\
&= \frac{1}{2}\sum_{p,q,r,s}\mathbf{P}_{\Omega_1}^2(p,r)\mathbf{V}_1(r,t)\mathbf{V}_1(r,q)\mathbf{U}_1(p,t)\mathbf{U}_1(p,q) = T_1
\end{aligned}$$

where the first and fourth line is due to $a^2 + b^2 \geq 2ab$ and the third line is due to the equivalence by switching $t \Leftrightarrow q$. We can prove $T_6' \geq T_6$ in the same way.

Next, by using the inequality $z \geq 1 + \log z,\ \forall z > 0$, we can easily prove $T_2' \geq T_2$. And similarly, $T_5' \geq T_5, T_9' \geq T_9$ and $T_{10}' \geq T_{10}$.

For T_3', we have $T_3' \geq \frac{\lambda}{2}\sum_{p,q}\mathbf{U}_1^2(p,q) = T_3$.

To show $T_6' \geq T_6$, we first consider the first term of T_6. By denoting $\mathbf{X} = \mathbf{M}^T\mathbf{U}_2^T\mathbf{D}_2\mathbf{U}_2\mathbf{M}$ which is symmetric, we have:

$$\begin{aligned}
&\frac{\gamma}{2}\mathrm{Tr}(\mathbf{D}_1\mathbf{U}_1\mathbf{X}\mathbf{U}_1^T) = \frac{\gamma}{2}\sum_{p,q,r,s}\mathbf{U}_1(p,r)(\mathbf{V}_1^T\mathbf{1}_1)(r,1)\mathbf{U}_1(p,s)\mathbf{X}(s,q)\mathbf{U}_1(p,q)\\
&= \frac{\gamma}{2}\sum_{p,q,r,s}\tilde{\mathbf{U}}_1(p,r)(\mathbf{V}_1^T\mathbf{1}_1)(r,1)\tilde{\mathbf{U}}_1(p,s)\mathbf{X}(s,q)\tilde{\mathbf{U}}_1(p,q)\mathbf{Q}(p,r)\mathbf{Q}(p,s)\mathbf{Q}(p,q)\\
&\leq \frac{\gamma}{2}\sum_{p,q,r,s}\tilde{\mathbf{U}}_1(p,r)(\mathbf{V}_1^T\mathbf{1}_1)(r,1)\tilde{\mathbf{U}}_1(p,s)\mathbf{X}(s,q)\tilde{\mathbf{U}}_1(p,q)(\frac{\mathbf{Q}(p,r)^3+\mathbf{Q}(p,s)^3+\mathbf{Q}(p,q)^3}{3})\\
&= \frac{\gamma}{6}\sum_{p,r}[((\tilde{\mathbf{U}}_1\mathbf{X})\odot\tilde{\mathbf{U}}_1)\mathbf{1}_{r_1}\mathbf{1}_1^T\mathbf{V}_1](p,r)\frac{\mathbf{U}_1^3(p,r)\tilde{\mathbf{U}}_1(p,r)}{\tilde{\mathbf{U}}_1^3(p,r)} + \frac{\gamma}{3}\sum_{p,q}[\mathrm{diag}(\tilde{\mathbf{U}}_1\mathbf{V}_1^T\mathbf{1})\tilde{\mathbf{U}}_1\mathbf{X}](p,q)\frac{\mathbf{U}_1^3(p,q)\tilde{\mathbf{U}}_1(p,q)}{\tilde{\mathbf{U}}_1^3(p,q)}\\
&\leq \frac{\gamma}{12}\sum_{p,q}[\frac{1}{2}((\tilde{\mathbf{U}}_1\mathbf{X})\odot\tilde{\mathbf{U}}_1)\mathbf{1}_{r_1}\mathbf{1}_1^T\mathbf{V}_1 + \mathrm{diag}(\tilde{\mathbf{U}}_1\mathbf{V}_1^T\mathbf{1})\tilde{\mathbf{U}}_1\mathbf{X}](p,q)\frac{3\mathbf{U}_1^4(p,q)+\tilde{\mathbf{U}}_1^4(p,q)}{\tilde{\mathbf{U}}_1^3(p,q)}
\end{aligned} \tag{30}$$

