BRIGHT: A Bridging Algorithm for Network Alignment

Yuchen Yan
yucheny5@illinois.edu
University of Illinois at Urbana-Champaign
IL, USA

Si Zhang
sizhang2@illinois.edu
University of Illinois at Urbana-Champaign
IL, USA

Hanghang Tong
htong@illinois.edu
University of Illinois at Urbana-Champaign
IL, USA

ABSTRACT

Multiple networks emerge in a wealth of high-impact applications. Network alignment, which aims to find the node correspondence across different networks, plays a fundamental role for many data mining tasks. Most of the existing methods can be divided into two categories: (1) consistency optimization based methods, which often explicitly assume the alignment to be *consistent* in terms of neighborhood topology and attribute across networks, and (2) network embedding based methods which learn low-dimensional node embedding vectors to infer alignment. In this paper, by analyzing representative methods of these two categories, we show that (1) the consistency optimization based methods are essentially specific random walk propagations from anchor links that might be too restrictive; (2) the embedding based methods no longer explicitly assume alignment consistency but inevitably suffer from the space disparity issue. To overcome these two limitations, we *bridge* these methods and propose a novel family of network alignment algorithms BRIGHT to handle both plain and attributed networks. Specifically, it constructs a space by random walk with restart (RWR) whose bases are one-hot encoding vectors of anchor nodes, followed by a shared linear layer. Our experiments on real-world networks show that the proposed family of algorithms BRIGHT outperform the state-of-the-arts for both plain and attributed network alignment tasks.

KEYWORDS

Network alignment; random walk with restart; network embedding

ACM Reference Format:
Yuchen Yan, Si Zhang, and Hanghang Tong. 2021. BRIGHT: A Bridging Algorithm for Network Alignment . In *Proceedings of the Web Conference 2021 (WWW '21), April 19–23, 2021, Ljubljana, Slovenia.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3442381.3450053

1 INTRODUCTION

Multiple networks play an important role in many applications, ranging from academic scholar analysis, bioinformatics, e-commerce product recommendation to financial fraud detection. Network alignment aims to find node correspondence across networks, which is a fundamental step for many downstream tasks. For example, by aligning users across different social networks, we can integrate multi-sourced information to model user preferences such that more personalized recommendations can be achieved [2]. In bioinformatics, human aging which is hard to study due to long human lifespan can be elucidated by identifying the functional orthologs and transferring the biological insights across model species and human via network alignment [5].

Many existing network alignment methods are often based on the alignment consistency principle and formulated as an optimization problem. One fundamental assumption underlying these methods is that the neighborhood topology and/or attributes of the nodes to be aligned are consistent across networks. Conventionally, network alignment can be formulated as a graph matching problem, which considers one network as a noisy permutation of the other [14]. The objective of this type of methods is to minimize $\|\mathbf{A_1} - \mathbf{P}\mathbf{A_2}\mathbf{P}^T\|_F^2$ where $\mathbf{A_1}$ and $\mathbf{A_2}$ are adjacency matrices of two networks and $\mathbf{P}$ is the node permutation matrix. In addition, the consistency assumption behind IsoRank [23] is that if two nodes are aligned together, their corresponding close neighbors are also likely to be aligned. FINAL [34] further integrates node and edge attribute consistency assumptions to cope with attributed network alignment. However, this consistency assumption could be violated due to network heterogeneity. For example, users may have preferences of certain social platforms (e.g. Facebook) than others (e.g. Twitter). In this way, they might behave actively in one social network while being quiet in the other. Moreover, as shown in Figure 1, different from career-oriented platform LinkedIn, edges in Google scholar network may describe the co-authorship, which introduces another type of heterogeneity (i.e., different semantics of relations) between these two networks.

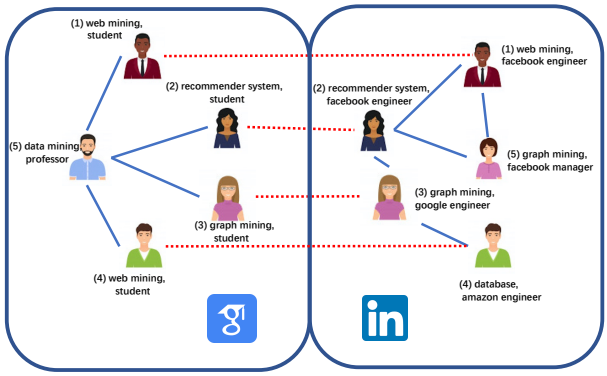


Figure 1: An illustrative example where the consistency assumption could be violated.

