iNEAT: Incomplete Network Alignment

***Abstract*—Network alignment and network completion are two fundamental cornerstones behind a wealth of high-impact graph mining applications. The state-of-the-arts 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 apriori, 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 paper, 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.**

I. Introduction

In the era of big data, networks arising from many high-impact applications are often *multi-sourced* (i.e., variety) and *incomplete* (i.e., veracity), e.g., the social networks from different social platforms, the protein-protein interaction (PPI) networks from multiple tissues, transaction networks from multiple financial institutes, etc. As such, network alignment (i.e., to find the node correspondence across multiple networks) and network completion (i.e., to infer the missing links) become two key and common tasks in many graph mining applications. Although the multi-sourced and incomplete characteristics often *co-exist* in many real networks, the state-of-the-arts have been largely addressing them *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 apriori [1], [2], whereas the existing network completion methods admit either a single network (i.e., matrix completion [3]) or multiple aligned networks (e.g., tensor completion [4]). How can we align two input networks when one or both of them are incomplete (e.g., with missing entries in the corresponding adjacency matrices of the input networks)?

A natural choice could be *completion-then-alignment*. That is, we first run network completion task on each of the two input networks separately, and then align the two complete networks. However, this strategy has some fundamental limits. First (*Alignment accuracy*), the promise of the *completion-then-alignment* 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 noisy links, which might compromise, or even outweight 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 [5]) have a time/space complexity that is 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]. By inferring the missing links, network completion would make each input network denser. If we simply conduct the network alignment task on such densified networks, it might make the computation even more intensive.

To address these limitation, we hypothesize that network alignment and network completion are inherently complementary with each other due to the following reasons. First, *(H1) alignment helps completion*. If two nodes across two networks are aligned together, intuitively, they might share similar connectivity patterns in both networks (e.g., connecting to the same set of nodes). Therefore, the knowledge about the existence or the absence of links in one network could help inferring the missing links in another network via alignment. Second, *(H2) completion helps alignment*. As mentioned above, network completion could potentially improve the alignment accuracy by providing higher-quality input networks for the alignment task. Moreover, network completion itself implicitly assumes a low-rank structure on the input networks, which, if harnessed appropriately, will actually *accelerate* the alignment process as we will show in the paper.

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. In detail, we formulate it as an optimization problem with the following two key ideas. First, 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

[1]Although the *empirical* runtime of some existing methods (e.g., *BigAlign* [2]) is *near-linear*, the big-O time complexity of these methods is still super-linear.

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.

The main contributions of the paper 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 multiplicative update to solve the optimization, and analyze its optimality, 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 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, in the 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 paper is organized as follows. Section 2 defines the incomplete network alignment problem and provides the preliminaries of the paper. Section 3 presents the proposed formulation of INEAT and Section 4 gives its optimization algorithm, followed by some analyses. Section 5 presents the experimental results. Related work and conclusion are given in Section 6 and Section 7.

II. Problem Definition

A. Problem Definition

Table I summarizes the main symbols and notations used throughout the paper. 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. That is, we only have the knowledge 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 paper aims to address.

TABLE I: 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 $\mathbf{A}_1$ and $\mathbf{A}_2$
$\mathbf{S}$	an $n_2 \times n_1$ alignment matrix between $\mathbf{A}_2$ and $\mathbf{A}_1$
$P_\Omega(\cdot), P_{\bar{\Omega}}(\cdot)$	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

Problem 1. INCOMPLETE NETWORK ALIGNMENT.

