Attributed Network Alignment: Problem Definitions and Fast Solutions

Si Zhang, Hanghang Tong

Abstract—
Networks are prevalent and often collected from multiple sources in many high-impact domains, which facilitate many emerging applications that require the connections across multiple networks. Network alignment (i.e., to find the node correspondence between different networks) has become the very first step in many applications and thus has been studied in decades. Although some existing works can use the attribute information as part of the alignment process, they still have certain limitations. For example, some existing network alignment methods can use node attribute similarities as part of the prior alignment information, whereas most of them solely explore the *topology consistency* without the consistency among attributes of the underlying networks. On the other hand, traditional graph matching methods encode both the node and edge attributes (and possibly the topology) into an affinity matrix and formulate it as a constrained nonconvex quadratic maximization problem. However, these methods cannot scale well to the large-scale networks. In this paper, we propose a family of network alignment algorithms (FINAL) to *efficiently* align the *attributed* networks. The key idea is to leverage the node/edge attribute information to guide the (topology-based) alignment process. We formulate this problem as a convex quadratic optimization problem, and develop effective and efficient algorithms to solve it. Moreover, we derive FINAL ON-QUERY, an online variant of FINAL that can find similar nodes for the query nodes across networks. We perform extensive evaluations on real networks to substantiate the superiority of our proposed approaches.

Index Terms—Network alignment, attributed networks, alignment consistency, on-query alignment

1 INTRODUCTION

Networks in many areas such as finance and social analysis, are often collected from multiple sources, leading to numerous emerging high-impact applications. However, an immense amount of these applications often require the knowledge of the relationships across multiple networks. Network alignment, on the other hand, is a powerful tool to explore the node correspondence amongst different networks, and has become the very first step to many applications. Finding the virtual identical twins across social networks, for instance, enables to measure the node proximity at a finer granularity [1]. Moreover, identification of the same customers in different transaction networks contributes to more comprehensive understandings of transaction behaviors and therefore helps detect financial fraud [2].

In the last two decades, there is a tremendous progress that has been made on network alignment [3]. For example, an early work *IsoRank* [4] propagates the node pairwise similarities in a random walk-like way in the Kronecker product graph. Some other works formulate network alignment as a maximum common subgraph problem, such as [5], [6]. Although some existing methods can use the node attribute information, most of them solely hypothesize the *topology consistency* as the basic assumption, i.e., same nodes connecting to the same or similar set of nodes. However, this assumption has two fundamental limitations that (1) different nodes might have similar connectivity patterns, and (2) same nodes could exhibit disparately across networks.

• *Si Zhang and Hanghang Tong are with the School of Computing Informatics, Decision Systems Engineering, Arizona State University, Tempe, AZ 85281.*
E-mail: {szhan172, htong6}@asu.edu

Meanwhile, many real-world networks are accompanied by rich numerical and categorical attribute information, including node attributes (e.g., user demographic information) and/or edge attributes (e.g., interaction information between users). Therefore, the fusion of both the topology consistency and attribute consistency might be a good cure to tackle these limitations. Researchers have been studying how to leverage the node attribute information and/or network structure to identify the unique users across different social networks [7], [8], [9], [10]. For example, [10] extracts some structural features of each node from the networks and trains a binary classifier using the structural features and other node attribute information, to identify the unique users across multiple networks. More recently, *COSNET* models both local consistency based on the node attributes and global consistency based on the network structures into an energy-based model to predict the anchor links [9]. These identity matching based methods only endorse the attributes on nodes and require some identified nodes in advance to train the model with network structures and node attributes. On the other side, the traditional graph matching approaches can encode both node and edge attributes together with the adjacency matrices into an affinity matrix [11], [12]. The nodes one-to-one mapping can be obtained by solving a nonconvex quadratic maximization problem. However, these methods are not scalable to the large-scale networks.

Alternatively, in this paper, we shift our attention to the *unsupervised* network alignment method that can not only use the topological information of the networks, but can also take advantages of both node and edge attributes. Yet, it still remains to be a daunting task to align attributed networks due to the following three challenges. First (*Q1.*

Formulation), it is not clear how to assimilate both the node and edge attribute information into the topology-based network alignment and formulate it as a single optimization problem. Second (*Q2. Algorithms*), the optimization problem behind the topology-based network alignment is often non-convex or even NP-hard (e.g., maximum common subgraph optimization problem [5], [6]). Introducing attributes into the alignment process could further perplex the corresponding optimization problem. Third (*Q3. Computations*), while the scalability of the alignment algorithms is much desirable, it remains unknown how to accelerate and scale up the algorithms by taking advantage of some intrinsic properties (e.g., low-rank) of real networks.

To address these challenges, in this paper, we propose a family of effective and efficient algorithms to solve the attributed network alignment problem. The key idea behind our algorithms is to generalize the *topology consistency* to *alignment consistency* and leverage attribute information to guide the topology-based alignment process. Different from [13], the algorithms are extended to multiple numerical/categorical attributes. However, the computational challenges lie in the matrix multiplication between sparse adjacency matrix and the alignment matrix. Thanks to the low-rank structure of many real-world networks, we further propose an approximation algorithm for speed-up.

In some applications, we might be interested in finding similar nodes across different networks (e.g., to find similar users on `LinkedIn` for a given user on `Facebook`). We define this problem as the on-query attributed network alignment problem and propose a *linear* approximation algorithm without solving the full alignment problem.

The main contributions are summarized as follows.

1) **Formulations.** We define the attributed network alignment problem and formulate it as a *convex* quadratic optimization problem. As a side product, our formulation helps reveal the quantitative relationships between the (attributed) network alignment problem and several other network mining problems.
2) **Algorithms and Analysis.** We propose a family of algorithms FINAL to efficiently solve the attributed network alignment problem. To speed up, we further propose an approximation algorithm to solve full alignment problem. We propose a *linear* online algorithm for on-query alignment problem. We then analyze the optimality, convergence, complexity and stability.
3) **Evaluations.** We perform extensive experiments to validate the efficacy of the proposed algorithms. Our evaluations demonstrate that (1) our algorithms significantly improve the alignment accuracy by up to 30% over the existing methods; (2) the proposed FINAL-N+ algorithm leads to a better trade-off between alignment accuracy and running time; and (3) our on-query alignment method scales *linearly*, with an around 90% ranking accuracy compared with the exact full alignment method and a near real-time response time.

The rest of the paper is organized as follows. Section 2 defines the attributed network alignment problem and the on-query attributed network alignment problem. Section 3 presents the proposed optimization formulation of FINAL and its solutions followed by some analyses. Section 4 proposes two speed-up methods for approximate full alignment and on-query alignment. Section 5 presents the experimental results. Related work and conclusion are given in Section 6 and Section 7.

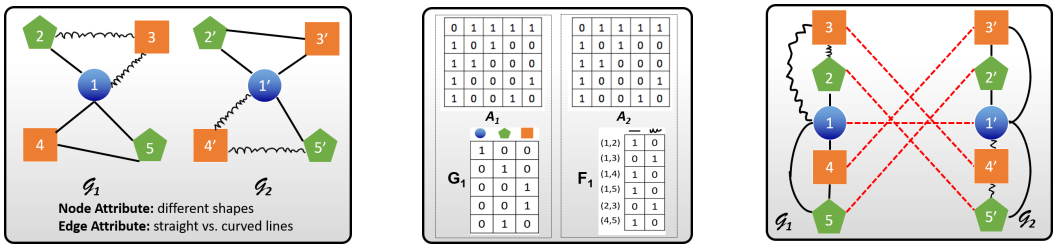
2 Problem Definitions

TABLE 1: Symbols and Notation

Symbols	Definition
$\mathcal{G} = \{\mathbf{A}, \mathbf{G}, \mathbf{F}\}$	an attributed network
$\mathbf{A}$	the adjacency matrix of the network
$\mathbf{G}$	the node attribute matrix of the network
$\mathbf{F}$	the edge attribute matrix of the network
n_1, n_2	# of nodes in $\mathcal{G}_1$ and $\mathcal{G}_2$
m_1, m_2	# of edges in $\mathcal{G}_1$ and $\mathcal{G}_2$
K, L	# of the node and edge attributes
a, b	node/edge indices of $\mathcal{G}_1$
x, y	node/edge indices of $\mathcal{G}_2$
v, w	node-pair indices of the vectorized alignment $\mathbf{s} = \text{vec}(\mathbf{S})$
$k(k'), l$	node/edge label indices
$\mathbf{I}, \mathbf{1}$	an identity matrix and a vector of 1s, respectively
$\mathbf{H}$	$n_2 \times n_1$ prior alignment preference matrix
$\mathbf{S}$	$n_2 \times n_1$ alignment matrix
r, p	reduced ranks
α	the parameter, $0 < \alpha < 1$
$\mathbf{a} = \text{vec}(\mathbf{A})$	vectorize a matrix $\mathbf{A}$ in column order
$\mathbf{Q} = \text{mat}(\mathbf{q}, n_2, n_1)$	reshape vector $\mathbf{q}$ into a $n_2 \times n_1$ matrix in column order
$\widetilde{\mathbf{A}}$	symmetrically normalize matrix $\mathbf{A}$
$\mathbf{D} = \text{diag}(\mathbf{d})$	diagonalize a vector $\mathbf{d}$
$\otimes$	Kronecker product
$\odot$	element-wise matrix product

Table 1 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. For matrix indexing, we use a convention similar to Matlab's syntax as follows. We use $\mathbf{A}(i, j)$ to denote the entry at the intersection of the i-th row and j-th column of matrix $\mathbf{A}$, $\mathbf{A}(i, :)$ to denote the i-th row of $\mathbf{A}$ and $\mathbf{A}(:, j)$ to denote the j-th column of $\mathbf{A}$. We denote the transpose of a matrix by the superscript T (e.g., $\mathbf{A}^T$ is the transpose of $\mathbf{A}$). We use $\widetilde{\ }$ on top to denote the symmetric normalization of a matrix (e.g., $\widetilde{\mathbf{A}} = \mathbf{D}^{-1/2}\mathbf{A}\mathbf{D}^{-1/2}$, where $\mathbf{D}$ is the degree matrix of $\mathbf{A}$). The vectorization of a matrix (in the column order) is denoted by vec(.), and the resulting vector is denoted by the corresponding bold lowercase letter (e.g., $\mathbf{a} = \text{vec}(\mathbf{A})$).

We represent an attributed network by a triplet: $\mathcal{G} = \{\mathbf{A}, \mathbf{G}, \mathbf{F}\}$, where (1) $\mathbf{A}$ is the adjacency matrix, and (2) $\mathbf{G}$ and $\mathbf{F}$ are the node attribute and edge attribute matrices, respectively. The attributes of node-*a* corresponds to the vector of $\mathbf{G}(a, :)$, and $\mathbf{F}_{(a,b)}$ describes the edge attribute vector of the edge between node-*a* and node-*b*. Note that for both node and edge categorical attribute values, they can be transformed into vectors by one hot encoding. Figure 1 presents an illustrative example. We can see from Figure 1(a), the set of nodes (2, 3, 4 and 5) from the first network share the exact same topology with another set of nodes ($2'$, $3'$, $4'$ and $5'$) in the second network. The topology alone would be inadequate to differentiate these two sets. On the other hand, we can see that (1) 2, $2'$, 5 and $5'$ share the same node categorical attribute value; (2) 3, $3'$, 4 and $4'$ share the same node categorical attribute value; and (3) the two edges incident to 3 share the same edge categorical attribute value with those incident to $4'$. These node/edge attributes could provide vital information to establish the accurate node-level alignment (i.e., 2 aligns to $5'$, 5 aligns to $2'$, etc.). This is exactly what this paper aims to address. Formally, the attributed network alignment problem is defined as follows.



