FINAL: Fast Attributed Network Alignment

Si Zhang
Arizona State University
szhan172@asu.edu

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

ABSTRACT

Multiple networks naturally appear in numerous high-impact applications. Network alignment (i.e., finding the node correspondence across different networks) is often the very first step for many data mining tasks. Most, if not all, of the existing alignment methods are solely based on the *topology* of the underlying networks. Nonetheless, many real networks often have rich attribute information on nodes and/or edges. In this paper, we propose a family of algorithms (FINAL) to align *attributed* networks. The key idea is to leverage the node/edge attribute information to guide (topology-based) alignment process. We formulate this problem from an optimization perspective based on the alignment consistency principle, and develop effective and scalable algorithms to solve it. Our experiments on real networks show that (1) by leveraging the attribute information, our algorithms can significantly improve the alignment accuracy (i.e., up to a 30% improvement over the existing methods); (2) compared with the exact solution, our proposed fast alignment algorithm leads to a more than 10× speed-up, while preserving a 95% accuracy; and (3) our on-query alignment method scales *linearly*, with an around 90% ranking accuracy compared with our exact full alignment method and a near real-time response time.

Categories and Subject Descriptors

H.2.8 [**Database Management**]: Database Applications—*Data Mining*

Keywords

Attributed network alignment; Alignment consistency; On-query alignment

1. INTRODUCTION

Multiple networks naturally appear in many high-impact application domains, ranging from computer vision, bioinformatics, web mining, chemistry to social network analysis. More often than not, network alignment (i.e., to find node correspondence across different networks) is virtually the very first step for any data mining task in these applications. For example, by linking users from different social network sites, we could recommend the products from one site (e.g., `eBay`) to the users from another site (e.g., `Facebook`) [29]. In bioinformatics, integrating different tissue-specific protein-protein interaction (PPI) networks has led to a significant improvement for candidate gene prioritization [16].

Despite the extensive research on network alignment (see Section 6 for a review), most, if not all, of those work focus on inferring the node correspondence *solely* based on the *topology*. For instance, *IsoRank* [21] propagates pairwise topology similarities in the product graph. *NetAlign* utilizes max-product belief propagation based on the network topology [2]. *BigAlign* and *UniAlign* [11] aim to infer the soft alignment based on the assumption that the adjacency matrix of one network is a noisy permutation of another network. A fundamental assumption behind these existing methods is the *topology consistency*. That is, the same node has a consistent connectivity structure across different networks (e.g., connecting to the same or similar set of the neighbors). However, such an assumption could be easily violated in some applications. For example, a user might be very active on one social network site (e.g., `Facebook`), but behaves more quietly on another site (e.g., `LinkedIn`) [11]; the same gene might exhibit dramatically different behaviors across different tissue-specific PPI network [16]. In these cases, the topology-based methods could lead to sub-optimal or even misleading alignment results.

At the same time, many real networks often have rich accompanying attributes on the nodes and/or edges (e.g., demographic information of the users, the communication types between different users), which might provide a complementary solution to address the node topology consistency assumption violation. Nonetheless, it remains a daunting task to align such attributed networks. To be specific, the following questions have largely remained open. First (*Q1. Formulation*), it is not clear how to assimilate node/edge attribute information into the topology-based network alignment formulation. For instance, many topology-based alignment approaches can often be formulated from the optimization perspective, yet it is unknown what its attributed counterpart is. Second (*Q2. Algorithms*), the optimization problem behind the topology-based network alignment is often non-convex or even NP-hard. Introducing attributes into the alignment process could only further complicate the corresponding optimization problem. How can we develop an effective solver for the attributed network alignment, with a similar or comparable time complexity to its topology-only counterpart? Third (*Q3. Scalable Computation*), how can we scale up the attributed network align-

KDD '16, August 13-17, 2016, San Francisco, CA, USA
 ISBN 978-1-4503-4232-2/16/08…$15.00
DOI: http://dx.doi.org/10.1145/2939672.2939766

ment process by taking advantage of some intrinsic properties (e.g., low-rank) of real networks? For some applications (e.g., cross-network search), 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`). How can we further speed up the computation for such an *on-query attributed network alignment* process, without solving the full alignment problem?

In this paper, we address the attributed network alignment problem, aiming to answer all these questions. The main contributions of this paper are as follows:

1. **Formulation.** We formulate the attributed network alignment from the optimization perspective. The key idea behind the proposed formulation is to leverage the node/edge attribute information to guide (topology-based) alignment process based on the *alignment consistency* principle. As a side product, our formulation helps reveal the quantitative relationships between the (attributed) network alignment problem and several other network mining problems (e.g., graph kernel, node proximity).

2. **Algorithms and Analysis.** We propose a family of algorithms FINAL to solve the attributed network alignment problem. Our analysis shows that the proposed FINAL algorithms are both effective and efficient - they converge to the global optima with a complexity that is comparable to the topology-only counterpart.

3. **Computations.** We further develop (1) an approximate algorithm FINAL-N+ to solve the full attributed network alignment problem, which reduces the time complexity from $O(mn)$ to $O(n^2)$ (where m and n are number of edges and nodes in the network, respectively); and (2) a *linear* algorithm to solve the on-query attributed network alignment, which achieves a good quality-speed trade-off.

4. **Evaluations.** We perform extensive experiments to validate the effectiveness and the efficiency of the proposed algorithms. Our experimental evaluations demonstrate that (1) our FINAL algorithms significantly improve the alignment accuracy by up to 30% over the existing methods; (2) the proposed FINAL-N+ algorithm leads to a more than 10× speed-up, while preserving a 95% accuracy compared with the exact method; 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. 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 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 $\tilde{}$ 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., $\mathbf{a} = \text{vec}(\mathbf{A})$).