Given: *(1) Incomplete adjacency matrices $\mathbf{A}_1$, $\mathbf{A}_2$ of two 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 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 $\mathbf{A}_2^*$.*

B. Preliminaries

A - Network Alignment. Most existing network alignment algorithms (such as *IsoRank* [6] and *FINAL* [5]), 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 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

[2]In [5], the authors generalize the topology consistency principle to further accommodate the additional node/edge attribute information, which is outside the scope of this paper

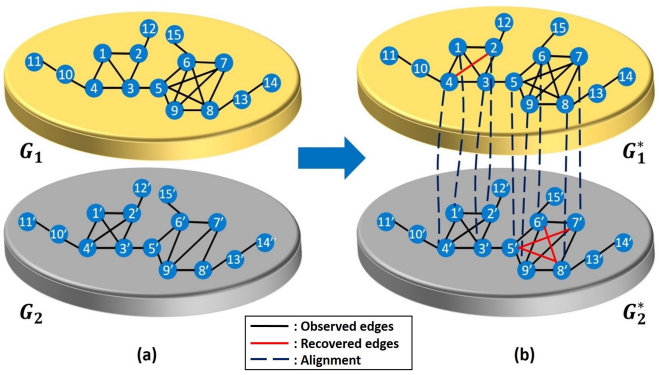


Fig. 1: An illustrative example. Figure 1(a) shows the input incomplete networks and Figure 1(b) shows part of the alignment across two complete networks.

matrix. Mathematically, this naturally leads to the following optimization problem.

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

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 [5], we use the scaled similarity matrix $\mathbf{S}$ as the alignment matrix throughout our paper 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 is to avoid trivial solutions.

In order to solve the network alignment problem in Eq. (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 $O(n^6)$. In [5], 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 [7]. However, since real-world networks are usually very large, it is very costly to directly minimize the nuclear norm of the adjacency matrices. In [8], 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:

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

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.

III. 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.

A. 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 in higher-quality networks. We use the factorization-based network completion (i.e., Eq (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 Eq. (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 Eq. (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.

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

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., Eq. (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 S. *Let $\hat{\mathbf{s}}$ be the solution of Eq. (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 Eq. (3), the closed-form solution of similarity matrix $\hat{\mathbf{S}}$ can be computed by

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

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

First, we rewrite $\mathbf{\Lambda}^{-1}$ as follows. Since for any two matrices $\mathbf{X}, \mathbf{Y}$, the eigenvalues of their product satisfies $\text{eig}(\mathbf{XY}) = \text{eig}(\mathbf{YX})$ [9], 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 Taylor expansion on $\mathbf{\Lambda}^{-1}$ as

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

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

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

Thus, by substituting the Kronecker product form of $(2\hat{\mathbf{D}})^{-1}$ into Eq. (5), the matrix $\mathbf{\Lambda}^{-1}$ can be derived as

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

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 Eq. (6) into Eq. (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

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

□

Remarks. Eq. (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. 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 low-rank structure part of $\mathbf{S}$ (i.e., $\mathbf{U}_2\mathbf{M}\mathbf{U}_1$) from the input 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., $\mathbf{S} \approx \mathbf{U}_2\mathbf{M}\mathbf{U}_1$). We then can calibrate the result by averaging between the learned $\mathbf{S}$ and the prior knowledge $\mathbf{H}$, 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*.

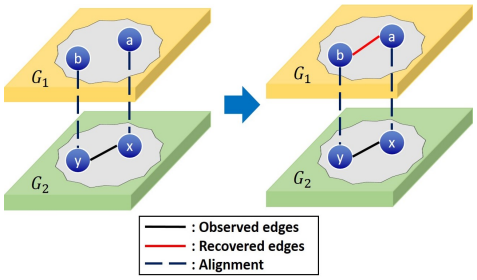


Fig. 2: Network completion via the alignment.

To take advantages of the low-rank structure of $\mathbf{S}$ under the above strategy, instead of minimizing Eq. (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:

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

B. Network Alignment Helps Network Completion.

Despite the effectiveness of the factorization-based network completion methods (i.e., Eq. (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

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

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.