(a) Input Attributed Networks. (b) Matrix Representation. (c) Alignment Output.

Fig. 1: An illustrative example of the attributed network alignment problem. (a): two input attributed networks. (b): the matrix representation for attributed networks, where two upper matrices represent the adjacency matrices, and the bottom matrices represent the node attribute and edge attribute matrices of $\mathcal{G}_1$ by using one hot encoding. (c): the desired alignment output (denoted by the red dashed lines).

Problem 1. ATTRIBUTED NETWORK ALIGNMENT.

Given: *(1) two undirected attributed networks* $\mathcal{G}_1 = \{\mathbf{A}_1, \mathbf{G}_1, \mathbf{F}_1\}$ *and* $\mathcal{G}_2 = \{\mathbf{A}_2, \mathbf{G}_2, \mathbf{F}_2\}$ *with* n_1 *and* n_2 *nodes respectively, (2 - optional) a prior alignment preference* $\mathbf{H}$.

Output: *the* $n_2 \times n_1$ *soft 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* $\mathcal{G}_2$.

In the above definition, we have an optional input, to encode the prior knowledge of pairwise alignment preference $\mathbf{H}$, which is an $n_2 \times n_1$ matrix. An entry in $\mathbf{H}$ reflects our prior knowledge of the likelihood to align two corresponding nodes across the two input networks. When such prior knowledge is absent, we set all entries of $\mathbf{H}$ equal, i.e., a uniform distribution. Without loss of generality, we assume that $\mathbf{A}_1$ and $\mathbf{A}_2$ share a comparable size, i.e., $O(n_1) = O(n_2) = O(n)$ and $O(m_1) = O(m_2) = O(m)$. This will also help simplify our complexity analyses.

Notice that the alignment matrix $\mathbf{S}$ in Problem 1 is essentially a *cross-network node similarity* matrix. In some applications, we might be interested in finding a small number of similar nodes in one network w.r.t a query node from the other network. For instance, we might want to find the top-10 most similar `LinkedIn` users for a given `Facebook` user. We could first solve Problem 1 and then return the corresponding row or column in the alignment matrix $\mathbf{S}$, which might be computationally too costly as well as unnecessary. Having this in mind, we further define the on-query attributed network alignment problem as follows:

Problem 2. ON-QUERY ATTRIBUTED ALIGNMENT.

Given: *(1) two undirected attributed networks* $\mathcal{G}_1 = \{\mathbf{A}_1, \mathbf{G}_1, \mathbf{F}_1\}$ *and* $\mathcal{G}_2 = \{\mathbf{A}_2, \mathbf{G}_2, \mathbf{F}_2\}$, *(2 -optional) a prior alignment preference* $\mathbf{H}$, *(3) a query node-a in* $\mathcal{G}_1$.

Output: *an* $n_2 \times 1$ *vector* $\mathbf{s}_a$ *measuring similarities between the query node-a in* $\mathcal{G}_1$ *and all the nodes in* $\mathcal{G}_2$ *efficiently.*

3 TOPOLOGY MEETS ATTRIBUTES

In this section, we present our solutions to Problem 1. We start by formulating Problem 1 as a regularized optimization problem, and then propose effective algorithms to solve it, followed by some theoretical analyses.

3.1 FINAL: Optimization Formulation

The key idea behind our proposed formulation lies in the *alignment consistency* principle, which basically says that the alignments between two pairs of nodes across two input networks should be consistent if these two pairs of nodes themselves are "similar/consistent" with each other. Let us elaborate this using the following example. In Figure 2, we are given two pairs of nodes: (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 $\mathcal{G}_2$. By the *alignment consistency* principle, we require the alignment between a and x, and that between b and y to be consistent (i.e., small $\|\mathbf{S}(x, a) - \mathbf{S}(y, b)\|$), if the following conditions hold:

- C1 *Topology Consistency*. Node a and b are close neighbors in $\mathcal{G}_1$ (i.e., large $\mathbf{A}_1(a, b)$), and x, y are also close neighbors in $\mathcal{G}_2$ (i.e., large $\mathbf{A}_2(x, y)$);
- C2 *Node Attribute Consistency*. Node a and x share the same or similar node attributes, and so do b and y;
- C3 *Edge Attribute Consistency*. Edge (a, b) and (x, y) share the same or similar edge attributes.

The intuition behind the *alignment consistency* principle is as follows. If we already know that node-*a* is aligned to node-*x* (i.e., large $\mathbf{S}(x, a)$), then their close neighbors (e.g., b and y) with same or similar node attributes should have a high chance to be aligned with each other (i.e., large $\mathbf{S}(y, b)$), where we say that b and y are close neighbors of a and x respectively if they are connected by the same or similar edge attributes, with large edge weights (i.e., large $\mathbf{A}_1(a, b)$ and $\mathbf{A}_2(x, y)$). This naturally leads to the following objective function which we wish to minimize in terms of the alignment matrix $\mathbf{S}$:

$$J_1(\mathbf{S}) = \sum_{a,b,x,y} [\frac{\mathbf{S}(x,a)}{\sqrt{f(x,a)}} - \frac{\mathbf{S}(y,b)}{\sqrt{f(y,b)}}]^2 \underbrace{\mathbf{A}_1(a,b)\mathbf{A}_2(x,y)}_{\text{C1: Topology Consistency}} \quad (1)$$
$$\times \underbrace{\Psi(x,a)\Psi(y,b)}_{\text{C2: Node Attribute Consistency}} \times \underbrace{\varphi((x,y),(a,b))}_{\text{C3: Edge Attribute Consistency}}$$

where (1) $a, b = 1, ..., n_1$, and $x, y = 1, ..., n_2$; (2) $\Psi(\cdot)$ is the function that measures the node attribute similarities between two nodes across networks; and (3) $\varphi(\cdot)$ measures the edge attribute similarities between two edges in two networks. Besides, the function $f(\cdot)$ is a node-pair normalization function. For instance, in our paper, we use the following function as $f(x, a)$

$$f(x, a) = \sum_{b,y} \mathbf{A}_1(a, b)\mathbf{A}_2(x, y)\Psi(x, a)\Psi(y, b)\varphi((x, y), (a, b))$$

which measures how many (weighted) neighbor-pairs a and x have that (1) share the same or similar node attributes between themselves (e.g., b and y), and (2) connect to a and x via the same or similar edge attributes, respectively. Note that the functions $\Psi(\cdot)$ and $\varphi(\cdot)$ can be any existing similarity function. In our paper, we use the cosine similarity to measure the similarity between node/edge attributes, i.e.,

$$\Psi(x, a) = (\frac{\mathbf{G}_1(a, :)}{\|\mathbf{G}_1(a, :)\|_2})(\frac{\mathbf{G}_2(x, :)}{\|\mathbf{G}_2(x, :)\|_2})^T$$
$$\varphi((x, y), (a, b)) = (\frac{\mathbf{F}_{1(a,b)}}{\|\mathbf{F}_{1(a,b)}\|_2})(\frac{\mathbf{F}_{2(x,y)}}{\|\mathbf{F}_{2(x,y)}\|_2})^T$$

where $\|\cdot\|_2$ is the vector L2 norm, $\mathbf{G}_1, \mathbf{G}_2$ represent

the node attribute matrices of $\mathcal{G}_1, \mathcal{G}_2$ respectively, and therefore $\mathbf{G}_1(a,:)$ is the feature vector of node-a. Besides, $\mathbf{F}_{1(a,b)}, \mathbf{F}_{2(x,y)}$ are the feature vectors of edge (a,b) in $\mathcal{G}_1$ and (x,y) in $\mathcal{G}_2$. To ease the computation, we denote $\mathbf{N}_1, \mathbf{N}_2$ as the normalized node attribute matrices where, for example, $\mathbf{N}_1(a,:) = \frac{\mathbf{G}_1(a,:)}{\|\mathbf{G}_1(a,:)\|_2}$. Next, we denote the edge feature vectors into a set of matrices. For edges in $\mathcal{G}_1$, we denote $\mathbf{E}_1^l(a,b)$ as the l-th normalized attribute value of the edge (a,b) and $\mathbf{E}_1$ is of same size as $\mathbf{A}_1$, i.e., $\mathbf{E}_1^l(a,b) = \frac{\mathbf{F}_{1(a,b)}(l)}{\|\mathbf{F}_{1(a,b)}\|_2}$. Similarly, we denote $\mathbf{E}_2^l(x,y)$ as the l-th normalized attribute value of edge (x,y). The cosine similarity functions $\Psi(\cdot)$ and $\varphi(\cdot)$ can be re-written as below.

$$\Psi(x,a) = \sum_{k=1}^{K} \mathbf{N}_1(a,k)\mathbf{N}_2(x,k) \tag{2}$$

$$\varphi((x,y),(a,b)) = \sum_{l=1}^{L} \mathbf{E}_1^l(a,b)\mathbf{E}_2^l(x,y) \tag{3}$$

Therefore, the objective function Eq. (1) can be written as

$$J_1(\mathbf{S}) = \sum_{a,b,x,y} [\frac{\mathbf{S}(x,a)}{\sqrt{f(x,a)}} - \frac{\mathbf{S}(y,b)}{\sqrt{f(y,b)}}]^2 \underbrace{\mathbf{A}_1(a,b)\mathbf{A}_2(x,y)}_{\text{C1: Topology Consistency}} \tag{4}$$
$$\times \underbrace{\sum_{k=1}^{K} \mathbf{N}_1(a,k)\mathbf{N}_2(x,k) \sum_{k'=1}^{K} \mathbf{N}_1(b,k')\mathbf{N}_2(y,k')}_{\text{C2: Node Attribute Consistency}}$$
$$\times \underbrace{\sum_{l=1}^{L} \mathbf{E}_1^l(a,b)\mathbf{E}_2^l(x,y)}_{\text{C3: Edge Attribute Consistency}}$$

where

$$f(x,a) = \sum_{b,y} \sum_{k,k'=1}^{K} \sum_{l=1}^{L} \mathbf{A}_1(a,b)\mathbf{A}_2(x,y)\mathbf{N}_1(a,k)\mathbf{N}_2(x,k) \times \mathbf{N}_1(b,k')\mathbf{N}_2(y,k')\mathbf{E}_1^l(a,b)\mathbf{E}_2^l(x,y)$$

Next, we present an equivalent matrix form of J_1, which is more convenient for the following algorithm description and the theoretical proof. By vectorizing the matrix $\mathbf{S}$ (i.e., $\mathbf{s} = \text{vec}(\mathbf{S})$), and with the notation of element-wise product and Kronecker product, Eq. (4) can be rewritten as

$$J_1(\mathbf{s}) = \sum_{v,w} [\frac{\mathbf{s}(v)}{\sqrt{\mathbf{D}(v,v)}} - \frac{\mathbf{s}(w)}{\sqrt{\mathbf{D}(w,w)}}]^2 \mathbf{W}(v,w) \tag{5}$$
$$= \mathbf{s}^T(\mathbf{I} - \widetilde{\mathbf{W}})\mathbf{s}$$

where $v = n_2(a-1)+x$, $w = n_2(b-1)+y$, $\mathbf{W} = \mathbf{N}[\mathbf{E} \odot (\mathbf{A}_1 \otimes \mathbf{A}_2)]\mathbf{N}$ and $\mathbf{N} = \text{diag}(\sum_{k=1}^{K} \mathbf{N}_1(:,k) \otimes \mathbf{N}_2(:,k))$, $\mathbf{E} = \sum_{l=1}^{L} \mathbf{E}_1^l \otimes \mathbf{E}_2^l$. $\widetilde{\mathbf{W}} = \mathbf{D}^{-\frac{1}{2}}\mathbf{W}\mathbf{D}^{-\frac{1}{2}}$ is the symmetric normalized matrix of $\mathbf{W}$. The diagonal degree matrix $\mathbf{D}$ of $\mathbf{W}$ is directly derived from $f(x,a)$ and defined as

$$\mathbf{D} = \mathbf{N}\text{diag}(\sum_{k=1}^{K} \sum_{l=1}^{L} ((\mathbf{E}_1^l \odot \mathbf{A}_1)\mathbf{N}_1(:,k)) \otimes ((\mathbf{E}_2^l \odot \mathbf{A}_2)\mathbf{N}_2(:,k))) \tag{6}$$

Note that some diagonal elements in $\mathbf{D}$ could be zero (e.g., $\mathbf{D}(v,v) = 0$). For such elements, we define the corresponding $\mathbf{D}(v,v)^{-1/2} \triangleq 0$.