Table 1: Symbols and Notation

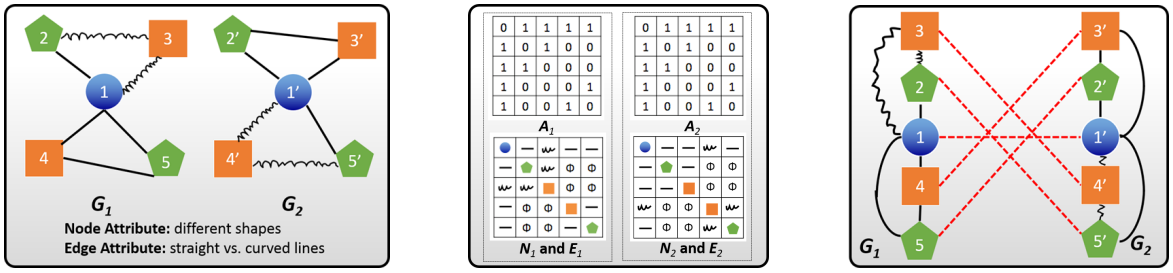
Symbols	Definition
$\mathcal{G} = \{\mathbf{A}, \mathbf{N}, \mathbf{E}\}$	an attributed network
$\mathbf{A}$	the adjacency matrix of the network
$\mathbf{N}$	the node attribute matrix of the network
$\mathbf{E}$	the edge attribute matrix of the network
n_1, n_2	# of nodes in $\mathcal{G}_1$ and $\mathcal{G}_2$
m_1, m_2	# of edges in $\mathcal{G}_1$ and $\mathcal{G}_2$
K, L	# of the node and edge labels
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, 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
$\mathbf{S}$	$n_2 \times n_1$ 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

We represent an attributed network by a triplet: $\mathcal{G} = \{\mathbf{A}, \mathbf{N}, \mathbf{E}\}$, where (1) $\mathbf{A}$ is the adjacency matrix, and (2) $\mathbf{N}$ and $\mathbf{E}$ are the node attribute matrix and the edge attribute matrix, respectively[1]. The attribute of node-a corresponds to the value of $\mathbf{N}(a,a)$, and $\mathbf{E}(a,b)$ describes the edge attribute of the edge between node-a and node-b. For a given node attribute value k, we define $\mathbf{N}^k$ as a diagonal matrix with the same size as $\mathbf{N}$, where $\mathbf{N}^k(a,a) = 1$ if node-a has the attribute value k and $\mathbf{N}^k(a,a) = 0$ otherwise. For a given edge attribute value l, we define $\mathbf{E}^l$ as a matrix of the same size with $\mathbf{E}$, where $\mathbf{E}^l(a,b) = 1$ if edge (a,b) has the attribute value l and $\mathbf{E}^l(a,b) = 0$ otherwise.

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′). 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 attribute value; (2) 3, 3′, 4 and 4′ share the same node attribute value; and (3) the two edges incident to 3 share the same edge 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′, 3 aligns to 4′ and 4 aligns to 3′). This is exactly what this paper aims to address. Formally, the attributed network alignment problem is defined as follows.

Problem 1. Attributed Network Alignment.
Given: *(1) two attributed networks* $\mathcal{G}_1 = \{\mathbf{A}_1, \mathbf{N}_1, \mathbf{E}_1\}$

[1]In this paper, we use 'graph' and 'network' interchangeably, and '(categorical) attribute' and 'label' interchangeably.



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

Figure 1: An illustrative example of the attributed network alignment problem. (a): two input attributed networks. (b): the matrix representation for attributed networks, where the upper matrices represent the adjacency matrices, and the bottom matrices represent the node attribute (the diagonal positions) and the edge attribute (the off-diagonal entries) matrices. (c): the desired alignment output (denoted by the red dashed lines).

and $\mathcal{G}_2 = \{\mathbf{A}_2, \mathbf{N}_2, \mathbf{E}_2\}$ *with* n_1 *and* n_2 *nodes respectively, (2 - optional) a prior alignment preference* $\mathbf{H}$.

Output: *the* $n_2 \times n_1$ *alignment/similarity matrix* $\mathbf{S}$*, where* $\mathbf{S}(x, a)$ *represents the alignment/similarity between node-*a *in* $\mathcal{G}_1$ *and 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 the complexity analysis in the next two sections.

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 NETWORK ALIGNMENT.

Given: *(1) two attributed networks* $\mathcal{G}_1 = \{\mathbf{A}_1, \mathbf{N}_1, \mathbf{E}_1\}$ *and* $\mathcal{G}_2 = \{\mathbf{A}_2, \mathbf{N}_2, \mathbf{E}_2\}$*, (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 for Problem 1. We start by formulating Problem 1 as a regularized optimization problem, and then propose effective algorithms to solve it, followed by some theoretic analysis.

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 the 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 all of the following conditions hold, including

- C1 *Topology Consistency.* 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.* a and x share the same node attribute value, and so do b and y;
- C3 *Edge Attribute Consistency.* Edge (a, b) and (x, y) share the same edge attribute value.