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

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

C. 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 Eq. (2), Eq. (9) and Eq. (11) together, the overall optimization problem is formulated as

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

IV. Proposed Optimization Algorithm

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

A. Optimization Algorithm

Since the overall objective function Eq. (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 until convergence. Due to the space limitation, we only show the minimization procedures over $\mathbf{U}_1$ and $\mathbf{M}$. Other variables such as $\mathbf{V}_1,\mathbf{U}_2,\mathbf{V}_2$ can be solved in a similar way as $\mathbf{U}_1$. First, we show the update algorithm over $\mathbf{U}_1$. The gradient of Eq. (2) with respect to $\mathbf{U}_1$ is computed by

$$\frac{\partial J_1}{\partial \mathbf{U}_1} = \mathbf{X}_1 - \mathbf{Y}_1 \quad (13)$$

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 Eq. (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 in terms of $\mathbf{U}_1$, thus the partial gradient is computed by

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

where

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

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

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

where

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

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 nonnegativity constraint of $\mathbf{U}_1$ leads to the following multiplicative update rule

$$\mathbf{U}_1(u,v) \leftarrow \mathbf{U}_1(u,v)\sqrt[4]{\frac{\mathbf{Y}_1(u,v) + \mathbf{Y}_2(u,v) + \mathbf{Y}_3(u,v)}{\mathbf{X}_1(u,v) + \mathbf{X}_2(u,v) + \mathbf{X}_3(u,v)}} \quad (16)$$

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

$$\frac{\partial J_2}{\mathbf{M}} = \mathbf{X}_4 - \mathbf{Y}_4 \quad (17)$$

where

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

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

$$\frac{\partial J_3}{\mathbf{M}} = \mathbf{X}_5 - \mathbf{Y}_5 \quad (18)$$

where

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

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 $\mathbf{U}_1, \mathbf{V}_1, \mathbf{U}_2, \mathbf{V}_2$ as Eq. (20), $\mathbf{M}$ as Eq. (21), $t = 1$;
2: **while** not converge and $t \leq t_{\max}$ **do**
3: Update $\mathbf{U}_1$ by Eq. (16) until convergence;
4: Update $\mathbf{V}_1$ until its convergence;
5: Update $\mathbf{U}_2$ until its convergence;
6: Update $\mathbf{V}_2$ until its convergence;
7: Update $\mathbf{M}$ by Eq. (19) until convergence;
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: $\mathbf{S} = \alpha\mathbf{U}_2\mathbf{M}\mathbf{U}_1^T + (1-\alpha)\mathbf{H}$.

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

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

Initialization. Since the optimization problem in Eq. (12) is not a joint convex problem, a good initialization of each variable group could play an important role of obtaining a good final 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 $\mathbf{U}_1 \geq \mathbf{0}$. Same as [10], we use the following multiplicative update rule to obtain the solution

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

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 Eq. (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 large number, e.g., 500.

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 $\mathbf{S}$ by averaging between $\mathbf{U}_2\mathbf{M}\mathbf{U}_1^T$ and $\mathbf{H}$ (line 11).

B. 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 Eq. (16) satisfies the 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 here for brevity.

Theorem 1. ***Optimality of Eq.*** (16)**.** *At convergence, the fixed-point solution of Eq.* (16) *satisfies the KKT condition.*

Proof. Let $\mathbf{\Sigma} \in \mathbb{R}^{n_1 \times r_1}$ be the Lagrangian multiplier. Define the Lagrangian function w.r.t $\mathbf{U}_1$ of Eq. (12) as

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

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

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

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 Eq. (16), at convergence, we have

$$\mathbf{U}_1 = \mathbf{U}_1 \odot (\frac{\mathbf{Y}_1 + \mathbf{Y}_2 + \mathbf{Y}_3}{\mathbf{X}_1 + \mathbf{X}_2 + \mathbf{X}_3})^{\frac{1}{4}} \tag{24}$$

which is equivalent to

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

Eq. (23) and Eq. (25) are equivalent, so we can see that at convergence, Eq. (16) satisfies the KKT condition, which guarantees a local optimum. □