In some cases where we want to encode the prior alignment preference matrix $\mathbf{H}$ into the alignment result, we add a regularization term $\|\mathbf{s} - \mathbf{h}\|_2^2$ where $\mathbf{h} = \text{vec}(\mathbf{H})$. When no such prior information is given, we set $\mathbf{h}$ as a uniform column vector. From the optimization perspective, this additional regularization term would also help prevent the trivial solutions, such as a zero alignment matrix $\mathbf{S}$ or the alignment matrix $\mathbf{S}$ where $\mathbf{S}(x,a) = \sqrt{f(x,a)}$.

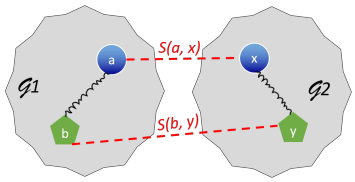


Fig. 2: An illustration of alignment consistency.

Putting everything together, our proposed optimization problem can be stated as follows.

$$\text{argmin}_{\mathbf{s}}\ J(\mathbf{s}) = \alpha \mathbf{s}^T(\mathbf{I} - \widetilde{\mathbf{W}})\mathbf{s} + (1-\alpha)\parallel \mathbf{s} - \mathbf{h} \parallel_2^2 \tag{7}$$

where α is the regularization parameter.

Handling Categorical Attributes. In [13], the authors consider the categorical node and edge attributes. We remark that the formulations in [13] are equivalent to our formulations by using one hot encoding on categorical attributes into vector representations. To be specific, we briefly show that the indicator function on categorical attributes is a special case of cosine similarity. For example, consider countries as node attributes including {China, USA, Canada, Germany} and three nodes from USA, USA, Canada, respectively. The one hot encoding of attribute value USA is represented as vector $(0,1,0,0)$, and that of attribute value Canada is $(0,0,1,0)$. Apparently, only when two nodes are both from USA, the cosine similarity of their node attributes is equal to 1 and otherwise 0, which is equivalent to the indicator function in [13]. Therefore, the formulations in [13] are special cases of the formulations in this paper.

As a result, the objective function Eq. (7) is a more generalized formulation which is capable of handling both numerical and categorical attributes.

3.2 FINAL: Optimization Algorithms

The objective function in Eq. (7) is essentially a quadratic function w.r.t. $\mathbf{s}$. We seek to find its fixed-point solution by setting its derivative to be zero

$$\frac{\partial J(\mathbf{s})}{\partial \mathbf{s}} = 2(\mathbf{I} - \alpha\widetilde{\mathbf{W}})\mathbf{s} + 2(\alpha - 1)\mathbf{h} = 0$$

which leads to the following equation

$$\mathbf{s} = \alpha\widetilde{\mathbf{W}}\mathbf{s} + (1-\alpha)\mathbf{h}$$
$$= \alpha\mathbf{D}^{-\frac{1}{2}}\mathbf{N}(\mathbf{E} \odot (\mathbf{A_1} \otimes \mathbf{A_2}))\mathbf{N}\mathbf{D}^{-\frac{1}{2}}\mathbf{s} + (1-\alpha)\mathbf{h} \tag{8}$$

We could directly develop an iterative algorithm based on Eq. (8). However, such an iterative procedure involves the Kronecker product between $\mathbf{A}_1$ and $\mathbf{A}_2$ whose time complexity is $O(m^2)$. Although the Kronecker product can be pre-computed, the $O(m^2)$ space complexity due to the memory cost and the $O(m^2)$ time complexity in each iteration due to the matrix-vector multiplication are still impractical for large networks.

In order to develop a more efficient algorithm, thanks to a key Kronecker product property (i.e., $\text{vec}(\mathbf{ABC}) = (\mathbf{C}^T \otimes \mathbf{A})\text{vec}(\mathbf{B})$), we re-write Eq. (8) as follows

$$\mathbf{s} = \alpha\mathbf{D}^{-\frac{1}{2}}\mathbf{N}\text{vec}(\sum_{l=1}^{L} (\mathbf{E}_2^l \odot \mathbf{A}_2)\mathbf{Q}(\mathbf{E}_1^l \odot \mathbf{A}_1)^T) + (1-\alpha)\mathbf{h} \tag{9}$$

where $\mathbf{Q}$ is an $n_2 \times n_1$ matrix reshaped by $\mathbf{q} = \mathbf{N}\mathbf{D}^{-\frac{1}{2}}\mathbf{s}$ in column order, i.e., $\mathbf{Q} = \text{mat}(\mathbf{q}, n_2, n_1)$. We can show

that Eq. (8) and Eq. (9) are equivalent with each other. The advantage of Eq. (9) is that it avoids the expensive Kronecker product, which leads to a more efficient iterative algorithm FINAL-NE (summarized in Algorithm 1).

Algorithm 1 FINAL-NE: Attributed Network Alignment.

Input: (1) $\mathcal{G}_1 = \{\mathbf{A}_1, \mathbf{G}_1, \mathbf{F}_1\}$ and $\mathcal{G}_2 = \{\mathbf{A}_2, \mathbf{G}_2, \mathbf{F}_2\}$, (2) optional prior alignment preference $\mathbf{H}$, (3) the regularization parameter α, and (4) the maximum iteration number $t_{\max}$.
Output: the $n_2 \times n_1$ alignment matrix $\mathbf{S}$ between $\mathcal{G}_1$ and $\mathcal{G}_2$.
1: Construct normalized node attribute matrices $\mathbf{N}_1, \mathbf{N}_2$;
2: Construct normalized edge attribute matrices $\mathbf{E}_1^l, \mathbf{E}_2^l$, $l = 1, \cdots, L$;
3: Compute the node attribute matrix $\mathbf{N}$ and degree matrix $\mathbf{D}$;
4: Initiate the alignment $\mathbf{s}$ as a uniform vector, and $t = 1$;
5: **while** $t \leq t_{\max}$ **do**
6: Compute vector $\mathbf{q} = \mathbf{N}\mathbf{D}^{-\frac{1}{2}}\mathbf{s}$;
7: Reshape $\mathbf{q}$ as $\mathbf{Q} = \text{mat}(\mathbf{q}, n_2, n_1)$;
8: Initiate an $n_2 \times n_1$ zero matrix $\mathbf{T}$;
9: **for** $l = 1 \to L$ **do**
10: Update $\mathbf{T} \leftarrow \mathbf{T} + (\mathbf{E}_2^l \odot \mathbf{A}_2)\mathbf{Q}(\mathbf{E}_1^l \odot \mathbf{A}_1)^T$;
11: **end for**
12: Update $\mathbf{s} \leftarrow \alpha\mathbf{D}^{-\frac{1}{2}}\mathbf{N}\text{vec}(\mathbf{T}) + (1-\alpha)\mathbf{h}$;
13: Set $t \leftarrow t + 1$;
14: **end while**
15: Reshape $\mathbf{s}$ to $\mathbf{S} = \text{mat}(\mathbf{s}, n_2, n_1)$.

Variants of FINAL-NE.

Our proposed FINAL-NE algorithm assumes that the input networks have both node and edge attributes. It is worth pointing out that it also works when the node and/or the edge attribute information is missing.

First, when only node attributes are available, we can set all entries in the edge attribute matrices $\mathbf{E}^l$ to 1 where an edge indeed exists. The intuition is that we treat all the edges in the networks to share a common edge attribute value. In this case, the fixed-point solution in Eq. (8) becomes

$$\begin{aligned}\mathbf{s} &= \alpha\mathbf{D}_n^{-\frac{1}{2}}\mathbf{W}_n\mathbf{D}_n^{-\frac{1}{2}}\mathbf{s} + (1-\alpha)\mathbf{h} \\ &= \alpha\mathbf{D}_n^{-\frac{1}{2}}\mathbf{N}(\mathbf{A}_1 \otimes \mathbf{A}_2)\mathbf{N}\mathbf{D}_n^{-\frac{1}{2}}\mathbf{s} + (1-\alpha)\mathbf{h}\end{aligned} \tag{10}$$

where $\mathbf{D}_n = \mathbf{N}\text{diag}(\sum_{k=1}^{K}(\mathbf{A}_1\mathbf{N}_1(:,k)) \otimes (\mathbf{A}_2\mathbf{N}_2(:,k)))$ denotes the degree matrix of $\mathbf{W}_n$. Similar to Eq. (9), we can use the vectorization operator to accelerate the computation. We refer to this variant as FINAL-N, and omit the detailed algorithm description due to space limitation.

Second, when only the edge attributes are available, we treat all nodes to share one common node attribute value by setting $\mathbf{N}$ to be an identity matrix. In this case, the fixed-point solution in Eq. (8) becomes

$$\mathbf{s} = \alpha\mathbf{D}_e^{-\frac{1}{2}}(\mathbf{E} \odot (\mathbf{A}_1 \otimes \mathbf{A}_2))\mathbf{D}_e^{-\frac{1}{2}}\mathbf{s} + (1-\alpha)\mathbf{h} \tag{11}$$

where $\mathbf{D}_e = \text{diag}(\sum_{l=1}^{L}[(\mathbf{E}_1^l \odot \mathbf{A}_1)\mathbf{1}] \otimes [(\mathbf{E}_2^l \odot \mathbf{A}_2)\mathbf{1}])$. Again, we omit the detailed algorithm description due to space, and refer to this variant as FINAL-E.

Finally, if neither the node attributes nor the edge attributes are available, Eq. (8) degenerates to

$$\mathbf{s} = \alpha\mathbf{D}_u^{-\frac{1}{2}}(\mathbf{A}_1 \otimes \mathbf{A}_2)\mathbf{D}_u^{-\frac{1}{2}}\mathbf{s} + (1-\alpha)\mathbf{h} \tag{12}$$

where $\mathbf{D}_u = \mathbf{D}_1 \otimes \mathbf{D}_2$, $\mathbf{D}_1$ and $\mathbf{D}_2$ are the degree matrix of $\mathbf{A}_1$ and $\mathbf{A}_2$. This variant is referred to as FINAL-P.

3.3 Proofs and Analysis

We first analyze the *convergence*, *optimality*, *complexity* and *stability* of our FINAL algorithms. Due to the space limit, we only present the results for the most general case (i.e., FINAL-NE). Then we analyze the relationships between FINAL and several classic graph mining problems.

We start with Lemma 1, which says the FINAL-NE algorithm converges to the global optimal solution of Eq. (7).

Lemma 1. ***Convergence and Optimality of* FINAL-NE**. *Algorithm 1 converges to the closed-form global minimal solution of* $J(\mathbf{s})$*:* $\mathbf{s} = (1-\alpha)(\mathbf{I} - \alpha\widetilde{\mathbf{W}})^{-1}\mathbf{h}$.