Network embedding based methods have been proposed recently that aim to learn low-dimensional node embedding vectors in order

WWW '21, April 19–23, 2021, Ljubljana, Slovenia

ACM ISBN 978-1-4503-8312-7/21/04.
https://doi.org/10.1145/3442381.3450053

to infer node alignment. Given a set of anchor node pairs [1], the goals of embedding based methods can be summarized as (1) to preserve the topology information of each network by node embeddings, and (2) in the meanwhile to make embeddings of anchor node pairs as close as possible. For plain networks, IONE [16] uses follower/followee edges as the structural information and embeds nodes similarly to LINE [25] to preserve both inner-network and cross-network node proximities. Chu et al. propose an embedding method that assumes the embedding space of two networks can be transformed linearly, which yet might be ineffective to capture the complex alignment relationships [3]. For attributed networks, Origin [36] utilizes graph convolutional networks [11] to capture both the topology and attribute information for network alignment. However, although embedding based methods do not explicitly assume alignment consistency in terms of topology/attribute, they inevitably introduce the embedding *space disparity issue* into the alignment task. As shown in Figure 2, even fixing anchor node pairs with same embeddings (denoted by green diamond nodes), node embeddings of two isomorphic networks can lie symmetrically in the embedding space, which could further mislead the alignment. In this way, it is crucial yet challenging to train node embeddings of two networks in one unified space.

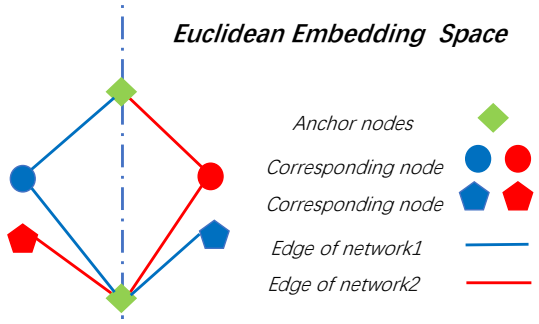


Figure 2: An illustrative example of the space disparity issue. Green nodes are anchor nodes, blue nodes represent the non-anchor nodes in network1 and red nodes represent the non-anchor nodes in network2. The blue round node of network1 should be aligned to the red round node of network2. However, because of the space disparity, the blue round node is closer to the red pentagonal node and aligned to it.

In this paper, we first analyze some representative methods of two categories, including (1) consistency optimization based methods and (2) embedding based methods. Specifically, we show that consistency optimization based methods that build upon topology/attribute consistency assumption (e.g., FINAL [34] and IsoRank [23]) can be considered as conducting random walks from anchor nodes *simultaneously* in different networks. However, random walk propagation with the exactly same number of steps in two networks might be too restrictive to capture structural inconsistency/differences between them (*restrictive propagation limitation*). On the other hand, by breaking down the typical objective functions of embedding based methods, we reveal that their loss functions are essentially the *upper bound* of the objective function of consistency optimization based methods. As we will show in Section 3.2, the error introduced by this upper bound can be accumulated, which eventually leads to the space disparity issue.

To address the above limitations, we propose a novel family of network alignment algorithms BRIGHT, including BRIGHT-U for plain networks and BRIGHT-A for attributed networks. The key idea is to use the training anchor links as the *bases/landmarks* to construct a certain unified space by random walk with restart (RWR) [27]. The RWR scores with respect to an anchor node are used as the values of initial node embeddings at the corresponding dimension. We then use a shared linear layer to learn the importance of the RWR scores at different dimensions. In this way, the proposed methods bear the flexibility of handling the restrictive propagation limitation and the ability of addressing the space disparity issue. For attributed networks, we additionally utilize a graph convolution network (GCN) module to leverage the attribute information. The main contributions of this paper can be summarized as follows.

- **Theoretic Analysis.** To our best knowledge, we are the first to reveal some fundamental limitations of the consistency optimization based methods and embedding based methods and aim to bridge them.
- **Novel Models.** We propose a family of novel network alignment algorithms BRIGHT, including BRIGHT-U for plain networks and BRIGHT-A for attributed networks.
- **Experimental Results.** We perform extensive experiments on five datasets in different scenarios, which demonstrate the effectiveness of the proposed model.