Then, we show the convergence of updating $\mathbf{U}_1$ under Eq. (16). First, the following lemma gives the auxiliary function for the objective function Eq. (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 Eq.* (12) *that contains $\mathbf{U}_1$, then the following function $Z(\mathbf{U}_1, \tilde{\mathbf{U}}_1)$*

$$\begin{aligned}
Z(\mathbf{U}_1, \tilde{\mathbf{U}}_1) &= -\gamma T_1 - \beta T_2 - \beta T_3 \\
&= \frac{1}{4}\sum_{p,q}[(\mathbf{P}_{\Omega_1} \odot (\tilde{\mathbf{U}}_1\mathbf{V}_1^T))\mathbf{V}_1](p,q)\frac{\mathbf{U}_1^4(p,q) + \tilde{\mathbf{U}}_1^4(p,q)}{\tilde{\mathbf{U}}_1^3(p,q)} \\
&- \sum_{p,q}[(\mathbf{P}_{\Omega_1} \odot \mathbf{A}_1)\mathbf{V}_1](p,q)\tilde{\mathbf{U}}_1(p,q)(1 + \log\frac{\mathbf{U}_1(p,q)}{\tilde{\mathbf{U}}_1(p,q)}) \\
&+ \sum_{p,q}(\frac{\lambda}{12}\tilde{\mathbf{U}}_1(p,q) + \frac{\gamma}{12}\tilde{\mathbf{X}}_2(p,q))\frac{3\mathbf{U}_1^4(p,q) + \tilde{\mathbf{U}}_1^4(p,q)}{\tilde{\mathbf{U}}_1^3(p,q)} \\
&+ \frac{\beta}{2}\sum_{p,q}\tilde{\mathbf{X}}_3(p,q)\frac{\mathbf{U}_1^4(p,q)}{\tilde{\mathbf{U}}_1^3(p,q)}
\end{aligned}$$

where $\tilde{\mathbf{X}}_2$ is same as $\mathbf{X}_2$ except that $\tilde{\mathbf{X}}_2$ is in terms of $\tilde{\mathbf{U}}_1$, while $\mathbf{X}_2$ is w.r.t $\mathbf{U}_1$. $\tilde{\mathbf{X}}_3$ is defined similarly. And

$$\begin{aligned}
T_1 &= \sum_{o,p,q,r,s}(\mathbf{M}^T\mathbf{U}_2^T\mathbf{U}_2\mathbf{V}_2^T\mathbf{U}_2\mathbf{M})(o,q)\mathbf{V}_1(p,r)\tilde{\mathbf{U}}_1(s,r)\tilde{\mathbf{U}}_1(s,o) \\
&\quad \times \tilde{\mathbf{U}}_1(p,q)(1 + \log\frac{\mathbf{U}_1(p,q)\mathbf{U}_1(s,r)\mathbf{U}_1(s,o)}{\tilde{\mathbf{U}}_1(p,q)\tilde{\mathbf{U}}_1(s,r)\tilde{\mathbf{U}}_1(s,o)}) \\
T_2 &= \sum_{o,p,q,r,s}\mathbf{P}_{\bar{\Omega}_1}(p,o)\mathbf{V}_1(o,q)(\mathbf{M}^T\mathbf{U}_2^T\mathbf{A}_2\mathbf{U}_2\mathbf{M})(s,r)\tilde{\mathbf{U}}_1(o,s) \\
&\quad \times \tilde{\mathbf{U}}_1(p,r)\tilde{\mathbf{U}}_1(p,q)(1 + \log\frac{\mathbf{U}_1(p,q)\mathbf{U}_1(o,s)\mathbf{U}_1(p,r)}{\tilde{\mathbf{U}}_1(p,q)\tilde{\mathbf{U}}_1(o,s)\tilde{\mathbf{U}}_1(p,r)}) \\
T_3 &= \sum_{p,q,r,s}[\mathbf{M}^T\mathbf{U}_2^T(\mathbf{P}_{\bar{\Omega}_2} \odot (\mathbf{V}_2\mathbf{U}_2^T))\mathbf{U}_2\mathbf{M}](s,q)\mathbf{A}_1(p,r)\tilde{\mathbf{U}}_1(r,s) \\
&\quad \times \tilde{\mathbf{U}}_1(p,q)(1 + \log\frac{\mathbf{U}_1(p,q)\mathbf{U}_1(r,s)}{\tilde{\mathbf{U}}_1(p,q)\tilde{\mathbf{U}}_1(r,s)})
\end{aligned}$$