Proof. Since $\widetilde{\mathbf{W}}$ is similar to the stochastic matrix $\mathbf{W}\mathbf{D}^{-1} = \mathbf{D}^{\frac{1}{2}}\widetilde{\mathbf{W}}\mathbf{D}^{-\frac{1}{2}}$, the eigenvalues of $\widetilde{\mathbf{W}}$ are within $[-1, 1]$. Given $0 < \alpha < 1$, the eigenvalues of $\alpha\widetilde{\mathbf{W}}$ are in $(-1, 1)$.

We denote the alignment vector $\mathbf{s}$ in the t-th iteration as $\mathbf{s}^{(t)}$. We have that

$$\mathbf{s}^{(t)} = \alpha^t\widetilde{\mathbf{W}}^t\mathbf{h} + (1-\alpha)\sum_{i=0}^{t-1}\alpha^i\widetilde{\mathbf{W}}^i\mathbf{h}$$

Since the eigenvalues of $\alpha\widetilde{\mathbf{W}}$ are in $(-1, 1)$, we have that $\lim_{t\to+\infty}\alpha^t\widetilde{\mathbf{W}}^t = 0$ and $\lim_{t\to+\infty}\sum_{i=0}^{t-1}\alpha^i\widetilde{\mathbf{W}}^i = (\mathbf{I} - \alpha\widetilde{\mathbf{W}})^{-1}$. Putting these together, we have that

$$\mathbf{s} = \lim_{t\to+\infty}\mathbf{s}^{(t)} = (1-\alpha)(\mathbf{I} - \alpha\widetilde{\mathbf{W}})^{-1}\mathbf{h}$$

Next, we prove that the above result is indeed the global minimal solution of the objective function defined in Eq. (7). We prove this by showing that $J(\mathbf{s})$ in Eq. (7) is convex. To see this, we have that the Hessian matrix of Eq. (7) is $\nabla^2 J = 2(\mathbf{I} - \alpha\widetilde{\mathbf{W}})$ with all eigenvalues of $2(\mathbf{I} - \alpha\widetilde{\mathbf{W}})$ greater than 0. In other words, we have that $\nabla^2 J$ is positive definite. Therefore, the objective function defined in Eq. (7) is convex, and its fixed-point solution by Algorithm 1 corresponds to its global minimal solution, which completes the proof. □

The time and space complexity of Algorithm 1 are summarized in Lemma 2. Notice that such a complexity is comparable to the complexity of topology-alone network alignment methods, such as *IsoRank* [4]. In the next section, we will propose an even faster algorithm.

Lemma 2. ***Complexity of* FINAL-NE**. *The time complexity of Algorithm 1 is* $O(Lmnt_{max} + LKn^2)$*, and its space complexity is* $O(n^2)$*. Here,* n *and* m *are the orders of the number of nodes and edges of the input networks, respectively;* K, L *denote the dimension of node and edge feature vectors respectively, and* t_{max} *is the maximum iteration number.*

Proof. It requires $O(nK + mL)$ time and space complexity for line 1-2. To compute $\mathbf{N}$, it takes $O(Kn^2)$ time complexity and $O(n^2)$ space complexity. Then based on Eq. (6), constructing $\mathbf{D}$ requires $O(2m + n^2)KL$ time complexity and $O(n^2)$ space complexity. Line 9-11 takes $O(Lmn)$ time complexity and $O(n^2)$ space complexity. Thus, line 5-14 with $t_{\max}$ iterations takes $O(Lmnt_{\max})$ time complexity and $O(n^2)$ space complexity. In total, the FINAL-NE algorithm takes $O(Lmnt_{\max} + LKn^2)$ time complexity and $O(n^2)$ space complexity. This completes the proof. □

Next, we present the stability analysis of the algorithm FINAL-NE and the analysis on other variants can be easily generalized. Due to the fact that many real-world networks are often noisy, this analysis assists to perceive how robust our proposed algorithms are to the noise/perturbation. Given that the alignment vector $\mathbf{s}$ is the solution to the linear system $(\mathbf{I} - \alpha\widetilde{\mathbf{W}})\mathbf{s} = (1-\alpha)\mathbf{h}$, the stability of FINAL-NE is equivalent to that of the corresponding linear system as:

Lemma 3. ***Stability of* FINAL-NE**. *If the perturbation on the input networks* $\delta = \max\{\|\Delta\mathbf{A}_1\|_F,\ \|\Delta\mathbf{A}_2\|_F\}$ *satisfies the following the conditions*

$$\delta \leq \sqrt{\min\left\{\frac{\epsilon n A}{\alpha B - \epsilon n^2 A}, \sqrt{\frac{\epsilon n A^2}{2\alpha} + \frac{C^2}{4}} - \frac{C}{2}, \frac{\epsilon n^2 A^2}{\alpha A + \alpha n B + \epsilon n^3 A}\right\} + \frac{D^2}{4}} - \frac{D}{2} \tag{13}$$

and

$$\min_i\{d_i\} \leq \frac{\alpha\|\mathbf{E} \odot (\mathbf{A}_1 \otimes \mathbf{A}_2)\|_F}{\epsilon n^2} \tag{14}$$

where $d_i = \mathbf{D}(i,i)$, $A = \min_i\{d_i\}$, $B = \|\mathbf{A}_1\|_F\|\mathbf{A}_2\|_F$, $C = \frac{1}{2}(B + \frac{A}{n} - \frac{\epsilon n^2 A}{\alpha})$, $D = \|\mathbf{A}_1\|_F + \|\mathbf{A}_2\|_F$, *and* $0 < \epsilon < \frac{1-\alpha}{(1+\alpha)n^2}$ *is a constant, the relative error of the system due to the perturbation is bounded by*

$$\frac{\|\hat{\mathbf{s}} - \mathbf{s}\|_2}{\|\mathbf{s}\|_2} \leq \frac{2\epsilon}{1-r}\kappa_F(\mathbf{I} - \alpha\tilde{\mathbf{W}}) < \frac{2\epsilon(1+\alpha)n^2}{1-\alpha-\epsilon(1+\alpha)n^2} \tag{15}$$

where $\kappa_F(\cdot)$ *is the condition number in the Frobenius norm.*

Proof. See the supplementary material. □

Finally, we analyze the relationships between the proposed FINAL algorithms and several classic graph mining problems. Due to the space limit, we omit the detailed proofs and summarize the major findings as follows.

A - FINAL *vs. Node Proximity* An important (single) network mining task is the node proximity, i.e., to measure the proximity/similarity between two nodes on the *same* network. By ignoring the node/edge attributes and setting $\mathbf{A}_1 = \mathbf{A}_2$, our FINAL algorithms, up to a scaling operation $\mathbf{D}^{1/2}$, degenerate to *SimRank* [14] - a prevalent choice for node proximity. Our FINAL algorithms are also closely related to another popular node proximity method, *random walk with restart* [15]. That is, Eq. (8) can be viewed as random walk with restart on the *attributed Kronecker graph* with $\mathbf{h}$ being the starting vector. Note that neither the standard *SimRank* nor *random walk with restart* considers the node or edge attribute information.

B - FINAL *vs. Graph Kernel* The alignment result $\mathbf{s}$ by our FINAL algorithms is closely related to the random walk based graph kernel [16]. To be specific, if $k(\mathcal{G}_1, \mathcal{G}_2)$ is the random walk graph kernel between the two input graphs and $\mathbf{p}$ is the stopping vector, we can show that $k(\mathcal{G}_1, \mathcal{G}_2) = \mathbf{p}^T\mathbf{s}$. This intuitively makes sense, as we can view the graph kernel/similarity as the weighted aggregation (by the stopping vector $\mathbf{p}$) over the pairwise cross-network node similarities (encoded by the alignment vector $\mathbf{s}$). We also remark that in the original random walk graph kernel [16], it mainly focuses on the node attribute information.

C - FINAL *vs. Existing Network Alignment Methods* If we ignore all the node and edge attribute information, our FINAL-P algorithm is equivalent to *IsoRank* [4] by scaling the alignment result and alignment preference by $\mathbf{D}^{1/2}$. We would like to point out that such a scaling operation is important to ensure the convergence of the iterative procedure. Recall that the key idea behind our optimization formulation is the *alignment consistency*. When the attribute information is absent, the *alignment consistency* principle is closely related to the concept of "squares" behind *NetAlign* algorithm [5]. Like most, if not all of the, existing network alignment algorithms, the node or the edge attribute information is ignored in *IsoRank* and *NetAlign*.

We remark that these findings are important in the following two aspects. First, they help establish a quantitative relationship between several, seemingly unrelated graph mining problems, which might in turn help better understand these existing graph mining problems. Second, these findings also have an important algorithmic implication. Take *SimRank* as an example, it was originally designed for plain graphs (i.e., without attributes), and was formulated from random walk perspective and it is not clear what the algorithm tries to optimize. By setting $\mathcal{G}_1 = \mathcal{G}_2$ and ignoring the attribute information, our objective function in Eq. (7) provides a natural way to interpret *SimRank* from an optimization perspective. By setting $\mathcal{G}_1 = \mathcal{G}_2$ alone, our FINAL algorithms can be directly used to measure node proximity on an *attributed* network. Finally, our upcoming FINAL ON-QUERY algorithm also naturally provides an efficient way (i.e., with a linear time complexity) for *on-query SimRank* with or without attribute information (i.e., finding the similarity between a given query node and all the remaining nodes in the same network).

4 SPEED-UP COMPUTATION

In this section, we address the computational issue. To be specific, we will focus on two scenarios. First, to solve Problem 1, our proposed FINAL algorithms in Section 3 have a time complexity of $O(mn)$, where we have dropped the lower order terms. We propose an effective approximate algorithm that reduces the time complexity to $O(n^2)$. Second, for Problem 2, solving the full alignment problem not only still requires $O(n^2)$ time, but also is unnecessary, as we essentially only need a column or a row from the alignment matrix $\mathbf{S}$. To address this issue, we propose an effective algorithm for Problem 2 with a *linear* time complexity. For presentation clarity, we restrict ourselves to the case where there is only node attribute information, although our proposed strategies can be naturally applied to the more general case where we have both node and edge attributes.

4.1 Speed-up FINAL-N

According to Lemma 1, the alignment vector $\mathbf{s}$ in FINAL-N converges to its closed-form solution as follows.

$$\begin{aligned}\mathbf{s} &= (1-\alpha)(\mathbf{I} - \alpha\widetilde{\mathbf{W}}_n)^{-1}\mathbf{h} \\ &= (1-\alpha)(\mathbf{I} - \alpha\mathbf{D}_n^{-\frac{1}{2}}\mathbf{N}(\mathbf{A}_1 \otimes \mathbf{A}_2)\mathbf{N}\mathbf{D}_n^{-\frac{1}{2}})^{-1}\mathbf{h}\end{aligned} \tag{16}$$

The key idea to speed up FINAL-N is to efficiently approximate such a closed-form solution. To be specific, we first approximate the two adjacency matrices by top-r eigenvalue decomposition: $\mathbf{A}_1 = \mathbf{U}_1\mathbf{\Lambda}_1\mathbf{U}_1^T$ and $\mathbf{A}_2 = \mathbf{U}_2\mathbf{\Lambda}_2\mathbf{U}_2^T$. Then the rank-$r$ approximation of $\mathbf{W}_n$ is $\hat{\mathbf{W}}_n = \mathbf{N}(\mathbf{U}_1 \otimes \mathbf{U}_2)(\mathbf{\Lambda}_1 \otimes \mathbf{\Lambda}_2)(\mathbf{U}_1^T \otimes \mathbf{U}_2^T)\mathbf{N}$. Substitute it into Eq. (16), we can approximate the alignment vector $\mathbf{s}$ as

$$\begin{aligned}\mathbf{s} \approx& (1-\alpha)[\mathbf{I} - \alpha\mathbf{D}_n^{-\frac{1}{2}}\mathbf{N}\mathbf{U}(\mathbf{\Lambda}_1 \otimes \mathbf{\Lambda}_2)\mathbf{U}^T\mathbf{N}\mathbf{D}_n^{-\frac{1}{2}}]^{-1}\mathbf{h} \\ =& (1-\alpha)(\mathbf{I} + \alpha\mathbf{D}_n^{-\frac{1}{2}}\mathbf{N}\mathbf{U}\mathbf{\Lambda}\mathbf{U}^T\mathbf{N}\mathbf{D}_n^{-\frac{1}{2}})\mathbf{h}\end{aligned} \tag{17}$$

where $\mathbf{U} = \mathbf{U}_1 \otimes \mathbf{U}_2$, and $\mathbf{\Lambda}$ is an $r^2 \times r^2$ matrix computed by Sherman-Morrison Lemma [17]: $\mathbf{\Lambda} = [(\mathbf{\Lambda}_1 \otimes \mathbf{\Lambda}_2)^{-1} - \alpha(\mathbf{U}_1^T \otimes \mathbf{U}_2^T)\mathbf{N}\mathbf{D}_n^{-1}\mathbf{N}(\mathbf{U}_1 \otimes \mathbf{U}_2)]^{-1}$.