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 node attribute value 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 y respectively if they are connected by the same edge attribute value, 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 $\mathbf{S}$:

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

where (1) $a, b = 1, ..., n_1$, and $x, y = 1, ..., n_2$; (2) $\mathbb{1}(\cdot)$ is the indicator function, which takes 1 if the condition inside the parenthesis is true and zero otherwise; and (3) $f(\cdot)$ is a node-pair normalization function that is defined as

$$f(x,a) = \begin{cases} \sum_{b,y} \mathbb{1}(\mathbf{E}_1(a,b) = \mathbf{E}_2(x,y))\mathbf{A}_1(a,b) & \text{if } \mathbf{N}_1(a,a) = \mathbf{N}_2(x,x) \\ \quad \times \mathbf{A}_2(x,y)\mathbb{1}(\mathbf{N}_1(b,b) = \mathbf{N}_2(y,y)) & \\ 1 & \text{otherwise} \end{cases} \quad (2)$$

The function $f(x, a)$ measures how many (weighted) neighbor-pairs a and x have that (1) share the same node attribute

value between themselves (e.g., b and y), and (2) connect to a and x via the same edge attribute value, respectively.

Notice that the indicator function $\mathbb{1}(\cdot)$ reflects whether the two input nodes/edges share the same attribute value[2], we factorize it as follows

$$\begin{aligned}\mathbb{1}(\mathbf{N}_1(a,a)=\mathbf{N}_2(x,x)) &= \sum_{k=1}^{K}\mathbf{N}_1^k(a,a)\mathbf{N}_2^k(x,x)\\ \mathbb{1}(\mathbf{E}_1(a,b)=\mathbf{E}_2(x,y)) &= \sum_{l=1}^{L}\mathbf{E}_1^l(a,b)\mathbf{E}_2^l(x,y)\end{aligned} \tag{3}$$

Substitute Eq. (3) into Eq. (1) and Eq. (2), we have

$$\begin{aligned}J_1(\mathbf{S}) = \sum_{a,b,x,y}[\frac{\mathbf{S}(x,a)}{\sqrt{f(x,a)}}-\frac{\mathbf{S}(y,b)}{\sqrt{f(y,b)}}]^2\sum_{k,k'=1}^{K}\sum_{l=1}^{L}\mathbf{N}_1^k(a,a)\mathbf{N}_2^k(x,x)\\ \times\mathbf{A}_1(a,b)\mathbf{E}_1^l(a,b)\mathbf{A}_2(x,y)\mathbf{E}_2^l(x,y)\mathbf{N}_1^{k'}(b,b)\mathbf{N}_2^{k'}(y,y)\end{aligned} \tag{4}$$

and for nodes with the same attribute value, i.e., $\mathbf{N}_1(a,a)=\mathbf{N}_2(x,x)$

$$\begin{aligned}f(x,a) = \sum_{b,y}\sum_{k=1}^{K}\sum_{l=1}^{L}\mathbf{N}_1^k(b,b)\mathbf{N}_2^k(y,y)\mathbf{E}_1^l(a,b)\mathbf{E}_2^l(x,y)\\ \times\mathbf{A}_1(a,b)\mathbf{A}_2(x,y)\end{aligned} \tag{5}$$

Next, we present an equivalent matrix form of J_1, which is more convenient for the following algorithm description and the theoretic proof. By vectorizing the alignment matrix $\mathbf{S}$ (i.e., $\mathbf{s}=\text{vec}(\mathbf{S})$), and with the notation of element-wise product and Kronecker product, Eq. (1) can be re-written as

$$\begin{aligned}J_1(\mathbf{s}) &= \sum_{v,w}[\frac{\mathbf{s}(v)}{\sqrt{\mathbf{D}(v,v)}}-\frac{\mathbf{s}(w)}{\sqrt{\mathbf{D}(w,w)}}]^2\mathbf{W}(v,w)\\ &= \mathbf{s}^T(\mathbf{I}-\widetilde{\mathbf{W}})\mathbf{s}\end{aligned} \tag{6}$$

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

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

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 $\mathbf{D}(v,v)^{-1/2}\triangleq 0$.

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

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

where $\|\cdot\|_F$ denotes the Frobenius norm, and α is the regularization parameter. Notice that compared with J_1, we have an additional regularization term $\|\mathbf{s}-\mathbf{h}\|_F^2$ to reflect the prior alignment preference, 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 a trivial solution of J_1 with a zero alignment matrix $\mathbf{S}$.

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

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

[2]We remark that by replacing the indicator function $\mathbb{1}(\cdot)$ by an attribute value similarity function, the proposed formulation can be naturally generalized to handle the numerical attributes on nodes and/or edges.

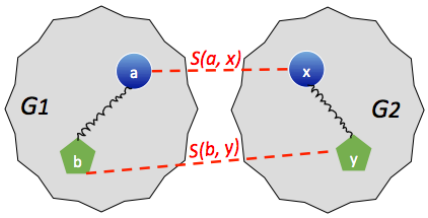


Figure 2: An illustration of alignment consistency.

which leads to the following equation

$$\begin{aligned}\mathbf{s} &= \alpha\widetilde{\mathbf{W}}\mathbf{s}+(1-\alpha)\mathbf{h}\\ &= \alpha\mathbf{D}^{-\frac{1}{2}}\mathbf{N}(\mathbf{E}\odot(\mathbf{A_1}\otimes\mathbf{A}_2))\mathbf{N}\mathbf{D}^{-\frac{1}{2}}\mathbf{s}+(1-\alpha)\mathbf{h}\end{aligned} \tag{8}$$

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 $O(m^2)$.