is an auxiliary function for $J(\mathbf{U}_1)$ *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* $\mathbf{U}_1$.

Proof. Omitted for brevity. □

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

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

Proof. Omitted for brevity. □

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

Proof. Omitted for brevity. □

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., $\mathbf{U}_1\mathbf{V}_1^T$, $\mathbf{U}_2\mathbf{M}\mathbf{U}_1$, etc.) as well as the need to store the potentially dense matrices (e.g., $\mathbf{A}_1^*$, $\mathbf{S}$, etc). 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 $\mathbf{V}_1$.

V. 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?

A. 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 II.

- *Collaboration Network*: We use the collaboration network in the general relativity and quantum cosmology (*Gr-Qc*) area from the e-print arXiv [11]. In the network, each node represents an author and there exists an edge if two authors have co-authored at least one paper.

TABLE II: 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

- *Infrastructure Network*: This dataset is a network of Autonomous Systems (AS) inferred from Oregon route-view [11]. In the network, nodes are the routers, and edges represent the peering information among routers.
- *Social Networks*: We use the social network collected from Google+ [12]. In the network, nodes are the users and an edge denotes that one user has the other user in her circles. We also use the Youtube network [13] where nodes are the Youtube users and edges represent the friendship among users.

Based on these datasets, 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. We run our algorithm and other comparison methods in all the pairs of incomplete networks.

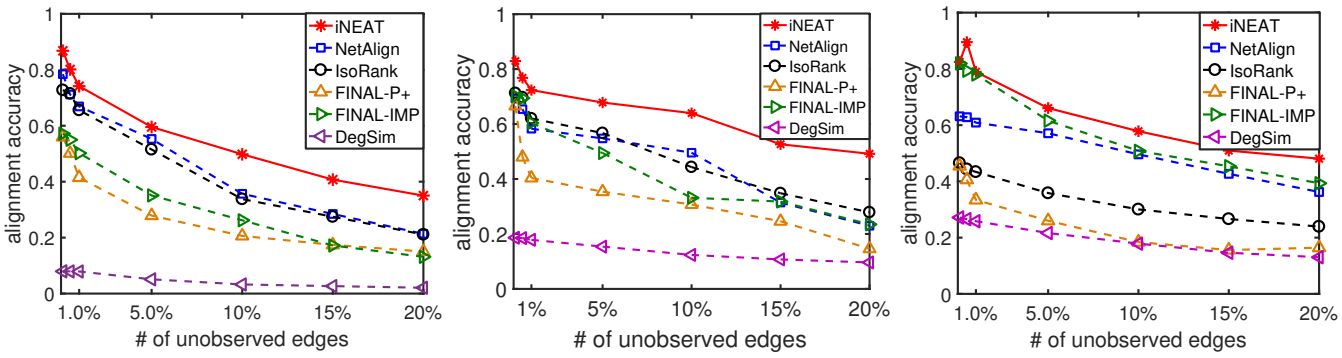
Comparison Methods.