Based on Eq. (17), our proposed FINAL-N+ algorithm is summarized in Algorithm 2. The time complexity of FINAL-N+ is summarized in Lemma 4. Note that we often have $r \ll n, m \ll n^2$ and $K \ll n$. Therefore, compared with FINAL-N, FINAL-N+ is more efficient in time complexity.

Algorithm 2 FINAL-N+: Low-Rank Approximation of FINAL-N.

Input: (1) $\mathcal{G}_1 = \{\mathbf{A}_1, \mathbf{G}_1\}$ and $\mathcal{G}_2 = \{\mathbf{A}_2, \mathbf{G}_2\}$, (2) optional prior alignment preference $\mathbf{H}$, (3) the regularization parameter α, and (4) the rank of eigenvalue decomposition r.
Output: approximate alignment matrix $\mathbf{S}$ between $\mathcal{G}_1$ and $\mathcal{G}_2$.
1: Construct normalized node attribute matrices $\mathbf{N}_1, \mathbf{N}_2$;
2: Construct node attribute matrix $\mathbf{N}$ and degree matrix $\mathbf{D}_n$;
3: Construct alignment preference vector $\mathbf{h} = \text{vec}(\mathbf{H})$;
4: Eigen-decomposition $\mathbf{U}_1\mathbf{\Lambda}_1\mathbf{U}_1^T \leftarrow \mathbf{A}_1$, $\mathbf{U}_2\mathbf{\Lambda}_2\mathbf{U}_2^T \leftarrow \mathbf{A}_2$
5: Compute $\mathbf{U} = \mathbf{U}_1 \otimes \mathbf{U}_2$;
6: Compute $\mathbf{\Lambda} = [(\mathbf{\Lambda}_1 \otimes \mathbf{\Lambda}_2)^{-1} - \alpha\mathbf{U}^T\mathbf{N}\mathbf{D}_n^{-1}\mathbf{N}\mathbf{U}]^{-1}$;
7: Compute $\mathbf{s}$ by Eq. (17);
8: Reshape vector $\mathbf{s}$ to $\mathbf{S} = \text{mat}(\mathbf{s}, n_2, n_1)$.

Lemma 4. ***Time Complexity of* FINAL-N+.** *FINAL-N+ takes $O(n^2r^4 + Kn^2)$ in time where n is the order of the number of nodes, r is the rank of eigenvalue decomposition and K is the dimension of node feature vectors.*

Proof. It requires $O(nK)$ and $O(Kn^2)$ time complexity to compute $\mathbf{N}_1(\mathbf{N}_2)$ and $\mathbf{N}$, respectively. Line 4 takes $O(mr+nr^2)$ time complexity for eigenvalue decomposition on $\mathbf{A}_1, \mathbf{A}_2$. Line 5 takes $O(n^2r^2)$ time complexity. The time complexity of line 6 and line 7 is $O(n^2r^4)$. In total, the time complexity of FINAL-N+ is $O(n^2r^4 + Kn^2)$. □

4.2 Proposed Solution for Problem 2

In Problem 2, we want to find an $n_2 \times 1$ vector $\mathbf{s}_a$ which measures the similarities between the query node-*a* in $\mathcal{G}_1$ and all the n_2 nodes in $\mathcal{G}_2$ (i.e., cross-network similarity search). It is easy to see that $\mathbf{s}_a$ is essentially the a-th column of the alignment matrix $\mathbf{S}$, or equivalently a certain portion of the alignment vector $\mathbf{s}$, i.e., $\mathbf{s}_a = \mathbf{S}(:, a) = \mathbf{s}(v : w)$ where $v = (a-1)n_2 + 1$ and $w = an_2$.

However, if we call FINAL-N or FINAL-N+ to find $\mathbf{S}$ and then return the ranking vector $\mathbf{s}_a$, it would take at least $O(n^2)$ time. To accelerate, we propose an approximate algorithm (FINAL ON-QUERY) which directly finds $\mathbf{s}_a$ in *linear* time, without solving the full alignment matrix $\mathbf{S}$.

We first relax the degree matrix $\mathbf{D}_n$ to its upper-bound $\hat{\mathbf{D}}_n = \mathbf{D}_1 \otimes \mathbf{D}_2$. There are two reasons for taking such a relaxation. First, it would take $O(n^2)$ time to compute the $\mathbf{D}_n$ matrix directly. On the other hand, $\hat{\mathbf{D}}_n$ can be indirectly expressed by the Kronecker product between $\mathbf{D}_1$ and $\mathbf{D}_2$, each of which only takes $O(m)$ time. Second, since $\hat{\mathbf{D}}_n$ is an upper-bound of the $\mathbf{D}_n$ matrix, such a relaxation will not affect the convergence of FINAL-N. By this relaxation, the fixed-point solution in Eq. (10) can be approximated as

$$\mathbf{s} = \alpha\mathbf{N}\hat{\mathbf{D}}_n^{-\frac{1}{2}}(\mathbf{A}_1 \otimes \mathbf{A}_2)\hat{\mathbf{D}}_n^{-\frac{1}{2}}\mathbf{N}\mathbf{s} + (1-\alpha)\mathbf{h} \quad (18)$$

where $\hat{\mathbf{D}}_n = \mathbf{D}_1 \otimes \mathbf{D}_2$.

By a similar procedure in FINAL-N+, the low-rank approximate solution for $\mathbf{s}$ is

$$\mathbf{s} \approx (1-\alpha)\mathbf{h} + \alpha(1-\alpha)\hat{\mathbf{D}}_n^{-\frac{1}{2}}\mathbf{N}\mathbf{U}\hat{\mathbf{\Lambda}}\mathbf{U}^T\mathbf{N}\hat{\mathbf{D}}_n^{-\frac{1}{2}}\mathbf{h} \quad (19)$$

where $\hat{\mathbf{\Lambda}} = [(\mathbf{\Lambda}_1 \otimes \mathbf{\Lambda}_2)^{-1} - \alpha\mathbf{U}^T\mathbf{N}\hat{\mathbf{D}}_n^{-1}\mathbf{U}]^{-1}$.

Since both $\hat{\mathbf{D}}_n$ and $\mathbf{N}$ are diagonal matrices, the ranking vector for node-*a* is

$$\begin{aligned}\mathbf{s}_a =&(1-\alpha)[\mathbf{h}(v:w) + \alpha[\hat{\mathbf{D}}_n^{-\frac{1}{2}}\mathbf{N}\mathbf{U}\hat{\mathbf{\Lambda}}\mathbf{U}^T\mathbf{N}\hat{\mathbf{D}}_n^{-\frac{1}{2}}\mathbf{h}](v:w)] \\ =&(1-\alpha)[\mathbf{H}(:,a) + \alpha\text{diag}(\sum_{k=1}^{K}\mathbf{N}_1(a,k)\mathbf{N}_2(:,k)) \quad (20) \\ &\times(\mathbf{D}_1(a,a)\mathbf{D}_2)^{-\frac{1}{2}}[\underbrace{(\mathbf{U}_1(a,:)\otimes\mathbf{U}_2)}_{O(nr^2)}\ \underbrace{\hat{\mathbf{\Lambda}}}_{O(n^2r^4+r^6)}\ \underbrace{\mathbf{U}^T\mathbf{N}\hat{\mathbf{D}}_N^{-\frac{1}{2}}\mathbf{h}}_{O(n^2r^2)}]\end{aligned}$$

Notice that Eq. (20) still needs $O(n^2)$ time due to the last two terms. We reduce the time cost for computing $\mathbf{g} = \mathbf{U}^T\mathbf{N}\hat{\mathbf{D}}_n^{-\frac{1}{2}}\mathbf{h}$ as follows. First, we take a rank-p singular value decomposition (SVD) on $\mathbf{H}$, i.e., $\mathbf{H} = \sum_{i=1}^{p}\sigma_i\mathbf{u}_i\mathbf{v}_i^T$. Then, by the vectorization operator, we have that

$$\begin{aligned}\mathbf{g} = \sum_{i=1}^{p}\sum_{k=1}^{K}\overbrace{\sigma_i(\mathbf{U}_1^T\text{diag}(\mathbf{N}_1(:,k))\mathbf{D}_1^{-\frac{1}{2}}\mathbf{v}_i)}^{O(nr)} \\ \otimes\underbrace{(\mathbf{U}_2^T\text{diag}(\mathbf{N}_2(:,k))\mathbf{D}_2^{-\frac{1}{2}}\mathbf{u}_i)}_{O(nr)} \quad (21)\end{aligned}$$

We can see that the time cost for Eq. (21) is reduced to $O(pKrn)$, which is linear w.r.t. the number of nodes n.

We reduce the time cost for $\hat{\mathbf{\Lambda}}$ by reformulating as follows, whose time complexity is $O(Knr^2 + Kr^4 + r^6)$

$$\begin{aligned}\hat{\mathbf{\Lambda}} = [\overbrace{(\mathbf{\Lambda}_2\otimes\mathbf{\Lambda}_1)^{-1}}^{O(r^2)} - \alpha\sum_{k=1}^{K}\overbrace{(\mathbf{U}_1^T\text{diag}(\mathbf{N}_1(:,k))\mathbf{D}_1^{-1}\mathbf{U}_1)}^{O(nr^2)} \\ \otimes\underbrace{(\mathbf{U}_2^T\text{diag}(\mathbf{N}_2(:,k))\mathbf{D}_2^{-1}\mathbf{U}_2)}_{O(nr^2)}]^{-1} \quad (22)\end{aligned}$$

Putting everything together, we have

$$\begin{aligned}\mathbf{s}_a =&(1-\alpha)\mathbf{H}(:,a) + \alpha(1-\alpha)\text{diag}(\sum_{k=1}^{K}\mathbf{N}_1(a,k)\mathbf{N}_2(:,k)) \quad (23) \\ &\times\underbrace{(\mathbf{D}_1(a,a)\mathbf{D}_2)^{-\frac{1}{2}}}_{O(n)}[\underbrace{(\mathbf{U}_1(a,:)\otimes\mathbf{U}_2)}_{O(nr^2)}\ \underbrace{\hat{\mathbf{\Lambda}}}_{O(Knr^2+Kr^4+r^6)}\ \underbrace{\mathbf{g}}_{O(pKnr)}]\end{aligned}$$

Based on Eq. (23), our proposed FINAL ON-QUERY algorithm is summarized in Algorithm 3. The time complexity of FINAL ON-QUERY is summarized in Lemma 5. Notice that we often have $r, p \ll n$, $m_H \ll m \ll n^2$ and $K \ll n$. FINAL ON-QUERY has a *linear* time complexity w.r.t the size of the input network, which is much more scalable than both FINAL-N and FINAL-N+.