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 (detailed proofs are omitted due to space). The advantage of Eq. (9) is that it avoids the expensive matrix 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) $\mathcal{G}_1=\{\mathbf{A}_1,\mathbf{N}_1,\mathbf{E}_1\}$ and $\mathcal{G}_2=\{\mathbf{A}_2,\mathbf{N}_2,\mathbf{E}_2\}$, (2) optional prior alignment preference $\mathbf{H}$, (3) the regularization parameter α, and (4) the maximum iteration number $t_{\max}$.
Output: the alignment matrix $\mathbf{S}$ between $\mathcal{G}_1$ and $\mathcal{G}_2$.

```
Construct degree matrix D and node attribute matrix N;
Initiate the alignment s = h = vec(H), and t = 1;
while t ≤ t_max do
    Compute vector q = ND^{-1/2} s;
    Reshape q as Q = mat(q, n2, n1);
    Initiate an n2 × n1 zero matrix T;
    for l = 1 → L do
        Update T ← T + (E2^l ⊙ A2) Q (E1^l ⊙ A1)^T;
    end for
    Update s ← αD^{-1/2} N vec(T) + (1 − α)h;
    Set t ← t + 1;
end while
Reshape s to S = mat(s, n2, n1).
```

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 matrix $\mathbf{E}$ 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 $\mathbf{D}_N=\text{diag}(\sum_{k,k'=1}^{K}(\mathbf{N}_1^k\mathbf{A}_1\mathbf{N}_1^{k'}\mathbf{1})\otimes(\mathbf{N}_2^k\mathbf{A}_2\mathbf{N}_2^{k'}\mathbf{1}))$ 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.

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

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

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

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

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 $\mathbf{A}_2$ respectively. This variant is referred to as FINAL-P.

3.3 Proofs and Analysis

In this subsection, we first analyze the *convergency*, the *optimality* and the *complexity* 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 Theorem 1, which says the proposed FINAL-NE algorithm converges to the global optimal solution of Eq. (7).

THEOREM 1. ***Convergency 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. To prove the convergency of Eq. (8), we first show the eigenvalues of $\alpha\widetilde{\mathbf{W}}$ are in $(-1, 1)$. 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 $[-1, 1]$. Since $0 < \alpha < 1$, 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 $\lim_{t\to+\infty} \sum_{i=0}^{t-1} \alpha^i \widetilde{W}^i = (\mathbf{I} - \alpha\widetilde{\mathbf{W}})^{-1}$. 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 $\nabla^2 J = 2(\mathbf{I} - \alpha\widetilde{\mathbf{W}})$. By the Weyl's inequality theorem [4], all eigenvalues of $2(\mathbf{I} - \alpha\widetilde{\mathbf{W}})$ are 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 1. Notice that such a complexity is comparable to the complexity of topology-alone network alignment methods, such as *IsoRank* [21]. In the next section, we will propose an even faster algorithm.

LEMMA 1. ***Complexity of* FINAL-NE**. *The time complexity of Algorithm 1 is $O(Lmnt_{max} + LK^2n^2)$, 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 number of unique node and edge attributes respectively, and t_{max} is the maximum iteration number.*

PROOF. Omitted for space. □

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* [12] - a prevalent choice for node proximity. Our FINAL algorithms are also closely related to another popular node proximity method, *random walk with restart* [24]. 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 [25]. 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 [25], it mainly focuses on the edge 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* [21] 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 [2]. 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 (i.e., keeping the attribute information), 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 will 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 will 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 Theorem 1, the alignment vector $\mathbf{s}$ in FINAL-N converges to its closed-form solution as follows.

$$\begin{aligned}\mathbf{s} &= (1-\alpha)(\mathbf{I} - \alpha\widetilde{\mathbf{W}}_N)^{-1}\mathbf{h} \\ &= (1-\alpha)(\mathbf{I} - \alpha\mathbf{D}_N^{-\frac{1}{2}}\mathbf{N}(\mathbf{A}_1 \otimes \mathbf{A}_2)\mathbf{N}\mathbf{D}_N^{-\frac{1}{2}})^{-1}\mathbf{h}\end{aligned} \tag{13}$$

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 $\mathbf{W}_N$ can be defined as follows

$$\begin{aligned}\hat{\mathbf{W}}_N &= \mathbf{N}[(\mathbf{U}_1\mathbf{\Lambda}_1\mathbf{U}_1^T) \otimes (\mathbf{U}_2\mathbf{\Lambda}_2\mathbf{U}_2^T)]\mathbf{N} \\ &= \mathbf{N}(\mathbf{U}_1 \otimes \mathbf{U}_2)(\mathbf{\Lambda}_1 \otimes \mathbf{\Lambda}_2)(\mathbf{U}_1^T \otimes \mathbf{U}_2^T)\mathbf{N}\end{aligned} \tag{14}$$

Substitute Eq. (14) into Eq. (13), we can approximate the alignment vector $\mathbf{s}$ as

$$\begin{aligned}\mathbf{s} \approx& (1-\alpha)[\mathbf{I} - \alpha\mathbf{D}_N^{-\frac{1}{2}}\mathbf{N}\mathbf{U}(\mathbf{\Lambda}_1 \otimes \mathbf{\Lambda}_2)\mathbf{U}^T\mathbf{N}\mathbf{D}_N^{-\frac{1}{2}}]^{-1}\mathbf{h} \\ =& (1-\alpha)(\mathbf{I} + \alpha\mathbf{D}_N^{-\frac{1}{2}}\mathbf{N}\mathbf{U}\mathbf{\Lambda}\mathbf{U}^T\mathbf{N}\mathbf{D}_N^{-\frac{1}{2}})\mathbf{h}\end{aligned} \tag{15}$$

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 [18]: $\mathbf{\Lambda} = [(\mathbf{\Lambda}_1 \otimes \mathbf{\Lambda}_2)^{-1} - \alpha(\mathbf{U}_1^T \otimes \mathbf{U}_2^T)\mathbf{N}\mathbf{D}_N^{-1}\mathbf{N}(\mathbf{U}_1 \otimes \mathbf{U}_2)]^{-1}$.