The rest of the paper is organized as follows. Section 2 defines network alignment problem. Section 3 gives the analysis of consistency optimization based methods and embedding based methods. Section 4 presents the proposed BRIGHT model. Section 5 shows the experimental results. Related works and conclusion are given in Section 6 and Section 7 respectively.

2 PROBLEM DEFINITION

Table 1 summarizes main notations used in this paper. Bold uppercase letters are used for matrices (e.g., $\mathbf{A}$). Bold lowercase letters are for vectors (e.g., $\mathbf{s}$) and lowercase letters (e.g., p) for scalars. For matrix, we use $\mathbf{A}(i,j)$ to denote the entry at the intersection of the i-th row and j-th column of matrix $\mathbf{A}$, The transpose of $\mathbf{A}$ is denoted by the superscript T (i.e., $\mathbf{A}^T$). $\hat{\mathbf{A}}$ is the normalized matrix of $\mathbf{A}$, e.g., the symmetrically normalized adjacency matrix $\hat{\mathbf{A}} = \mathbf{D}^{-\frac{1}{2}}\mathbf{A}\mathbf{D}^{-\frac{1}{2}}$ where $\mathbf{D}$ is the degree matrix of $\mathbf{A}$. The corresponding lowercase is used to denote the vectorization of a matrix (e.g., $\mathbf{s} = \text{vec}(\mathbf{S})$).

An attributed network is represented as $\mathcal{G} = \{\mathbf{A}, \mathbf{X}\}$, where (1) $\mathbf{A}$ is the adjacency matrix, and (2) $\mathbf{X}$ is the node attribute matrix. The network alignment task can be defined as follows:

Problem 1. *Semi-supervised attributed network alignment.*
Given: *(1) two attributed networks $\mathcal{G}_1 = \{\mathbf{A}_1, \mathbf{X}_1\}$ and $\mathcal{G}_2 = \{\mathbf{A}_2, \mathbf{X}_2\}$; (2) a set of anchor node pairs $\mathcal{L}$.*

[1] In this paper, we use anchor node pairs and anchor links interchangeably.

Table 1: Symbols and Notations.

Symbol	Definition
$\mathcal{G}_1, \mathcal{G}_2$	input networks
$\mathbf{A_1}, \mathbf{A_2}$	adjacency matrices of networks
$\mathbf{X_1}, \mathbf{X_2}$	node attribute matrices of networks
$\mathbf{R_1}, \mathbf{R_2}$	RWR score matrices
$\mathbf{E_1}, \mathbf{E_2}$	embedding matrices of networks
n_1, n_2	# of nodes in $\mathcal{G}_1, \mathcal{G}_2$
m_1, m_2	# of edges in $\mathcal{G}_1, \mathcal{G}_2$
a, b	node indices of $\mathcal{G}_1$
x, y	node indices of $\mathcal{G}_2$
l	an anchor link between $\mathcal{G}_1$ and $\mathcal{G}_2$
$\mathcal{L}$	the set of anchor node links
$\mathbf{I}$	an identity matrix
$\mathbf{1}$	a vector of 1s
$\mathbf{H}$	$n_2 \times n_1$ prior alignment preference
$\mathbf{S}$	$n_2 \times n_1$ alignment/similarity matrix
$\mathbf{r}$	the random walk with restart vector
α	the parameter, $0 < \alpha < 1$
β	the restart probability in RWR
$d(\cdot,\cdot)$	the distance for node embeddings
$\mathbf{s} = \text{vec}(\mathbf{S})$	vectorization of matrix $\mathbf{S}$ in the column order
$\hat{\mathbf{A}}$	normalized matrix of $\mathbf{A}$
$\mathbf{D} = \text{diag}(\mathbf{d})$	diagonal matrix of the degree vector $\mathbf{d}$
$\otimes$	Kronecker product
$[\cdot\|\cdot]$	matrix or vector concatenation

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

Remarks. If we do not have $\mathbf{X}_1$ and $\mathbf{X}_2$, this will become the *semi-supervised plain network alignment* problem. If we do not have $\mathcal{L}$, this will become the *unsupervised attributed network alignment* problem.