Algorithm 3 FINAL ON-QUERY: Approximate On-Query Algorithm for Node Attributed Networks.

Input: (1) $\mathcal{G}_1 = \{\mathbf{A}_1, \mathbf{G}_1\}$ and $\mathcal{G}_2 = \{\mathbf{A}_2, \mathbf{G}_2\}$, (2) optional prior alignment preference $\mathbf{H}$, (3) the regularization parameter α, (4) the rank of eigenvalue decomposition r, and (5) the rank of SVD for $\mathbf{H}$ p.
Output: approximate ranking vector $\mathbf{s}_a$ between node-*a* in $\mathcal{G}_1$ and all nodes in $\mathcal{G}_2$.
Pre-Compute:
1: Compute degree matrices $\mathbf{D}_1$ and $\mathbf{D}_2$;
2: Rank r eigenvalue decomposition $\mathbf{U}_1\mathbf{\Lambda}_1\mathbf{U}_1^T \leftarrow \mathbf{A}_1$;
3: Rank r eigenvalue decomposition $\mathbf{U}_2\mathbf{\Lambda}_2\mathbf{U}_2^T \leftarrow \mathbf{A}_2$;
4: Rank p singular value decomposition $\sum_{i=1}^{p}\sigma_i\mathbf{u}_i\mathbf{v}_i^T \leftarrow \mathbf{H}$;
5: Compute $\mathbf{g}$ by Eq. (21);
6: Compute $\hat{\mathbf{\Lambda}}$ by Eq. (22);
Online-Query:
7: Compute $\mathbf{s}_a$ by Eq. (23).

Lemma 5. Time complexity of FINAL ON-QUERY. *The time complexity of* FINAL ON-QUERY *is $O(r^6 + mr + nr^2 + m_Hp + np^2 + Knr^2 + Kr^4 + pKnr)$ where n, m are the orders of the number of nodes and edges respectively, r, p is the rank of eigenvalue decomposition and SVD, K is the number of node attributes and m_H is the number of non-zero elements in $\mathbf{H}$.*

Proof. The eigenvalue decomposition of $\mathbf{A}_1$ and $\mathbf{A}_2$ takes $O(mr + nr^2)$. The complexity of SVD in line 4 requires $O(m_Hp+np^2)$ time. As Eq. (23) shows, its time complexity is $O(Knr^2+Kr^4+r^6+pKnr)$. Thus the total time complexity

of Algorithm 3 is $O(r^6 + mr + nr^2 + m_H p + np^2 + Knr^2 + Kr^4 + pKnr)$. □

5 Experimental Results

In this section, we present the experimental results and analysis of our proposed algorithms FINAL. The experiments are designed to evaluate the following aspects:

- *Effectiveness*: How accurate are our algorithms for aligning attributed networks?
- *Efficiency*: How fast are our proposed algorithms?

5.1 Experimental Setup

Datasets. We evaluate our proposed algorithms on eight real-world attributed networks.

- *Co-Authorship Network:* This dataset contains 42,252 nodes and 210,320 edges [18]. Each author has a feature vector which represents the number of publications of the author in each of 29 major conferences.
- *Douban:* This Douban dataset was collected in 2010 and contains $50k$ users and $5M$ edges [19]. Each user has rich information, such as the location and offline event participation. Each edge has an attribute representing whether two users are contacts or friends.
- *Flickr:* This dataset was collected in 2014 and consists of 215,495 users and 9,114,557 friend relationships. Users have detailed profile information, such as gender, hometown and occupation, each of which can be treated as the node attributes [9].
- *Lastfm:* This dataset was collected in 2013 and contains 136,420 users and 1,685,524 following relationships [9]. A detailed profile of some users is also provided, including gender, age and location, etc.
- *Myspace:* This dataset contains 854,498 users and 6,489, 736 relationships. The profile of users includes gender, hometown and religion, etc. [9].
- *ACM Citation:* This dataset was collected in 2016 and it contains 2,381,688 papers. Each paper has a list of authors as well as the venue of the paper.
- *DBLP Citation:* This dataset was collected in 2016 and it contains 3,272,991 papers. Each paper has a list of authors as well as its venue.
- *ArnetMiner:* ArnetMiner dataset consists of the information up to year 2013. The whole dataset has 1,053,188 nodes and 3,916,907 undirected edges [9].

Based on these datasets, we construct the following six alignment scenarios for evaluations.

S1. Co-Authorship vs. Co-Authorship. We extract a subnetwork with 9,143 users/nodes from the original dataset, together with their publications in each conference. We randomly permute this subnetwork with noisy edge weights and treat it as the second network. We choose the most active conference of a given author as the node attribute, i.e., the conference with the most publications. We construct the prior alignment preference $\mathbf{H}$ based on the node degree similarity. For this scenario, the prior alignment matrix $\mathbf{H}$ alone leads to a very poor alignment result, with only 0.6% one-to-one alignment accuracy.

S2. Douban Online vs. Douban Offline. We construct an alignment scenario for *Douban* dataset in the same way as [19]. We construct the offline network according to users' co-occurrence in social gatherings. We treat people as (1) 'contacts' of each other if they participate in the same offline events more than ten times but less than twenty times, and (2) 'friends' if they co-participate in more than twenty social gatherings. The constructed offline network has 1,118 users and we extract a subnetwork with 3,906 nodes from the provided online network that contains all these offline users. We treat the location of a user as the node attribute, and 'contacts'/'friends' as the edge attribute. We use the degree similarity to construct the prior alignment preference $\mathbf{H}$ which itself leads to 7.07% one-to-one alignment accuracy.

S3. Flickr vs. Lastfm. We have the partial ground-truth alignment for these two datasets [9]. We extract the subnetworks from them that contain the given ground-truth nodes. The two subnetworks have 12,974 nodes and 15,436 nodes, respectively. We consider the gender of a user as node attribute. For those users with the missing information of gender, we treat them as 'unknown'. Same as [9], we sort nodes by their pagerank scores and label 1% highest nodes as 'opinion leaders', the next 10% nodes as 'middle class' and remaining nodes as 'ordinary users'. Edges are attributed by the level of people they connect to (e.g., leader with leader). We use the username similarity as the prior alignment preference by the Jaro-Winkler distance [20]. The username similarity alone can correctly align 61.50% users.

S4. Flickr-Myspace. We have the partial ground-truth alignment for these two datasets. We extract two subnetworks that contain these ground-truth nodes. The subnetwork of *Flickr* has 6,714 nodes and the subnetwork of *Myspace* has 10,733 nodes. We use the same way as *S3* for node attributes, edge attributes and the prior alignment preference. The username similarity achieves 61.80% accuracy.

S5. ACM-DBLP Co-authorship. We extract from both datasets the papers that are published in four areas, including data mining, machine learning, database and information retrieval/web mining. We construct the co-authorship networks based on each paper's co-author relationship. That is, if two authors co-author a paper in above areas, then we link this two authors in the co-authorship network. Then, we extract from the two constructed co-authorship networks the subgraphs that contain 9,872 nodes and 39,561 edges in ACM co-authorship network and 9,916 nodes and 44,808 edges in DBLP co-authorship network, respectively. Besides, we consider both numerical and categorical attributes. For numerical node attributes, we treat the number of papers of an author in each of the 17 venues as an attribute, which leads to a 17 dimensional feature vector. We use the four areas as the node categorical attributes. That is, for example, if an author published the most papers in data mining area, then we label this node as 'data mining'. We consider categorical edge attributes, and similarly, we use the area where two authors mostly collaborate with each other. We use the degree similarity matrix as the prior alignment preference which alone can only correctly align 20.76% users.

S5. ArnetMiner-ArnetMiner. We use the same method as *S1* to construct the alignment scenario as well as the prior alignment preference. This scenario contains the largest networks, and therefore is used for efficiency evaluations.

Comparison Methods. For the proposed FINAL algorithms, we test the following variants, including (1) FINAL-NE with categorical node and edge attributes; (2) FINAL-NE(N) with numerical node and edge attributes; (3) FINAL-N with categorical node attributes; (4) FINAL-

N(N) with numerical node attributes; (5) FINAL-E with categorical edge attributes; (6) FINAL-N+, a low-rank approximation of FINAL-N. We compare them with the following existing network alignment algorithms including (1) *IsoRank* [4], (2) *NetAlign* [5], (3) *UniAlign* [21], (4) *Klau's Algorithm* [6], (5) *HubAlign* [22] and (6) *RRWM* [23].

Machines and Repeatability. Experiments are performed on a Windows machine with four 3.6GHz Intel Cores and 32G RAM. The algorithms are programmed with MATLAB[1].

5.2 Effectiveness Analysis

We first evaluate the impact of the permutation noise on the alignment accuracy. We use a heuristic greedy matching algorithm [24] as a post-processing step on the similarity matrix to obtain the one-to-one alignments between the two input networks, and then compute the alignment accuracy with respect to the ground-truth. The results are summarized in Figure 3. We have the following observations. First, all of our proposed methods outperform the existing alignment methods. Specifically, our FINAL algorithms achieve an up to 30% improvement in terms of the alignment accuracy over the comparison methods. Second, FINAL-N and FINAL-E both outperform the existing methods in most scenarios, yet are not as good as FINAL-NE, suggesting that node attributes and edge attributes might be complementary in terms of improving the alignment accuracy. Third, the alignment accuracy of FINAL-N+ is very close to its exact counterpart FINAL-N (i.e., with a 95% accuracy compared with FINAL-N). Fourth, by jointly considering the attributes and the topology of networks, our methods are more resilient to the permutation noise. Moreover, for the two networks whose topologies are dramatically different from each other (e.g., Douban online-offline networks), the accuracy gap between FINAL-N+ and the existing methods is even bigger (Figure 3(b)). This is because in this case, the topology information alone (*IsoRank*, *NetAlign*, *Klau* and *HubAlign*) could actually mislead the alignment process. Finally, as Figure 3(c) shows, using numerical attributes (i.e., FINAL-NE(N) and FINAL-N(N)) could further improve the performance of FINAL-NE with categorical attributes.

Second, we evaluate the impact of the noise in the prior alignment preference (i.e., $\mathbf{H}$) on the alignment results, which is summarized in Figure 4. As expected, a higher noise in $\mathbf{H}$ has more negative impacts on the alignment accuracy for most of the methods. Nonetheless, our FINAL algorithms still consistently outperform all other four existing methods across different noise levels.

In addition, we conduct the comparisons between the proposed FINAL algorithm and other baseline methods to show the alignment performance by using various information. For *RRWM*, we compute the cosine similarity matrix of the node attributes and edge attributes respectively, then combine them with the Kronecker product of the adjacency matrices to form the affinity matrix [11]. For the rest of comparison scenarios, we calculate the cosine similarity values among node attributes as the node similarities across networks, which will then be used as the prior alignment matrix $\mathbf{H}$ of the baseline methods (named as *NodeSim*). Similarly, the average between the node attribute similarity matrix and the originally designed prior matrix (used in Figure 3 and Figure 4), is considered as the prior matrix (named as *Hybrid*). The results are summarized in Table 2. First, we observe that given the exact same set of information, the proposed FINAL-NE outperforms the *RRWM* algorithm in terms of the alignment accuracy. This indicates even with the more rigorous constraints of the optimization problem in *RRWM*, the intricacy of solving the problem itself might mislead the alignment solution. Second, we observe that given the node attributes and prior alignment matrix, our proposed FINAL-N method outperforms other baseline methods. This indicates although the additive combination of attributes and the prior knowledge could lead to an improvement within the baseline methods themselves, our algorithms still achieve a better performance due to the *alignment consistency*.