Based on Eq. (15), our proposed FINAL-N+ algorithm is summarized in Algorithm 2. The time complexity of FINAL-N+ is summarized in Lemma 2. Notice that we often have $r \ll n$, $m \ll n^2$ and $K \ll n$. Therefore, compared with FINAL-N, FINAL-N+ is much more efficient in time complexity.

Lemma 2. ***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 number of node attributes.*

Proof. Omitted for space. □

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

Input: (1) $\mathcal{G}_1 = \{\mathbf{A}_1, \mathbf{N}_1\}$ and $\mathcal{G}_2 = \{\mathbf{A}_2, \mathbf{N}_2\}$, (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 degree matrix $\mathbf{D}_N$ and node attribute matrix $\mathbf{N}$;
2: Construct alignment preference vector $\mathbf{h} = \text{vec}(\mathbf{H})$;
3: Eigenvalue decomposition $\mathbf{U}_1\mathbf{\Lambda}_1\mathbf{U}_1^T \leftarrow \mathbf{A}_1$, $\mathbf{U}_2\mathbf{\Lambda}_2\mathbf{U}_2^T \leftarrow \mathbf{A}_2$
4: Compute $\mathbf{U} = \mathbf{U}_1 \otimes \mathbf{U}_2$;
5: Compute $\mathbf{\Lambda} = [(\mathbf{\Lambda}_1 \otimes \mathbf{\Lambda}_2)^{-1} - \alpha\mathbf{U}^T\mathbf{N}\mathbf{D}_N^{-1}\mathbf{N}\mathbf{U}]^{-1}$;
6: Compute $\mathbf{s}$ by Eq. (15);
7: Reshape vector $\mathbf{s}$ to $\mathbf{S} = \text{mat}(\mathbf{s}, n_2, n_1)$.

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 $\mathbf{S}$ (or $\mathbf{s}$) and then return the ranking vector $\mathbf{s}_a$, it would take at least $O(n^2)$ time. Next, we propose an approximate algorithm (FINAL On-Query) which directly finds the ranking vector $\mathbf{s}_a$ 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

$$\mathbf{s} = \alpha\mathbf{N}\hat{\mathbf{D}}_N^{-\frac{1}{2}}(\mathbf{A}_1 \otimes \mathbf{A}_2)\hat{\mathbf{D}}_N^{-\frac{1}{2}}\mathbf{N}\mathbf{s} + (1-\alpha)\mathbf{h} \tag{16}$$

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

$$\mathbf{s} \approx (1-\alpha)\mathbf{h} + \alpha(1-\alpha)\hat{\mathbf{D}}_N^{-\frac{1}{2}}\mathbf{N}\mathbf{U}\hat{\mathbf{\Lambda}}\mathbf{U}^T\mathbf{N}\hat{\mathbf{D}}_N^{-\frac{1}{2}}\mathbf{h} \tag{17}$$

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

$$\begin{aligned}\mathbf{s}_a =& (1-\alpha)[\mathbf{h}(v:w) + \alpha[\hat{\mathbf{D}}_N^{-\frac{1}{2}}\mathbf{N}\mathbf{U}\hat{\mathbf{\Lambda}}\mathbf{U}^T\mathbf{N}\hat{\mathbf{D}}_N^{-\frac{1}{2}}\mathbf{h}](v:w)] \\ =& (1-\alpha)[\mathbf{H}(:,a) + \alpha[(\mathbf{D}_1(a,a)\mathbf{D}_2)^{-\frac{1}{2}}(\sum_{k=1}^{K}\mathbf{N}_1^k(a,a)\mathbf{N}_2^k)] \\ &\times[\underbrace{(\mathbf{U}_1(a,:) \otimes \mathbf{U}_2)}_{O(nr^2)} \underbrace{\hat{\mathbf{\Lambda}}}_{O(n^2r^4+r^6)} \underbrace{\mathbf{U}^T\mathbf{N}\hat{\mathbf{D}}_N^{-\frac{1}{2}}\mathbf{h}}_{O(n^2r^2)}]\end{aligned} \tag{18}$$

Notice that Eq. (18) 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

$$\mathbf{g} = \sum_{i=1}^{p}\sum_{k=1}^{K} \underbrace{\sigma_i(\overbrace{\mathbf{U}_1^T\mathbf{N}_1^k\mathbf{D}_1^{-\frac{1}{2}}\mathbf{v}_i}^{O(nr)}) \otimes (\overbrace{\mathbf{U}_2^T\mathbf{N}_2^k\mathbf{D}_2^{-\frac{1}{2}}\mathbf{u}_i}^{O(nr)})}_{O(nr+r^2)=O(nr)} \tag{19}$$

We can see that the time cost for Eq. (19) is reduced to $O(pKrn)$, which is linear w.r.t the number of nodes n.

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

$$\hat{\mathbf{\Lambda}} = [\underbrace{(\mathbf{\Lambda}_2 \otimes \mathbf{\Lambda}_1)^{-1}}_{O(r^2)} - \alpha \sum_{k=1}^{K} \underbrace{(\overbrace{\mathbf{U}_1^T\mathbf{N}_1^k\mathbf{D}_1^{-1}\mathbf{U}_1}^{O(nr^2)}) \otimes (\overbrace{\mathbf{U}_2^T\mathbf{N}_2^k\mathbf{D}_2^{-1}\mathbf{U}_2}^{O(nr^2)})}_{O(nr^2+r^4)}]^{-1} \tag{20}$$