- *Alignment*. To evaluate the alignment performance of our proposed algorithm INEAT, we compare it with the following existing network alignment algorithms, including (1) *NetAlign* [14], (2) *IsoRank* [6], (3) *FINAL-P+* [5]. Besides, in order to validate whether alignment and imputation are mutually beneficial from each other, we use the low-rank networks completed solely by Eq. (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 Eq. (2) (*NMF-IMP*), (2) an accelerated proximal gradient based nuclear norm minimization method (*NNLS*) [15], (3) a Riemannian trust-region based matrix completion method (*RTRMC*) [16].

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

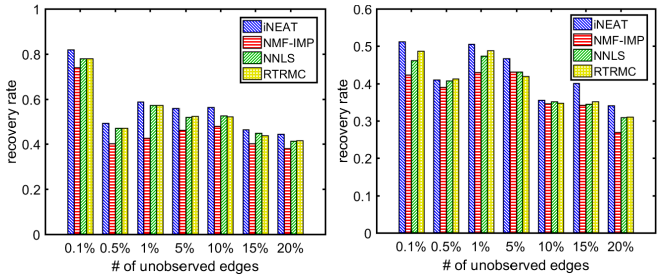
B. Effectiveness Analysis

We first evaluate the alignment accuracy with different numbers of unobserved edges in the incomplete networks. We



(a) Cr-Qc collaboration network. (b) Oregon infrastructure network. (c) Google+ social network.

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



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

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

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 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 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*, *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 Figure 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 paper). Then, we calculate the recovery rate over the total number of missing edges. 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.

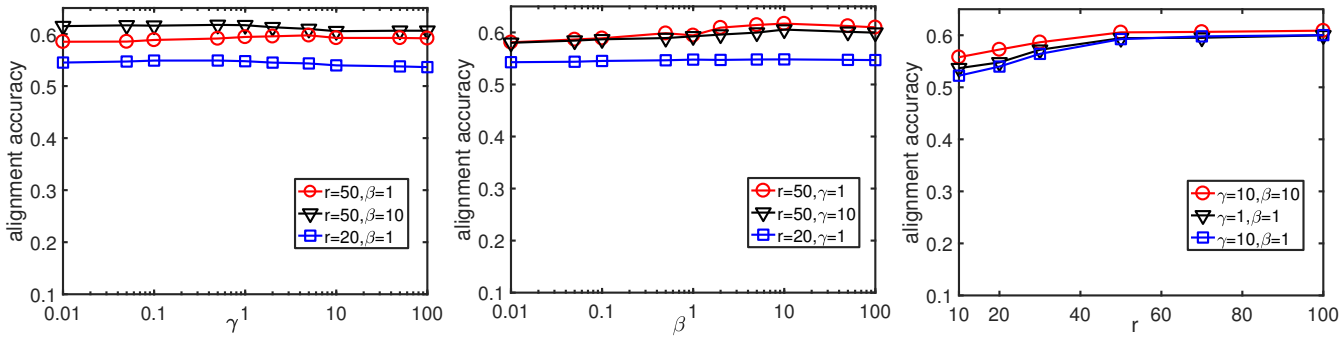
Third, we study how different parameters affect the alignment accuracy. In our experiments, we mainly study three parameters, including (1) γ which controls the importance of alignment task, (2) β which controls the importance of cross-network completion task, and (3) r which is the rank of the complete 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.

C. 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*.

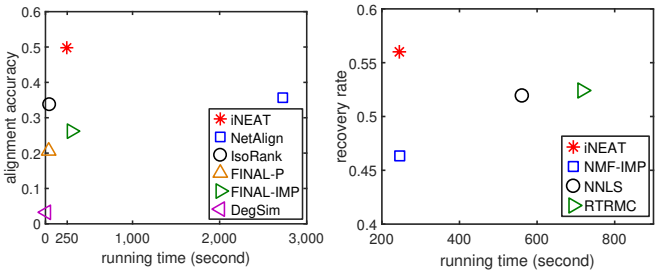
On the other hand, to evaluate the quality of network completion, note that the running time is the time for completing two incomplete networks. As Figure 6(b) shows, iNEAT obtains a better recovery rate and less running time. To be specific, compared with *NMF-IMP*, iNEAT can recover 10% more missing 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



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

Fig. 5: Parameter study on collaboration networks with 5% unobserved edges: study the effect of the parameters γ, β and network rank r 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.