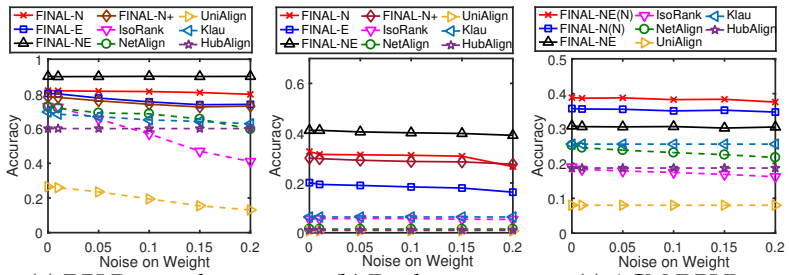
1. The source code of our algorithms can be downloaded here: http://www.public.asu.edu/~szhan172/FINAL-KDD16.zip.

5.3 Efficiency Analysis

Quality-Speed Trade-off. We first evaluate how different methods balance the alignment accuracy and the running time for the full network alignment problem (i.e., Problem 1). The results are summarized in Figure 5. Note that the results of *RRWM* are not included here because it takes days to finish the computation, which is not even comparable with other methods. As we can see, the running time of our proposed exact methods is only slightly higher than its topology-alone counterpart (i.e., *IsoRank*), and in the meanwhile, they all achieve a 10%-20% accuracy improvement. FINAL-NE(N) and FINAL-N(N) are faster than FINAL-NE in Figure 5 (c) because using numerical attributes could lead to a faster convergence. Besides, FINAL-N+ and *UniAlign* are the fastest, yet the proposed FINAL-N+ produces a much higher alignment accuracy. We do not show the balance of *Klau's Algorithm* in Figure 5 (a) and (b), because the running time is usually several hours which is not comparable with other methods. For *NetAlign* and *Klau's Algorithm*, we observe that they take much longer running time when the input prior alignment preference matrix $\mathbf{H}$ is not sparse enough, as it involves a time-consuming Hungarian step during each iteration.

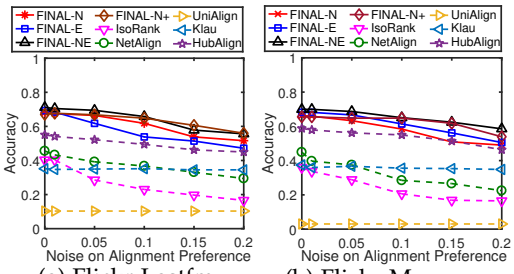
Second, we evaluate the quality-speed trade-off for on-query alignment problem. Here, we treat the top-10 ranking results by FINAL-N as the ground-truth, and compare the average ranking accuracy of 500 random nodes with two proposed approximate algorithms (FINAL-N+ and FINAL ON-QUERY). The results are summarized in Figure 6. We observe that (1) FINAL-N+ preserves a 95% ranking accuracy, with a more than $10\times$ speedup over FINAL-N, (2) FINAL ON-QUERY preserves an about 90% ranking accuracy, and it is $100\times$ faster than the exact FINAL-N.

Scalability. We first evaluate the scalability of FINAL-N+, which is summarized in Figure 7. We can see that the running time is quadratic w.r.t the number of nodes of the input networks, which is consistent with the time complexity results in Lemma 4. Second, we evaluate the scalability of FINAL ON-QUERY, for both its pre-compute phase and online-query phase. As we can see from Figure 8, the running time is *linear* w.r.t the number of nodes in both stages, which is consistent with Lemma 5. In addition,



(a) DBLP co-author. (b) Douban. (c) ACM-DBLP.

Fig. 3: (Higher is better.) Alignment accuracy vs. the noise level in networks. ($t_{\max} = 30$, $r = 5$).

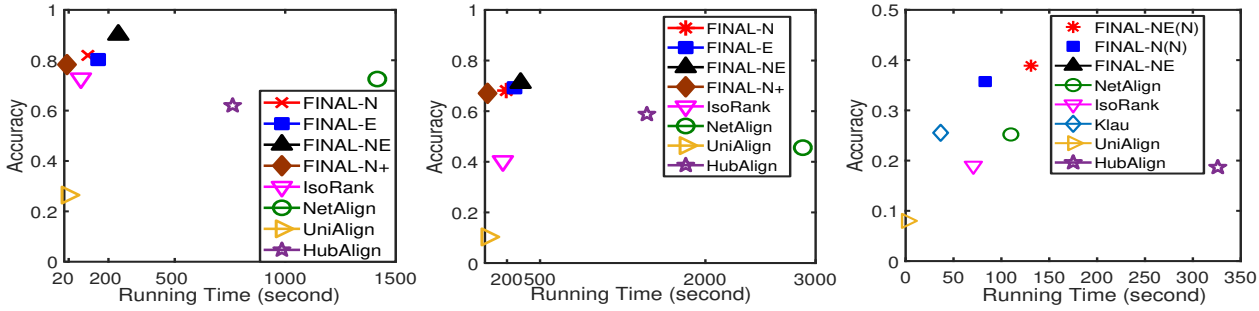


(a) Flickr-Lastfm. (b) Flickr-Myspace.

Fig. 4: (Higher is better.) Alignment accuracy vs. the noise level in $\mathbf{H}$. ($t_{\max} = 30$, $r = 5$).

	FINAL-NE	FINAL-N	RRWM	IsoRank			NetAlign			Klau			HubAlign		
				Original	NodeSim	Hybrid	Original	NodeSim	Hybrid	Original	NodeSim	Hybrid	Original	NodeSim	Hybrid
DBLP-Coauthor	**0.9011**	**0.8193**	0.8732	0.7281	0.8097	0.5568	0.7254	0.7121	0.8179	0.6961	0.4800	0.5075	0.6201	0.6313	0.6127
Douban	**0.4114**	**0.3265**	0.0420	0.0546	0.1404	0.3050	0.0150	0.2066	0.0242	0.0635	0.2299	0.0564	0.0106	0.0206	0.0528
ACM-DBLP	**0.3889**	**0.3573**	0.3300	0.1897	0.1777	0.3108	0.2520	0.2879	0.3368	0.2553	0.1994	0.2588	0.1867	0.1325	0.2606
Flickr-Lastfm	**0.7114**	**0.6814**	0.6416	0.4027	0.2987	0.6345	0.4558	0.2345	0.5376	0.3518	0.2832	0.5155	0.5878	0.3097	0.6216
Flickr-Myspace	**0.6992**	**0.6629**	0.6779	0.3596	0.2509	0.5431	0.4494	0.2060	0.3708	0.3745	0.2022	0.4082	0.5506	0.1573	0.4307

TABLE 2: Alignment with different alignment prior matrices.



(a) DBLP co-author networks. (b) Flickr-Lastfm networks. (c) ACM-DBLP co-author networks.

Fig. 5: Balance between the accuracy and the speed. $t_{\max} = 30$, $r = 5$.

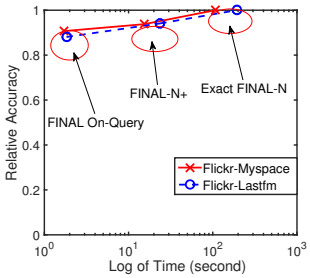


Fig. 6: Balance of on-query alignment.

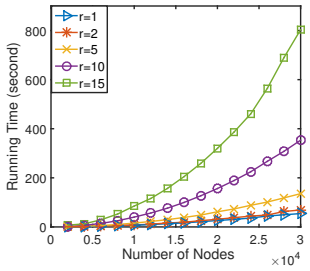
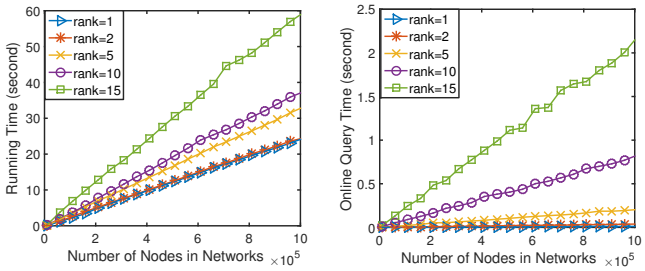


Fig. 7: Scalability of FINAL-N+.

the actual online-query time on the entire ArnetMiner data set (with $r = 10$) is less than 1 second, suggesting that the proposed FINAL ON-QUERY method might be well suitable for the real-time query response.



(a) Pre-compute phase. (b) Online-query phase.

Fig. 8: Scalability of FINAL ON-QUERY.

6 RELATED WORK

The network alignment has attracted lots of research interests with extensive literatures. It appears in numerous domains, ranging from bioinformatics [4], [22], [25], computer vision [11], [12], to data mining [21], [26].

Network Alignment. A classic alignment approach can be attributed to *IsoRank* algorithm [4], which is in turn inspired by PageRank [27]. The original *IsoRank* algorithm propagates the pairwise node similarity in the Kronecker product graph. Several approximate algorithms have been proposed to speed up its computation. *IsoRankN* [25] extends the original *IsoRank* algorithm and uses a similar approach as PageRank-Nibble [28] to align multiple networks.

Bayati et al. [29] propose a maximum weight matching algorithm for graph alignment using the max-product belief propagation [30]. Bradde et al. [31] propose another distributed message-passing algorithm based on belief propagation for protein-protein interaction network alignment. Besides, *NetAlign* [5] is proposed by formulating the network alignment problem as an integer quadratic programing problem to maximize the number of "squares". *BigAlign* formulates the bipartite network alignment problem and uses the alternating projected gradient descent to solve it [21]. Zhang et al. solve the multiple anonymized network alignment in two steps, i.e., unsupervised anchor link inference and transitive multi-network matching [32]. Other multiple network alignment works include [33], [34], [35]. More recently, Liu et al. leverages the network embedding techniques for network alignment [36]. Moreover, [37] studied the partial co-alignment problem which is to align both users and their locations simultaneously. Mohammadi et al. propose a tensor-based network alignment for higher-order network alignment [38]. *MAGNA++* is another state-

of-the-arts that simultaneously maximizes the node conservation and edge conservation used in bioinformatics [39]. Chen et.al propose a community-based nework alignment method that can not only find the node-level alignment, but also the alignment among the communities across networks [40]. [41] proves that the alignment matrix has a low rank structure thanks to the low rank characteristics of many real-world networks. Compared with these network alignment methods, our FINAL algorithms is more generalizable to handle the consistency among both topology and the numerical/categorical node and/or edge attributes. In a closely related thread, most of the graph matching algorithms can naturally use both the node and edge attributes and the graph topology [11], [12]. However, these algorithms cannot scale well to large-scale networks.

Anchor Link Inference. A related problem is to identify users from multiple social networks (i.e., the cross-site user identification problem). Early works include the user profile based matching [42], [43] and the network structure based matching methods [44]. For example, [42] identifies the users across different social platforms by combining both the usernames and the tags in the user profiles. Tan et al. [45] propose a subspace learning method, which models user relationship by a hypergraph. However, the lack of the combinations of the node attributes and the structural information often leads to inaccurate results. To alleviate this issue, Zafarani et al. identify users by modeling user behavior patterns based on human limitations, exogenous and endogenous factors [8]. Liu et al. propose a method to identify same users by behavior modeling, structure consistency modeling and learning by multi-objective optimization [46]. *COSNET* [9] considers both local the global consistency and uses an energy-based model to find connections among multiple heterogeneous networks. Besides, *CLF* collectively predicts the anchor and social links, and then propagate the predicted links across partially aligned probabilistic networks [47]. Man et al. propose a network embedding based method for anchor link prediction, which leverages the observed anchor links to capture the specific regularities and then learns the stable cross-network mappings [48]. However, all these methods require the pre-aligned nodes.