3 ANALYSIS

In this section, we analyze both consistency optimization based methods and embedding based methods. Through the analysis, we show that the consistency optimization based methods conduct restrictive random walk propagation of anchor links and embedding based methods are essentially the relaxed versions of the consistency optimization based methods.

3.1 Consistency Optimization Based Methods

For consistency optimization based methods, the anchor node links set $\mathcal{L}$ is encoded in the form of the prior alignment matrix $\mathbf{H}$, where the entry corresponding to an anchor link is 1 and 0 otherwise. We start with the semi-supervised plain network alignment task and attributed network alignment task will be discussed later in this subsection. For plain network alignment task, topology consistency is a basic assumption, which assumes that if a and b are neighbors in $\mathcal{G}_1$, x, y are neighbors in $\mathcal{G}_2$ and if a is aligned to x, it is very likely that b should be aligned to y. In addition, the final similarity matrix $\mathbf{S}$ should be consistent with prior alignment matrix $\mathbf{H}$. The objective function of consistency optimization based methods [23, 34] can be formulated as:

$$O_c(\mathbf{S}) = \alpha \sum_{a,b,x,y} [\frac{\mathbf{S}(x,a)}{\sqrt{\mathbf{d}_2(x)\mathbf{d}_1(a)}} - \frac{\mathbf{S}(y,b)}{\sqrt{\mathbf{d}_2(y)\mathbf{d}_1(b)}}]^2 \mathbf{A}_1(a,b)\mathbf{A}_2(x,y) + (1-\alpha)\|\mathbf{S}-\mathbf{H}\|_F^2 \tag{1}$$

where $\mathbf{d}_2(x), \mathbf{d}_1(a), \mathbf{d}_2(y), \mathbf{d}_1(b)$ are the degrees of nodes x, a, y, b and α is the parameter. This objective function has a closed form solution given by [34]:

$$\mathbf{s} = (1-\alpha)(\mathbf{I} - \alpha\hat{\mathbf{W}})^{-1}\mathbf{h} \tag{2}$$

where $\mathbf{s}$ and $\mathbf{h}$ are vectorizations of $\mathbf{S}$ and $\mathbf{H}$ respectively, $\mathbf{W} = \mathbf{A}_1 \otimes \mathbf{A}_2$ is the Kronecker product of $\mathbf{A}_1$ and $\mathbf{A}_2$ and $\hat{\mathbf{W}}$ is the symmetrically normalized matrix of $\mathbf{W}$. It is equal to the following summation:

$$\mathbf{s} = (1-\alpha)\sum_{t=0}^{\infty} \alpha^t \hat{\mathbf{W}}^t \mathbf{h} \tag{3}$$

Then, let us take any node pair (b, y) into consideration, where b is a node in $\mathcal{G}_1$ and y is a node in $\mathcal{G}_2$. Assuming that (b, y) corresponds to the i-th entry in $\mathbf{s}$, we can break the prior alignment vector $\mathbf{h}$ for each anchor link:

$$\mathbf{s}(i) = (1-\alpha)\sum_{j=0}^{n_2\times n_1 - 1}\sum_{t=0}^{\infty} \alpha^t \hat{\mathbf{W}}^t(i,j)\mathbb{1}(\mathbf{h}(j)) \tag{4}$$

where $\mathbb{1}(\cdot)$ takes 1 if the condition inside the parenthesis is true and zero otherwise. If $\mathbf{h}(j) = 0$, the value of the corresponding item in summation is 0, which will be ignored. If $\mathbf{h}(j) = 1$, let us take the item $\alpha^t \hat{\mathbf{W}}^t(i,j)$ for analysis. Since $\hat{\mathbf{W}} = \mathbf{D}_{\mathbf{W}}^{-\frac{1}{2}}\mathbf{W}\mathbf{D}_{\mathbf{W}}^{-\frac{1}{2}}$ and $\mathbf{W} = \mathbf{A}_1 \otimes \mathbf{A}_2$, $\hat{\mathbf{W}}^t(i,j)\mathbb{1}(\mathbf{h}(j))$ means that the anchor link represented by $\mathbf{h}(j) = 1$ moves exactly t steps and it will arrive the node pair (b, y), which is represented by $\mathbf{s}(i)$. So, the node pair (b, y) will get a score $\mathbf{s}(i)$ from the anchor link $\mathbf{h}(j) = 1$ with a decaying rate α. At last, scores from all anchor links will be added together.