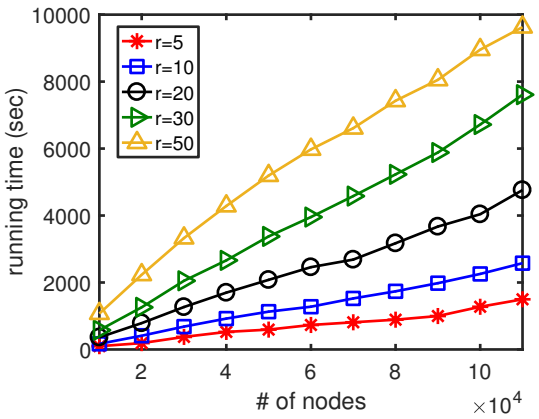


Fig. 7: Running time vs. the number of nodes in the networks.

the number of nodes in the networks which is consistent with our time complexity analysis.

VI. Related Work

Network Alignment. Network alignment is a fundamental task in many real-world applications, including bioinformatics [17], computer vision [18], database [19] and data mining [5].

Many network alignment algorithms are based on the *topology consistency*. For example, one well-known method *IsoRank* computes the cross-network pairwise topology similarities by a weighted sum of 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 [6]. In addition, *IsoRankN* extends the original *IsoRank* algorithm by using PageRank-Nibble to align multiple networks [17]. Koutra et al. propose *BigAlign* algorithm to align across the bipartite networks which assumes that one network is the noisy permutation of the other network and infers the soft alignment by using alternative projected gradient descent method [2]. Zhang et al. use the similar permutation assumptions of the networks and introduce an additional *transitivity* property to align multiple networks based on the topology information [1]. From a different point of view, *NetAlign* formulates the network alignment problem as an integer programming problem to maximize the number of "squares", i.e., the number of neighboring node pairs that are aligned [14]. And it is solved by a message-passing approach.

More recently, there are alignment algorithms to align the attributed networks. For instance, *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 two attributed networks [20]. Zhang et al. propose an attributed network alignment algorithm by adopting both the topological and attribute consistency principles [5]. This work formulates these consistency principles into a convex quadratic problem and propose a fixed-point solution to solve it. For the similar cross-site user identification problem, Zafarani et al. [21] models the user behaviors based on human limitations, exogenous and endogenous factors. 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 an Expectation Maximization (EM) based method that fits the network structure under the Kronecker graph model and re-estimates the parameters using a scalable Gibbs sampling approach iteratively [22]. 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 [3]. 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 (SVT) which iteratively shrinkages the singular values to minimize the nuclear norm [23]. 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 [15]. 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 [16]. In addition, there are some recent work to complete multiple *aligned* networks by tensor completion [24], [4]. Nonetheless, how these input networks are aligned at the first place was not answered in these work.

VII. 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 paper, 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 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.

References

[1] J. Zhang and S. Y. Philip, "Multiple anonymized social networks alignment," in *Data Mining (ICDM), 2015 IEEE International Conference on*. IEEE, 2015, pp. 599–608.

[2] D. Koutra, H. Tong, and D. Lubensky, "Big-align: Fast bipartite graph alignment," in *2013 IEEE 13th International Conference on Data Mining*. IEEE, 2013, pp. 389–398.

[3] F. Masrour, I. Barjesteh, R. Forsati, A.-H. Esfahanian, and H. Radha, "Network completion with node similarity: A matrix completion approach with provable guarantees," in *Advances in Social Networks Analysis and Mining (ASONAM), 2015 IEEE/ACM International Conference on*. IEEE, 2015, pp. 302–307.