Similarly, by denoting $\hat{\mathbf{X}} = \mathbf{M}^T\mathbf{U}_2^T\hat{\mathbf{D}}_2\mathbf{U}_2\mathbf{M}$, we can show that:

$$\frac{\gamma}{2}\mathrm{Tr}(\hat{\mathbf{D}}_1\mathbf{U}_1\hat{\mathbf{X}}\mathbf{U}_1^T) \leq \frac{\gamma}{12}\sum_{p,q}[\frac{1}{2}\mathbf{1}_1\mathbf{1}_{r_1}^T(\tilde{\mathbf{U}}_1^T\odot(\hat{X}\tilde{\mathbf{U}}_1^T))\mathbf{V}_1 + \mathrm{diag}(\mathbf{V}_1\tilde{\mathbf{U}}_1^T\mathbf{1}_1)\tilde{\mathbf{U}}_1\hat{\mathbf{X}}](p,q)\frac{3\mathbf{U}_1^4(p,q)+\tilde{\mathbf{U}}_1^4(p,q)}{\tilde{\mathbf{U}}_1^3(p,q)} \tag{31}$$

Then by adding Equations (30) and (31) together, we can obtain $T_6 \leq T_6'$.

Denote $\mathbf{T} = \mathbf{M}^T\mathbf{U}_2^T\mathbf{A}_2\mathbf{U}_2\mathbf{M}$, we have:

$$\begin{aligned}
T_7' &= \frac{\beta}{2}\sum_{p,q}[(\mathbf{P}_{\bar{\Omega}_1}\odot(\tilde{\mathbf{U}}_1\mathbf{T}\tilde{\mathbf{U}}_1^T))\tilde{\mathbf{U}}_1\mathbf{T}](p,q)\frac{\mathbf{U}_1^4(p,q)}{\tilde{\mathbf{U}}_1^3(p,q)}\\
&= \frac{\beta}{2}\sum_{p,q,r,s,t,u}\mathbf{P}_{\bar{\Omega}_1}(p,r)\mathbf{T}(s,t)\mathbf{T}(u,q)\tilde{\mathbf{U}}_1(p,s)\tilde{\mathbf{U}}_1(r,t)\tilde{\mathbf{U}}_1(r,u)\tilde{\mathbf{U}}_(p,q)\mathbf{Q}^4(p,q)
\end{aligned}$$

Due to the symmetry of $\mathbf{P}_{\bar{\Omega}_1}$ and the fact that $\mathbf{P}_{\bar{\Omega}_1} = \mathbf{P}_{\bar{\Omega}_1}\odot\mathbf{P}_{\bar{\Omega}_1}$, by switching the indices $s \Leftrightarrow t, u \Leftrightarrow q, r \Leftrightarrow p$, we have:

$$T_7' = \frac{\beta}{2}\sum_{p,q,r,s,t,u}\mathbf{P}_{\bar{\Omega}_1}^2(p,r)\mathbf{T}(s,t)\mathbf{T}(u,q)\tilde{\mathbf{U}}_1(p,s)\tilde{\mathbf{U}}_1(r,t)\times\tilde{\mathbf{U}}_1(r,u)\tilde{\mathbf{U}}_(p,q)\mathbf{Q}^4(r,u)$$

And we can similar results by switching $s \Leftrightarrow u, t \Leftrightarrow q, p \Leftrightarrow r$ and $s \Leftrightarrow q, t \Leftrightarrow u, p \Leftrightarrow r$. In this way, we have:

$$\begin{aligned}
T_7' &= \frac{\beta}{2}\sum_{p,q,r,s,t,u}\mathbf{P}_{\bar{\Omega}_1}^2(p,r)\mathbf{T}(s,t)\mathbf{T}(u,q)\tilde{\mathbf{U}}_1(p,s)\tilde{\mathbf{U}}_1(r,t)\tilde{\mathbf{U}}_1(r,u)\tilde{\mathbf{U}}_1(p,q)\\
&\quad\times[\frac{\mathbf{Q}^4(p,q)+\mathbf{Q}^4(r,u)+\mathbf{Q}^4(r,t)+\mathbf{Q}^4(p,s)}{4}]\\
&\geq \frac{\beta}{2}\sum_{p,q,r,s,t,u}\mathbf{P}_{\bar{\Omega}_1}^2(p,r)\mathbf{T}(s,t)\mathbf{T}(u,q)\tilde{\mathbf{U}}_1(p,s)\tilde{\mathbf{U}}_1(r,t)\tilde{\mathbf{U}}_1(r,u)\tilde{\mathbf{U}}_(p,q)\mathbf{Q}(p,q)\mathbf{Q}(r,u)\mathbf{Q}(r,t)\mathbf{Q}(p,s)\\
&= \frac{\beta}{2}\sum_{p,q,r,s,t,u}\mathbf{P}_{\bar{\Omega}_1}^2(p,r)\mathbf{T}(s,t)\mathbf{T}(u,q)\mathbf{U}_1(p,s)\mathbf{U}_1(r,t)\mathbf{U}_1(r,u)\mathbf{U}_1(p,s)\\
&= T_7
\end{aligned}$$