Putting everything together, the ranking vector of node-a now becomes

$$\mathbf{s}_a = (1-\alpha)\mathbf{H}(:,a) + \alpha(1-\alpha)[\overbrace{(\mathbf{D}_1(a,a)\mathbf{D}_2)^{-\frac{1}{2}}}^{O(n)} \overbrace{\sum_{k=1}^{K}\mathbf{N}_1^k(a,a)\mathbf{N}_2^k}^{O(Kn)}] \times [\underbrace{(\mathbf{U}_1(a,:)\otimes\mathbf{U}_2)}_{O(nr^2)} \underbrace{\hat{\mathbf{\Lambda}}}_{O(Knr^2+Kr^4+r^6)} \underbrace{\mathbf{g}}_{O(pKnr)}] \tag{21}$$

Based on Eq. (21), our proposed FINAL On-Query algorithm is summarized in Algorithm 3. The time complexity of FINAL On-Query is summarized in Lemma 3. 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) $\mathcal{G}_1 = \{\mathbf{A}_1, \mathbf{N}_1\}$ and $\mathcal{G}_2 = \{\mathbf{A}_2, \mathbf{N}_2\}$, (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: Compute $\mathbf{D}_a = \mathbf{D}_1(a,a)\mathbf{D}_2$, and $\mathbf{N}_a = \sum_{k=1}^{K}\mathbf{N}_1^k(a,a)\mathbf{N}_2^k$;
3: Rank r eigenvalue decomposition $\mathbf{U}_1\mathbf{\Lambda}_1\mathbf{U}_1^T \leftarrow \mathbf{A}_1$;
4: Rank r eigenvalue decomposition $\mathbf{U}_2\mathbf{\Lambda}_2\mathbf{U}_2^T \leftarrow \mathbf{A}_2$;
5: Rank p singular value decomposition $\sum_{i=1}^{p}\sigma_i\mathbf{u}_i\mathbf{v}_i^T \leftarrow \mathbf{H}$;
6: Compute $\mathbf{g}$ by Eq. (19);
7: Compute $\hat{\mathbf{\Lambda}}$ by Eq. (20);

Online-Query:

8: Compute $\mathbf{s}_a$ by Eq. (21).

Lemma 3. **Time complexity of FINAL On-Query.** *The time complexity of* FINAL On-Query *is* $O(r^6 + mr + nr^2 + m_H p + np^2 + Knr^2 + Kr^4 + pKnr)$. *where* n, m *are the orders of the number of nodes and edges respectively,* r, p *is the rank of eigenvalue decomposition and SVD, respectively,* K *is the number of node attributes and* m_H *is the number of non-zero elements in* $\mathbf{H}$.

Proof. Omitted for brevity. □

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 six real-world attributed networks.

- *DBLP Co-Authorship Network:* This dataset contains 42,252 nodes and 210,320 edges [19]. 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 [31]. 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 [30].
- *Lastfm:* This dataset was collected in 2013 and contains 136,420 users and 1,685,524 following relationships [30]. 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. [30].
- *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 [30].

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

S1. DBLP vs. DBLP. 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 [31]. 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 $\mathbf{H}$. The prior alignment matrix $\mathbf{H}$ alone leads to 7.07% one-to-one alignment accuracy.

S3. Flickr vs. Lastfm. We have the partial ground-truth alignment for these two datasets [30]. 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 [30], 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 [6]. The username similarity alone can correctly align 61.50% users.

S4. Flickr-Myspace. Same as *S3*, 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 alone can correctly align 61.80% users.

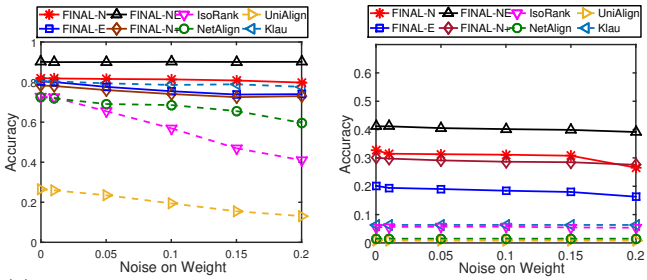
S5. ArnetMiner-ArnetMiner. We use the same method as *S1* to construct the alignment scenario as well as the prior alignment preference. This scenario contains the largest networks, and therefore is used for efficiency evaluations.

Comparison Methods. For the proposed FINAL algorithms, we test the following variants, including (1) FINAL-NE with both node and edge attributes; (2) FINAL-N with node attributes only; (3) FINAL-E with edge attributes only; (4) FINAL-N+, a low-rank based approximate algorithm of FINAL-N. We compare them with the following existing network alignment algorithms including (1) *IsoRank* [21], (2) *NetAlign* [2], (3) *UniAlign* [11] and (4) *Klau's Algorithm* [2, 9].

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. We intend to release the source code as well as all the non-proprietary datasets after the paper is published.

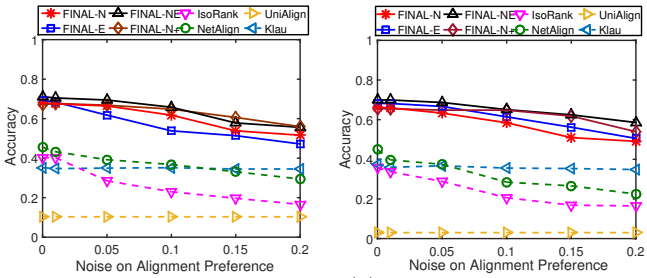
5.2 Effectiveness Analysis

We first evaluate the impact of the permutation noise on the alignment accuracy. We use a heuristic greedy matching algorithm [10] 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 three existing alignment methods. Specifically, FINAL-NE achieves a 20%-30% improvement in terms of the alignment accuracy over the existing methods (i.e., *IsoRank*, *NetAlign* and *UniAlign*). Second, FINAL-N and FINAL-E both outperform the existing methods, 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. Finally, 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



(a) DBLP co-author networks. ($\alpha = 0.8$). (b) Douban online-offline networks. ($\alpha = 0.6$).