7 Conclusion

In this paper, we study the attributed network alignment problem, including the full alignment version as well as its on-query variant. To address these problem, we formulate our generalized alignment consistency principle into a quadratic optimization problem. We first introduce our proposed exact algorithms FINAL that can align across attributed networks with a provable optimality, convergence and stability, and with a comparable complexity with their topology-alone counterparts. We then propose (1) an approximate alignment algorithm (FINAL-N+) to further reduce the time complexity, and (2) an effective alignment algorithm (FINAL On-Query) to solve the on-query network alignment problem with a *linear* time complexity. We conduct extensive empirical evaluations on real networks, which demonstrate the efficacy of our proposed algorithms.

8 Acknowledgement

This material is supported by the National Science Foundation under Grant No. IIS-1651203, IIS-1715385, IIS-1743040, and CNS-1629888, by DTRA under the grant number HDTRA1-16-0017, by the United States Air Force and DARPA under contract number FA8750-17-C-0153, by Army Research Office under the contract number W911NF-16-1-0168, and by the U.S. Department of Homeland Security under Grant Award Number 2017-ST-061-QA0001. The content of the information in this document does not necessarily reflect the position or the policy of the Government, and no official endorsement should be inferred. The U.S. Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation here on.

References

[1] J. Ni, H. Tong, W. Fan, and X. Zhang, "Inside the atoms: ranking on a network of networks," in *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2014, pp. 1356–1365.

[2] V. Van Vlasselaer, C. Bravo, O. Caelen, T. Eliassi-Rad, L. Akoglu, M. Snoeck, and B. Baesens, "Apate: A novel approach for automated credit card transaction fraud detection using network-based extensions," *Decision Support Systems*, vol. 75, pp. 38–48, 2015.

[3] F. Emmert-Streib, M. Dehmer, and Y. Shi, "Fifty years of graph matching, network alignment and network comparison," *Information Sciences*, vol. 346, pp. 180–197, 2016.

[4] 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*, 2008.

[5] M. Bayati, M. Gerritsen, D. F. Gleich, A. Saberi, and Y. Wang, "Algorithms for large, sparse network alignment problems," in *Data Mining, 2009. ICDM'09. Ninth IEEE International Conference on*. IEEE, 2009, pp. 705–710.

[6] G. W. Klau, "A new graph-based method for pairwise global network alignment," *BMC bioinformatics*, vol. 10, no. 1, p. S59, 2009.

[7] S. Bartunov, A. Korshunov, S.-T. Park, W. Ryu, and H. Lee, "Joint link-attribute user identity resolution in online social networks," in *Proceedings of the 6th International Conference on Knowledge Discovery and Data Mining, Workshop on Social Network Mining and Analysis. ACM*, 2012.

[8] 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.

[9] 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.

[10] X. Kong, J. Zhang, and P. S. Yu, "Inferring anchor links across multiple heterogeneous social networks," in *Proceedings of the 22nd ACM international conference on Information & Knowledge Management*. ACM, 2013, pp. 179–188.

[11] F. Zhou and F. De la Torre, "Factorized graph matching," in *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*. IEEE, 2012, pp. 127–134.

[12] ——, "Deformable graph matching," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2013, pp. 2922–2929.

[13] S. Zhang and H. Tong, "Final: Fast attributed network alignment," in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 2016, pp. 1345–1354.

[14] C. Li, J. Han, G. He, X. Jin, Y. Sun, Y. Yu, and T. Wu, "Fast computation of simrank for static and dynamic information networks," in *Proceedings of the 13th International Conference on Extending Database Technology*. ACM, 2010, pp. 465–476.

[15] H. Tong, C. Faloutsos, and J.-Y. Pan, "Fast random walk with restart and its applications," in *Sixth International Conference on Data Mining (ICDM'06)*. IEEE, 2006, pp. 613–622.

[16] S. V. N. Vishwanathan, N. N. Schraudolph, R. Kondor, and K. M. Borgwardt, "Graph kernels," *Journal of Machine Learning Research*, vol. 11, no. Apr, pp. 1201–1242, 2010.

[17] W. W. Piegorsch and G. Casella, "Erratum: inverting a sum of matrices," *SIAM review*, vol. 32, no. 3, pp. 470–470, 1990.

[18] A. Prado, M. Plantevit, C. Robardet, and J.-F. Boulicaut, "Mining graph topological patterns: Finding covariations among vertex descriptors," *IEEE Transactions on Knowledge and Data Engineering*, vol. 25, no. 9, pp. 2090–2104, 2013.
[19] E. Zhong, W. Fan, J. Wang, L. Xiao, and Y. Li, "Comsoc: adaptive transfer of user behaviors over composite social network," in *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2012, pp. 696–704.
[20] W. Cohen, P. Ravikumar, and S. Fienberg, "A comparison of string metrics for matching names and records," in *Kdd workshop on data cleaning and object consolidation*, vol. 3, 2003, pp. 73–78.
[21] 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.
[22] S. Hashemifar and J. Xu, "Hubalign: an accurate and efficient method for global alignment of protein–protein interaction networks," *Bioinformatics*, vol. 30, no. 17, pp. i438–i444, 2014.
[23] M. Cho, J. Lee, and K. M. Lee, "Reweighted random walks for graph matching," in *European conference on Computer vision*. Springer, 2010, pp. 492–505.
[24] G. Kollias, S. Mohammadi, and A. Grama, "Network similarity decomposition (nsd): A fast and scalable approach to network alignment," *IEEE Transactions on Knowledge and Data Engineering*, vol. 24, no. 12, pp. 2232–2243, 2012.
[25] 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.
[26] B. Du, S. Zhang, N. Cao, and H. Tong, "First: Fast interactive attributed subgraph matching," in *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 2017, pp. 1447–1456.
[27] L. Page, S. Brin, R. Motwani, and T. Winograd, "The pagerank citation ranking: Bringing order to the web." Stanford InfoLab, Tech. Rep., 1999.
[28] R. Andersen, F. Chung, and K. Lang, "Local graph partitioning using pagerank vectors," in *null*. IEEE, 2006, pp. 475–486.
[29] M. Bayati, D. Shah, and M. Sharma, "Maximum weight matching via max-product belief propagation," *arXiv preprint cs/0508101*, 2005.
[30] J. S. Yedidia, W. T. Freeman, and Y. Weiss, "Understanding belief propagation and its generalizations," *Exploring artificial intelligence in the new millennium*, vol. 8, pp. 236–239, 2003.
[31] S. Bradde, A. Braunstein, H. Mahmoudi, F. Tria, M. Weigt, and R. Zecchina, "Aligning graphs and finding substructures by a cavity approach," *EPL (Europhysics Letters)*, vol. 89, no. 3, p. 37009, 2010.
[32] 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.
[33] V. Vijayan and T. Milenkovi, "Multiple network alignment via multimagna++," *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, pp. 1–1, 2018.
[34] V. Vijayan, E. Krebs, L. Meng, and T. Milenkovic, "Pairwise versus multiple network alignment," *arXiv preprint arXiv:1709.04564*, 2017.
[35] E. Malmi, S. Chawla, and A. Gionis, "Lagrangian relaxations for multiple network alignment," *Data Mining and Knowledge Discovery*, vol. 31, no. 5, pp. 1331–1358, 2017.
[36] L. Liu, W. K. Cheung, X. Li, and L. Liao, "Aligning users across social networks using network embedding." in *IJCAI*, 2016, pp. 1774–1780.
[37] J. Zhang and P. S. Yu, "Pct: partial co-alignment of social networks," in *Proceedings of the 25th International Conference on World Wide Web*. International World Wide Web Conferences Steering Committee, 2016, pp. 749–759.
[38] S. Mohammadi, D. F. Gleich, T. G. Kolda, and A. Grama, "Triangular alignment tame: A tensor-based approach for higher-order network alignment," *IEEE/ACM Transactions on Computational Biology and Bioinformatics (TCBB)*, vol. 14, no. 6, pp. 1446–1458, 2017.
[39] V. Vijayan, V. Saraph, and T. Milenković, "Magna++: Maximizing accuracy in global network alignment via both node and edge conservation," *Bioinformatics*, vol. 31, no. 14, pp. 2409–2411, 2015.
[40] Z. Chen, X. Yu, B. Song, J. Gao, X. Hu, and W.-S. Yang, "Community-based network alignment for large attributed network," in *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*. ACM, 2017, pp. 587–596.
[41] S. Zhang, H. Tong, J. Tang, J. Xu, and W. Fan, "ineat: Incomplete network alignment," in *Data Mining (ICDM), 2017 IEEE International Conference on*. IEEE, 2017, pp. 1189–1194.
[42] T. Iofciu, P. Fankhauser, F. Abel, and K. Bischoff, "Identifying users across social tagging systems." in *ICWSM*, 2011.
[43] M. Motoyama and G. Varghese, "I seek you: searching and matching individuals in social networks," in *Proceedings of the eleventh international workshop on Web information and data management*. ACM, 2009, pp. 67–75.
[44] N. Korula and S. Lattanzi, "An efficient reconciliation algorithm for social networks," *Proceedings of the VLDB Endowment*, vol. 7, no. 5, pp. 377–388, 2014.
[45] S. Tan, Z. Guan, D. Cai, X. Qin, J. Bu, and C. Chen, "Mapping users across networks by manifold alignment on hypergraph." in *AAAI*, vol. 14, 2014, pp. 159–165.
[46] S. Liu, S. Wang, F. Zhu, J. Zhang, and R. Krishnan, "Hydra: Large-scale social identity linkage via heterogeneous behavior modeling," in *Proceedings of the 2014 ACM SIGMOD international conference on Management of data*. ACM, 2014, pp. 51–62.
[47] J. Zhang and S. Y. Philip, "Integrated anchor and social link predictions across social networks." in *IJCAI*, 2015, pp. 2125–2132.
[48] T. Man, H. Shen, S. Liu, X. Jin, and X. Cheng, "Predict anchor links across social networks via an embedding approach." in *IJCAI*, vol. 16, 2016, pp. 1823–1829.

Si Zhang received the B.Eng degree in information engineering from Xi'an Jiaotong University, in 2014. He is working toward the PhD degree in the School of Computing, Informatics, and Decision System Engineering, Arizona State University. His current research interests include large scale data mining and machine learning, especially graph mining.

Hanghang Tong is currently an assistant professor at School of Computing, Informatics, and Decision Systems Engineering (CIDSE), Arizona State University since August 2014. Before that, he was an assistant professor at Computer Science Department, City College, City University of New York, a research staff member at IBM T.J. Watson Research Center and a Post-doctoral fellow in Carnegie Mellon University. He received his M.Sc and Ph.D. degree from Carnegie Mellon University in 2008 and 2009, both majored in Machine Learning. His research interest is in large scale data mining for graphs and multimedia. He has received several awards, including NSF CAREER award (2017), ICDM 2015 Highest-Impact Paper Award, four best paper awards (TUP'14, CIKM'12, SDM'08, ICDM'06), five 'bests of conference' (KDD'16, SDM'15, ICDM'15, SDM'11 and ICDM'10) and one best demo, honorable mention (SIGMOD17). He has published over 100 referred articles. He is an associated editor of SIGKDD Explorations (ACM), an action editor of Data Mining and Knowledge Discovery (Springer), and an associate editor of Neurocomping Journal (Elsevier); and has served as a program committee member in multiple data mining, databases and artificial intelligence venues (e.g., SIGKDD, SIGMOD, AAAI, WWW, CIKM, etc).