Similarly, it can be proved that $T'_8 \geq T_8$. Then by adding $T'_i,\ i = 1, \ldots, 10$ together, we have:

$$Z(\mathbf{U}_1, \tilde{\mathbf{U}}_1) = \sum_{i=1}^{10} T'_i \geq \sum_{i=1}^{10} T_i = J(\mathbf{U}_1) \tag{32}$$

Second, we can directly prove $Z(\mathbf{U}_1, \mathbf{U}_1) = J(\mathbf{U}_1)$ by substituting $\tilde{\mathbf{U}}_1$ with $\mathbf{U}_1$ in all above inequalities.

Finally, to prove that the auxiliary function $Z(\mathbf{U}_1, \tilde{\mathbf{U}}_1)$ is convex w.r.t. $\mathbf{U}_1$, for the sake of brevity, we briefly show the Hessian matrix of $Z(\mathbf{U}_1, \tilde{\mathbf{U}}_1)$ is positive semi-definite. To start with, the derivative $\frac{\partial Z(\mathbf{U}_1, \tilde{\mathbf{U}}_1)}{\partial \mathbf{U}_1(p,q)}$ can be calculated as:

$$\frac{\partial Z(\mathbf{U}_1, \tilde{\mathbf{U}}_1)}{\partial \mathbf{U}_1(p,q)} = (\tilde{\mathbf{X}}_1 + \tilde{\mathbf{X}}_2 + \tilde{\mathbf{X}}_3)(p,q)\frac{\mathbf{U}_1^3(p,q)}{\tilde{\mathbf{U}}_1^3(p,q)} - (\tilde{\mathbf{Y}}_1 + \tilde{\mathbf{Y}}_2 + \tilde{\mathbf{Y}}_3)(p,q)\frac{\tilde{\mathbf{U}}_1(p,q)}{\mathbf{U}_1(p,q)} \tag{33}$$

where $\tilde{\mathbf{X}}_i, \tilde{\mathbf{Y}}_i,\ i = 1, 2, 3$ are all in terms of $\tilde{\mathbf{U}}_1$ while sharing the same formulas with $\mathbf{X}_i, \mathbf{Y}_i, i = 1, 2, 3$. For example, $\tilde{\mathbf{X}}_1 = [\mathbf{P}_{\Omega_1} \odot (\tilde{\mathbf{U}}_1\mathbf{V}_1^T)]\mathbf{V}_1 + \lambda\tilde{\mathbf{U}}_1$. Then the Hessian matrix w.r.t. $\mathbf{U}_1$ is computed by:

$$\frac{\partial^2 Z(\mathbf{U}_1, \tilde{\mathbf{U}}_1)}{\partial \mathbf{U}_1(p,q)\partial \mathbf{U}_1(r,s)} = \delta_{pr}\delta_{qs}\left[3(\tilde{\mathbf{X}}_1 + \tilde{\mathbf{X}}_2 + \tilde{\mathbf{X}}_3)(p,q)\frac{\mathbf{U}_1^2(p,q)}{\tilde{\mathbf{U}}_1^3(p,q)} + (\tilde{\mathbf{Y}}_1 + \tilde{\mathbf{Y}}_2 + \tilde{\mathbf{Y}}_3)(p,q)\frac{\tilde{\mathbf{U}}_1(p,q)}{\mathbf{U}_1^2(p,q)}\right] \tag{34}$$

where δ_{pr}, δ_{qs} are the Kronecker delta functions, i.e., $\delta_{pr} = 1$ if $p = r$; $\delta_{pr} = 0$ otherwise. Therefore, the Hessian matrix $\nabla^2_{\mathbf{U}_1} Z(\mathbf{U}_1, \tilde{\mathbf{U}}_1)$ is a diagonal matrix with nonnegative elements. As a result, the Hessian matrix is positive semi-definite, which means the auxiliary function $Z(\mathbf{U}_1, \tilde{\mathbf{U}}_1)$ is convex w.r.t. $\mathbf{U}_1$. In this way, the global minima of $Z(\mathbf{U}_1, \tilde{\mathbf{U}}_1)$ is obtained by setting its first-order derivative (i.e., Equation (33)) to be zero which further leads to the solution consistent with Equation (26).□

ACKNOWLEDGMENT

The content of the information in this document does not necessarily reflect the position or the policy of the Government or Amazon, 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.