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



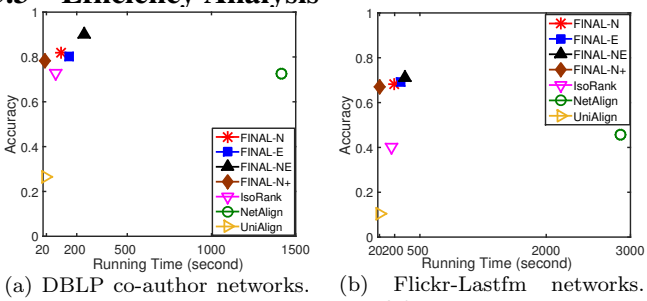
(a) Flickr-Lastfm networks. (b) Flickr-Myspace networks.

Figure 4: (Higher is better.) Alignment accuracy vs. the noise level in $\mathbf{H}$. ($\alpha = 0.3$, $t_{\max} = 30$, $r = 5$).

the existing methods is even bigger (Figure 3(b)). This is because in this case, the topology information alone (*IsoRank*, *NetAlign* and *Klau*) could actually mislead the alignment process.

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.

5.3 Efficiency Analysis



(a) DBLP co-author networks. $\alpha = 0.8$. (b) Flickr-Lastfm networks. $\alpha = 0.3$.

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

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. As we can see, the running time of our proposed exact methods (e.g., FINAL-N, FINAL-E) 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-N+ and *UniAlign* are the fastest, yet the proposed FINAL-N+ produces a much higher alignment

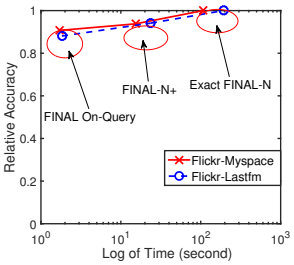


Figure 6: Alignment accuracy vs. running time for on-query alignment.

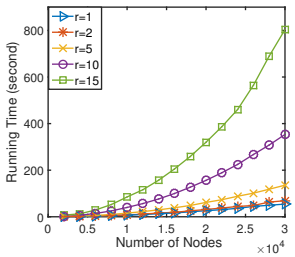
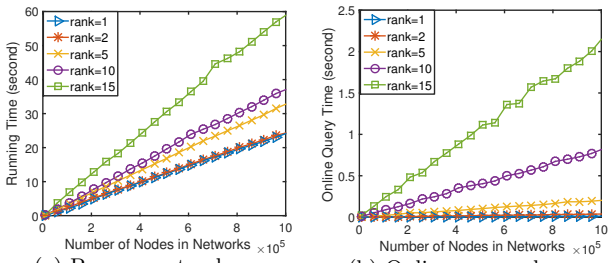


Figure 7: Scalability of FINAL-N+.

accuracy. *NetAlign* takes the longest running time as it involves a time-consuming, greedy/max-weight matching process during each iteration. We do not show the balance of *Klau's Algorithm* because the running time is usually several hours which is not comparable with other methods. Overall, FINAL-N+ achieves the best trade-off between the alignment accuracy and the running time.

Second, we evaluate the quality-speed trade-off for on-query alignment problem (i.e., Problem 2). 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 can see, that (1) FINAL-N+ preserves a 95% ranking accuracy, with a more than 10× speedup over FINAL-N, (2) FINAL On-Query preserves an about 90% ranking accuracy, and it is 100× 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 2. 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 3. In addition, 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.

Figure 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 database schema matching [15], bioinformatics [21, 13, 8], chemistry [22], computer vision [7, 20], to data mining [11, 2].

A classic alignment approach can be attributed to *IsoRank* algorithm [21], which is in turn inspired by PageRank [17]. 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. Kollias et al. [10] propose an efficient method based on uncoupling and decomposition. *SpaIsoRank* modifies *IsoRank* to maximize the number of "squares" for sparse networks [2]. In addition, *IsoRankN* [13] extends the original *IsoRank* algorithm and uses a similar approach as PageRank-Nibble [1] to align multiple networks.

Bayati et al. [3] propose a maximum weight matching algorithm for graph alignment using the max-product belief propagation [26]. Bradde et al. [5] propose another distributed message-passing algorithm based on belief propagation for protein-protein interaction network alignment. Recently, *NetAlign* [2] is proposed by formulating the network alignment problem as an integer quadratic programing problem to maximize the number of "squares". A near-optimal solution is obtained by finding the maximum a posteriori (MAP) assignment using message-passing approach. *BigAlign* formulates the bipartite network alignment problem and uses the alternating projected gradient descent to solve it [11]. Zhang et al. solve the multiple anonymized network alignment in two steps, i.e., unsupervised anchor link inference and transitive multi-network matching [28].

A related problem is to identify users from multiple social networks (i.e., the cross-site user identification problem). Zafarani et al. identify users by modeling user behavior patterns based on human limitations, exogenous and endogenous factors [27]. Tan et al. [23] propose a subspace learning method, which models user relationship by a hypergraph. Liu et al. propose a method to identify same users by behavior modeling, structure consistency modeling and learning by multi-objective optimization [14]. *COSNET* [30] considers both local the global consistency and uses an energy-based model to find connections among multiple heterogeneous networks.

7. CONCLUSION

In this paper, we study the attributed network alignment problem, including the full alignment version as well as its on-query variant. We propose an optimization-based formulation based on the alignment consistency principle. We propose a family of effective and efficient algorithms to solve the attributed network alignment problem. In detail, we first propose exact algorithms (FINAL) which are proved to converge to the global optima, 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 that (1) by assimilating the attribute information during the alignment process, the proposed FINAL algorithms significantly improve the alignment accuracy by up to 30% over the existing methods; (2) the proposed approximate alignment algorithm (FINAL-N+) achieves a good balance between the running time and the alignment accuracy; and (3) the proposed on-query alignment algorithm (FINAL On-Query) (a) preserves an around 90%+ ranking accuracy, (b) scales linearly w.r.t the

size of the input networks, and (c) responds in near real time. Future work includes generalizing FINAL algorithms to handle dynamic networks.