From the above analysis we know that consistency optimization based methods are random walk propagation of anchor links, which is shown in Figure 3. Methods for attributed network alignment task relying on node/edge attribute consistency to prune the infeasible choices when choosing the direction of next step, which is a generalized version of plain methods. From Eq. (4), we can see that:

- The equation requires that only when two anchor nodes of an anchor link (e.g., $\mathbf{h}(i)$) arrive at the given node pair (b, y) with the exactly same step t ($\hat{\mathbf{W}}^t(i,j)\mathbb{1}(\mathbf{h}(j))$), can the given node pair (b, y) receive a score. Even a little perturbation (e.g., deleting a node/edge) could make the node pair (b, y) receive a zero score from this anchor link.
- The weights of scores from all anchor links are set equally as 1*s* and simply added together. If some anchor link has a negative effect on the alignment task, these methods are not able to weed out such an anchor link.

3.2 Embedding Based Methods

For embedding based methods, we show that they can be viewed as the relaxation of consistency optimization methods with objective

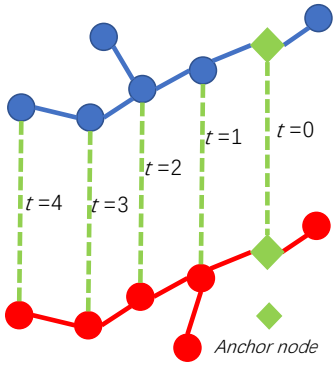


Figure 3: An example of random walk propagation of anchor link with 4 steps. The anchor link between two networks moves step by step ($t = 0$ to $t = 4$). Once it arrives at a pair of nodes, this pair of nodes will obtain a score for alignment.

function in Eq. (1). Specifically, we show that consistency optimization based methods *directly* optimize node similarities across networks (i.e., $\mathbf{S}(y, b)$). In contrast, embedding based methods relax it into *indirectly* optimizing the node similarities through a path of node pairs $\{(y, x), (x, a), (a, b)\}$. To analyze the space disparity issue, we start from analyzing the loss functions of embedding based methods. Loss functions of these methods [3, 16, 36] can be generally written as follows:

$$O = O_1^{in} + O_2^{in} + O^{cross} \tag{5}$$

where O_1^{in} and O_2^{in} are the within network loss functions of $\mathcal{G}_1$ and $\mathcal{G}_2$ respectively, and O^{cross} is the cross network loss. Considering one negative sample for each positive sample, O_1^{in} can be written as

$$O_1^{in} = \sum_a (d(a, b) - d(a, c)) \tag{6}$$

where $d(\cdot, \cdot)$ is the distance or dissimilarity function of two nodes in the embedding space, which could be Euclidean distance or negative cosine similarity as a dissimilarity measure. Here, (a, b) denotes a positive node pair which has an edge between them [25] or is sampled in a fixed length random walk [21] and (a, c) is a negative sample of node pairs. O_2^{in} can be computed similarly as O_1^{in} but in terms of $\mathcal{G}_2$. In addition, the O^{cross} can be written as:

$$O^{cross} = \sum_{(l_1, l_2) \in \mathcal{L}} d(l_1, l_2) \tag{7}$$

where l_1 and l_2 are corresponding training anchor node pairs in $\mathcal{G}_1$ and $\mathcal{G}_2$.

To better illustrate our analysis, we simplify the alignment setting into the following scenario that (1) $\mathcal{G}_1$ and $\mathcal{G}_2$ are isomorphic k-regular graphs; (2) $d(\cdot, \cdot)$ is a distance metric; (3) the alignment matrix $\mathbf{S}$ is computed by $e^{-d(\cdot,\cdot)}$ (e.g., $\mathbf{S}(x, a) = e^{-d(x,a)}$); and (4) the positive node pairs (a, b) and (x, y) have edges between them.