REFERENCES

[1] Reid Andersen, Fan Chung, and Kevin Lang. 2006. Local graph partitioning using pagerank vectors. In*Proceedings of the 47th Annual IEEE Symposium on Foundations of Computer Science, (FOCS'06).* IEEE, 475–486.

[2] Nicola Barbieri, Francesco Bonchi, and Giuseppe Manco. 2014. Who to follow and why: Link prediction with explanations. In *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining.* ACM, 1266–1275.

[3] Mohsen Bayati, David F. Gleich, Amin Saberi, and Ying Wang. 2013. Message-passing algorithms for sparse network alignment. *ACM Transactions on Knowledge Discovery from Data* 7, 1 (2013), 3.

[4] Johannes Berg and Michael Lässig. 2004. Local graph alignment and motif search in biological networks. *Proceedings of the National Academy of Sciences of the United States of America* 101, 41 (2004), 14689–14694.

[5] Nicolas Boumal and Pierre-antoine Absil. 2011. RTRMC: A riemannian trust-region method for low-rank matrix completion. In *Proceedings of the Advances in Neural Information Processing Systems.* 406–414.

[6] Ulrik Brandes. 2008. On variants of shortest-path betweenness centrality and their generic computation. *Social Networks* 30, 2 (2008), 136–145.

[7] Jian-Feng Cai, Emmanuel J. Candès, and Zuowei Shen. 2010. A singular value thresholding algorithm for matrix completion. *SIAM Journal on Optimization* 20, 4 (2010), 1956–1982.

[8] 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.

[9] Chris Ding, Xiaofeng He, and Horst D. Simon. 2005. On the equivalence of nonnegative matrix factorization and spectral clustering. In *Proceedings of the 2005 SIAM International Conference on Data Mining*. SIAM, 606–610.

[10] 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*. 1339–1347.

[11] Boxin Du and Hanghang Tong. 2019. MrMine: Multi-resolution Multi-network embedding. In *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*. 479–488.

[12] Mohammed El-Kebir, Jaap Heringa, and Gunnar W. Klau. 2015. Natalie 2.0: Sparse global network alignment as a special case of quadratic assignment. *Algorithms* 8, 4 (2015), 1035–1051.

[13] 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.

[14] Mark Heimann, Haoming Shen, and Danai Koutra. 2018. Node representation learning for multiple networks: The case of graph alignment. *Arxiv Preprint Arxiv:1802.06257* (2018).

[15] Xiangnan Kong, Jiawei Zhang, and Philip S. Yu. 2013. Inferring anchor links across multiple heterogeneous social networks. In *Proceedings of the 22nd ACM International Conference on Information & Knowledge Management*. ACM, 179–188.

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

[17] Jure Leskovec, Deepayan Chakrabarti, Jon Kleinberg, Christos Faloutsos, and Zoubin Ghahramani. 2010. Kronecker graphs: An approach to modeling networks. *Journal of Machine Learning Research* 11, Feb (2010), 985–1042.

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

[19] Jure Leskovec and Julian J. Mcauley. 2012. Learning to discover social circles in ego networks. In *Proceedings of the Advances in Neural Information Processing Systems*. 539–547.

[20] Chung-Shou Liao, Kanghao Lu, Michael Baym, Rohit Singh, and Bonnie Berger. 2009. IsoRankN: Spectral methods for global alignment of multiple protein networks. *Bioinformatics* 25, 12 (2009), i253–i258.

[21] David Liben-Nowell and Jon Kleinberg. 2007. The link-prediction problem for social networks. *Journal of the Association for Information Science and Technology* 58, 7 (2007), 1019–1031.

[22] Ji Liu, Przemyslaw Musialski, Peter Wonka, and Jieping Ye. 2013. Tensor completion for estimating missing values in visual data. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 35, 1 (2013), 208–220.

[23] Li Liu, William K. Cheung, Xin Li, and Lejian Liao. 2016. Aligning users across social networks using network embedding. In *Proceedings of the 25th International Joint Conference on Artificial Intelligence*. 1774–1780.

[24] Yuanyuan Liu, Fanhua Shang, Hong Cheng, James Cheng, and Hanghang Tong. 2014. Factor matrix trace norm minimization for low-rank tensor completion. In *Proceedings of the 2014 SIAM International Conference on Data Mining*. SIAM, 866–874.