[4] Y. Liu, F. Shang, H. Cheng, J. Cheng, and H. Tong, "Factor matrix trace norm minimization for low-rank tensor completion," in *Proceedings of the 2014 SIAM International Conference on Data Mining*. SIAM, 2014, pp. 866–874.

[5] S. Zhang and H. Tong, "Final: Fast attributed network alignment," in *Proceedings of the 22th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. ACM*, 2016.

[6] R. Singh, J. Xu, and B. Berger, "Global alignment of multiple protein interaction networks with application to functional orthology detection," *Proceedings of the National Academy of Sciences*, vol. 105, no. 35, pp. 12 763–12 768, 2008.

[7] B. Recht, "A simpler approach to matrix completion," *Journal of Machine Learning Research*, vol. 12, no. Dec, pp. 3413–3430, 2011.

[8] J. D. Rennie and N. Srebro, "Fast maximum margin matrix factorization for collaborative prediction," in *Proceedings of the 22nd international conference on Machine learning*. ACM, 2005, pp. 713–719.

[9] M. S. Pedersen, B. Baxter, B. Templeton, C. RishÃÿj, D. L. Theobald, E. Hoeghrasmussen, G. Casteel, J. B. Gao, K. Dedecius, and K. Strim, "The matrix cookbook," 2008.

[10] C. Ding, X. He, and H. D. Simon, "On the equivalence of nonnegative matrix factorization and spectral clustering," in *Proceedings of the 2005 SIAM International Conference on Data Mining*. SIAM, 2005, pp. 606–610.

[11] J. Leskovec, J. Kleinberg, and C. Faloutsos, "Graph evolution: Densification and shrinking diameters," *ACM Transactions on Knowledge Discovery from Data (TKDD)*, vol. 1, no. 1, p. 2, 2007.

[12] J. Leskovec and J. J. Mcauley, "Learning to discover social circles in ego networks," in *Advances in neural information processing systems*, 2012, pp. 539–547.

[13] J. Yang and J. Leskovec, "Defining and evaluating network communities based on ground-truth," *Knowledge and Information Systems*, vol. 42, no. 1, pp. 181–213, 2015.

[14] M. Bayati, D. F. Gleich, A. Saberi, and Y. Wang, "Message-passing algorithms for sparse network alignment," *ACM Transactions on Knowledge Discovery from Data (TKDD)*, vol. 7, no. 1, p. 3, 2013.

[15] K.-C. Toh and S. Yun, "An accelerated proximal gradient algorithm for nuclear norm regularized linear least squares problems," *Pacific Journal of optimization*, vol. 6, no. 615-640, p. 15, 2010.

[16] N. Boumal and P.-a. Absil, "Rtrmc: A riemannian trust-region method for low-rank matrix completion," in *Advances in neural information processing systems*, 2011, pp. 406–414.

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

[18] C. Schellewald and C. Schnörr, "Probabilistic subgraph matching based on convex relaxation," in *International Workshop on Energy Minimization Methods in Computer Vision and Pattern Recognition*. Springer, 2005, pp. 171–186.

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

[20] Y. Zhang, J. Tang, Z. Yang, J. Pei, and P. S. Yu, "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, 2015, pp. 1485–1494.

[21] R. Zafarani and H. Liu, "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, 2013, pp. 41–49.

[22] M. Kim and J. Leskovec, "The network completion problem: Inferring missing nodes and edges in networks," in *Proceedings of the 2011 SIAM International Conference on Data Mining*. SIAM, 2011, pp. 47–58.

[23] J.-F. Cai, E. J. Candès, and Z. Shen, "A singular value thresholding algorithm for matrix completion," *SIAM Journal on Optimization*, vol. 20, no. 4, pp. 1956–1982, 2010.

[24] J. Liu, P. Musialski, P. Wonka, and J. Ye, "Tensor completion for estimating missing values in visual data," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 35, no. 1, pp. 208–220, 2013.