8. ACKNOWLEDGEMENTS

This work is partially supported by the National Science Foundation under Grant No. IIS1017415, by DTRA under the grant number HDTRA1-16-0017, by Army Research Office under the contract number W911NF-16-1-0168, by National Institutes of Health under the grant number R01LM 011986, Region II University Transportation Center under the project number 49997-33 25 and a Baidu gift. We would like to sincerely thank Dr. Jie Tang and Dr. Yutao Zhang for their generosity to share the datasets, and anonymous reviewers for their insightful and constructive comments.

9. REFERENCES

[1] R. Andersen, F. Chung, and K. Lang. Local graph partitioning using pagerank vectors. IEEE, 2006.

[2] M. Bayati, M. Gerritsen, D. F. Gleich, A. Saberi, and Y. Wang. Algorithms for large, sparse network alignment problems. IEEE, 2009.

[3] M. Bayati, D. Shah, and M. Sharma. Maximum weight matching via max-product belief propagation. IEEE, 2005.

[4] R. Bhatia. Linear algebra to quantum cohomology: the story of alfred horn's inequalities. *The American Mathematical Monthly*, 108(4):289–318, 2001.

[5] 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)*, 89(3):37009, 2010.

[6] W. Cohen, P. Ravikumar, and S. Fienberg. A comparison of string metrics for matching names and records. 2003.

[7] D. Conte, P. Foggia, C. Sansone, and M. Vento. Thirty years of graph matching in pattern recognition. *International journal of pattern recognition and artificial intelligence*, 18(03):265–298, 2004.

[8] S. Hashemifar and J. Xu. Hubalign: an accurate and efficient method for global alignment of protein–protein interaction networks. *Bioinformatics*, 30(17):i438–i444, 2014.

[9] G. W. Klau. A new graph-based method for pairwise global network alignment. *BMC bioinformatics*, 10(Suppl 1):S59, 2009.

[10] G. Kollias, S. Mohammadi, and A. Grama. Network similarity decomposition (nsd): A fast and scalable approach to network alignment. *Knowledge and Data Engineering, IEEE Transactions on*, 24(12):2232–2243, 2012.

[11] D. Koutra, H. Tong, and D. Lubensky. Big-align: Fast bipartite graph alignment. In *Data Mining (ICDM), 2013 IEEE 13th International Conference on*, pages 389–398. IEEE, 2013.

[12] 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. ACM, 2010.

[13] C.-S. Liao, K. Lu, M. Baym, R. Singh, and B. Berger. Isorankn: spectral methods for global alignment of multiple protein networks. *Bioinformatics*, 25(12):i253–i258, 2009.

[14] S. Liu, S. Wang, F. Zhu, J. Zhang, and R. Krishnan. Hydra: Large-scale social identity linkage via heterogeneous behavior modeling. ACM, 2014.

[15] S. Melnik, H. Garcia-Molina, and E. Rahm. Similarity flooding: A versatile graph matching algorithm and its application to schema matching. IEEE, 2002.

[16] 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*, pages 1356–1365. ACM, 2014.

[17] L. Page, S. Brin, R. Motwani, and T. Winograd. The pagerank citation ranking: bringing order to the web. 1999.

[18] W. W. Piegorsch and G. Casella. Erratum: inverting a sum of matrices. *SIAM review*, 32(3):470, 1990.

[19] A. Prado, M. Plantevit, C. Robardet, and J.-F. Boulicaut. Mining graph topological patterns: Finding covariations among vertex descriptors. *Knowledge and Data Engineering, IEEE Transactions on*, 25(9):2090–2104, 2013.

[20] C. Schellewald and C. Schnörr. Probabilistic subgraph matching based on convex relaxation. Springer, 2005.

[21] 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*, 105(35):12763–12768, 2008.

[22] A. Smalter, J. Huan, and G. Lushington. Gpm: A graph pattern matching kernel with diffusion for chemical compound classification. IEEE, 2008.

[23] S. Tan, Z. Guan, D. Cai, X. Qin, J. Bu, and C. Chen. Mapping users across networks by manifold alignment on hypergraph. 2014.

[24] H. Tong, C. Faloutsos, and J.-Y. Pan. Fast random walk with restart and its applications. 2006.

[25] S. V. N. Vishwanathan, N. N. Schraudolph, R. Kondor, and K. M. Borgwardt. Graph kernels. *The Journal of Machine Learning Research*, 11:1201–1242, 2010.

[26] J. S. Yedidia, W. T. Freeman, and Y. Weiss. Understanding belief propagation and its generalizations. *Exploring artificial intelligence in the new millennium*, 8:236–239, 2003.

[27] 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*, pages 41–49. ACM, 2013.

[28] J. Zhang and S. Y. Philip. Multiple anonymized social networks alignment. *Network*, 3(3):6, 2015.

[29] Y. Zhang. Browser-oriented universal cross-site recommendation and explanation based on user browsing logs. ACM, 2014.

[30] Y. Zhang, J. Tang, Z. Yang, J. Pei, and P. S. Yu. Cosnet: Connecting heterogeneous social networks with local and global consistency. ACM, 2015.

[31] E. Zhong, W. Fan, J. Wang, L. Xiao, and Y. Li. Comsoc: adaptive transfer of user behaviors over composite social network. ACM, 2012.