Since consistency optimization based methods are random walk propagation of anchor links, we also conduct our analysis from anchor links. Let (a, x) is be an anchor link between $\mathcal{G}_1$ and $\mathcal{G}_2$, (a, b) be an edge in $\mathcal{G}_1$ and (x, y) be an edge in $\mathcal{G}_2$. Then, the corresponding item of (a, x) in Eq. (1) to minimize the objective function is as following:

$$O_c(\mathbf{S}) = \frac{1}{K^2}(1 - e^{-d(y,b)})^2 \mathbf{A}_1(a, b)\mathbf{A}_2(x, y) \tag{8}$$

where K is the degree in K-regular graph. In the ideal case, obviously, $d(y, b) = 0$ is a trivial solution. Taking the loss function of embedding based methods into consideration, O_1 minimizes $d(a, b)$ and O_2 minimizes $d(x, y)$. Here, we omit the negative sampling item $d(a, c)$ for now, and it will be included later. O^{cross} actually minimizes $d(a, x)$. So, the item for (a, b, x, y) in the loss function O will become:

$$O = d(b, a) + d(a, x) + d(x, y) \tag{9}$$

which is optimizing an upper bound of $d(b, y)$ because:

$$d(b, y) \leq d(b, a) + d(a, x) + d(x, y) \tag{10}$$

As we can see, embedding based methods optimize over $d(b, a)$, $d(a, x)$, $d(x, y)$ for node pair (b, y). However, Eq. (8) directly optimizes over $d(b, y)$. Suppose (a, b) and (x, y) are not totally equivalent nodes in $\mathcal{G}_1$ and $\mathcal{G}_2$, i.e., $d(b, a) > 0$ and $d(x, y) > 0$. In the ideal case, when embedding based methods converge, $d(x, a) = 0$ and there exists a very small positive number ϵ satisfying the condition that $\epsilon < \min\{d(b, a), d(x, y)\}$. In this case, there exists an arbitrary $\delta \in [0, 2\epsilon]$ such that $d(b, y)$ can converge to δ. This implies that even if the loss function converges ideally, $d(b, y)$ is not necessarily 0, which means that even if the loss function converges ideally, $d(b, y)$ is not necessarily 0. Now, let us take the negative sampling item $d(a, c)$ into consideration. This item aims to make embeddings of other nodes not linked to a away from the node a's embedding, which leads to the situation that even if $d(y, b) \neq 0$, it is small enough to distinguish the right node pair (e.g., (b, y) in Figure 4) from the wrong pair (e.g., (y, c_1) in Figure 4). However, the small δ error will accumulate along with the length of random walks from anchor links (e.g., a incorrectly aligned pair (z_2, c_1) in Figure 4), especially when anchor links are sparse in the network, which leads to a severe space disparity issue. If (a, b) are sampled in a fixed length of random walk instead of being directly linked, the error becomes larger. After showing that embedding based methods are essentially *relaxed* versions of consistency optimization based methods, we present a proposition about the space disparity issue:

Proposition 1. *An embedding based network alignment method able to handle the space disparity issue must satisfy the following necessary but not sufficient condition. Given two isomorphic graphs $\mathcal{G}_1$ and $\mathcal{G}_2$ and some anchor links $\mathcal{L}$ between them, for any node pair (a, x) across $\mathcal{G}_1$ and $\mathcal{G}_2$ that should be aligned, a and x have the same embedding with $d(a, x) = 0$ in the output of the model.*

The correctness of Proposition 1 is straightforward. The goal of network alignment is to make the distance between node pairs that should be aligned equal to 0. In other words, starting from anchor links, any small δ of $d(a, x)$ in current embedding based methods is not allowed during the training process. However, this is not a sufficient condition for the correctness of network alignment as trivial solutions with meaningless node embeddings exist (e.g., $\mathbf{0}$ for all nodes). In this way, the network alignment model needs to be capable of learning informative node embeddings.

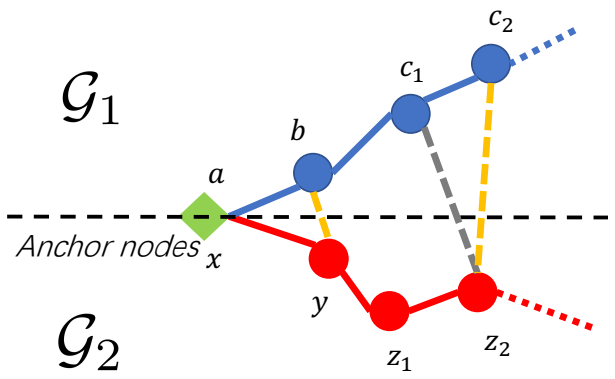


Figure 4: An illustrative example of error accumulation along the path from anchor link (green diamond node). Yellow dash lines represent the correct alignment and grey dash line represents the wrong alignment. Node y in $\mathcal{G}_2$ can still be aligned to node b in $\mathcal{G}_1$ (yellow dash line). Node z_2 in $\mathcal{G}_2$ should be ideally aligned to node c_2 in $\mathcal{G}_1$, but it is closer to node c_1 in $\mathcal{G}_1$ due to the accumulated error (grey dash line).