[25] Noël Malod-Dognin and Nataša Pržulj. 2015. L-GRAAL: Lagrangian graphlet-based network aligner. *Bioinformatics* 31, 13 (2015), 2182–2189.

[26] Hazel N. Manners, Ahed Elmsallati, Pietro H. Guzzi, Swarup Roy, and Jugal K. Kalita. 2017. Performing local network alignment by ensembling global aligners. In *Proceedings of the 2017 IEEE International Conference on Bioinformatics and Biomedicine (BIBM'17)*. IEEE, 1316–1323.

[27] Farzan Masrour, Iman Barjesteh, Rana Forsati, Abdol-Hossein Esfahanian, and Hayder Radha. 2015. Network completion with node similarity: A matrix completion approach with provable guarantees. In *Proceedings of the 2015 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM'15)*. IEEE, 302–307.

[28] Aditya Krishna Menon and Charles Elkan. 2011. Link prediction via matrix factorization. In *Proceedings of the Joint European Conference on Machine Learning and Knowledge Discovery in Databases*. Springer, 437–452.

[29] Kurt Miller, Michael I. Jordan, and Thomas L. Griffiths. 2009. Nonparametric latent feature models for link prediction. In *Proceedings of the Advances in Neural Information Processing Systems*. 1276–1284.

[30] Marco Mina and Pietro Hiram Guzzi. 2014. Improving the robustness of local network alignment: Design and extensive assessment of a markov clustering-based approach. *IEEE/ACM Transactions on Computational Biology and Bioinformatics* 11, 3 (2014), 561–572.

[31] K. B. Petersen, M. S. Pedersen, and others. 2008. The matrix cookbook, vol 7. *Technical University of Denmark* 15 (2008).

[32] Benjamin Recht. 2011. A simpler approach to matrix completion. *Journal of Machine Learning Research* 12, Dec (2011), 3413–3430.

[33] Jasson D. M. Rennie and Nathan Srebro. 2005. Fast maximum margin matrix factorization for collaborative prediction. In *Proceedings of the 22nd International Conference on Machine Learning*. ACM, 713–719.

[34] 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.

[35] Sucheta Soundarajan, Tina Eliassi-Rad, Brian Gallagher, and Ali Pinar. 2016. MaxReach: Reducing network incompleteness through node probes. In *Proceedings of the 2016 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM'16)*. IEEE, 152–157.

[36] Kim-Chuan Toh and Sangwoon Yun. 2010. An accelerated proximal gradient algorithm for nuclear norm regularized linear least squares problems. *Pacific Journal of Optimization* 6, 615–640 (2010), 15.

[37] Rianne van den Berg, Thomas N. Kipf, and Max Welling. 2017. Graph convolutional matrix completion. *arXiv preprint arXiv:1706.02263*.

[38] Vipin Vijayan, Vikram Saraph, and T. Milenković. 2015. MAGNA++: Maximizing accuracy in global network alignment via both node and edge conservation. *Bioinformatics* 31, 14 (2015), 2409–2411.

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

[40] Reza Zafarani and Huan Liu. 2013. Connecting users across social media sites: A behavioral-modeling approach. In *Proceedings of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 41–49.

[41] Jiawei Zhang, Jianhui Chen, Shi Zhi, Yi Chang, S. Yu Philip, and Jiawei Han. 2017. Link prediction across aligned networks with sparse and low rank matrix estimation. In *Proceedings of the 2017 IEEE 33rd International Conference on Data Engineering (ICDE'17)*. IEEE, 971–982.

[42] Jiawei Zhang and S. Yu Philip. 2015. Multiple anonymized social networks alignment. In *Proceedings of the 2015 IEEE International Conference onData Mining (ICDM'15)*. IEEE, 599–608.

[43] Si Zhang and Hanghang Tong. 2016. Final: Fast attributed network alignment. In *Proceedings of the 22th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. ACM*.

[44] Si Zhang and Hanghang Tong. 2018. Attributed network alignment: Problem definitions and fast solutions. *IEEE Transactions on Knowledge and Data Engineering* 31, 9 (2018), 1680–1692.

[45] Si Zhang, Hanghang Tong, Jiejun Xu, Yifan Hu, and Ross Maciejewski. 2019. Origin: Non-rigid network alignment. In *Proceedings of the 2019 IEEE International Conference on Big Data*. IEEE.

[46] 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.

Received May 2018; revised November 2019; accepted February 2020