4 METHOD

In this section, we present the proposed model BRIGHT, including BRIGHT-U for plain networks and BRIGHT-A for attributed networks. We start with the overview of our model, and then detail the two proposed algorithms (BRIGHT-U and BRIGHT-A), followed by the training process.

4.1 Overview

To address both the restrictive propagation limitation and the space disparity issue simultaneously, the proposed BRIGHT strives to satisfy the following requirements:

- *Requirement 1*: It should lift the restriction of having exactly same steps of random walks in both networks, and adjust the way to combine anchor links' weights.
- *Requirement 2*: It should build a unified embedding space to meet the condition in Proposition 1.

The framework of BRIGHT is shown in Figure 5. The key idea of BRIGHT is to treat the anchor links as the *landmarks* in the alignment task. Correspondingly, the one-hot vectors of these anchor links form the *bases* of some space that captures the proximity by RWR. Each dimension of initial embeddings is computed by the RWR scores w.r.t an anchor link instead of the local neighbor structure [3, 16]. The initial embeddings represent the relative positions of all nodes in the entire network. Followed by the shared modules, the proposed methods meet the condition in Propostion 1 as we will show in subsection 4.2. For *Requirement 1*, the *restart* process in RWR relieves the exact t step limitation by conducting separate RWRs in two networks and the shared linear layer in BRIGHT-U can adjust weights of scores from different anchor links for combination.

4.2 BRIGHT-U

For plain network alignment, only $\mathbf{A}_1$, $\mathbf{A}_2$ and the anchor links set $\mathcal{L}$ are given. First, we aim to avoid the restrictive propagation limitation. At the same time, we want to build a unified embedding space for both two networks to be aligned to address the space disparity issue. The key intuition is that, the set of anchor links $\mathcal{L}$ provides the *landmarks* for all nodes in both networks. Relative positions based on anchor links can form a unified space for all nodes, regardless which network they belong to. According to the analysis in section 3, the previous consistency optimization based methods are random walk from anchor links with the exactly same steps. Therefore, we use random walk with restart to measure the relative position between nodes and anchor links. The separate RWRs in two networks help avoid the same step restriction and bring more flexibility to the random walk process, which is more robust than other distance measurements like shortest path distance. Given an anchor link $l \in \mathcal{L}$ (we use l_1 and l_2 to denote the same l in $\mathcal{G}_1$ and $\mathcal{G}_2$), the RWR score vector $\mathbf{r}_{l_1}$ of size $n_1 \times 1$ can be obtained as:

$$\mathbf{r}_{l_1} = (1-\beta)\hat{\mathbf{W}}_1\mathbf{r}_{l_1} + \beta\mathbf{e}_{l_1} \tag{11}$$

where $\hat{\mathbf{W}}_1 = (\mathbf{D}^{-1}\mathbf{A}_1)^T$ is the row normalized matrix of $\mathbf{A}_1$, β is the restart probability and $\mathbf{e}_{l_1}$ is one-hot vector with $\mathbf{e}_{l_1}(l_1) = 1$ and all other entries are 0. The final $\mathbf{r}_{l_1}$ can be solved as:

$$\mathbf{r}_{l_1} = \beta(\mathbf{I} - (1-\beta)\hat{\mathbf{W}}_1)^{-1}\mathbf{e}_{l_1} \tag{12}$$

Here we give an intuitive explanation on why this process satisfies the condition in Proposition 1. Given two isomorphic graphs $\mathcal{G}_1$ and $\mathcal{G}_2$ and (a, x) as a ground-truth aligned node pair across $\mathcal{G}_1$ and $\mathcal{G}_2$, the relative position of node a to anchor link l_1 is $\mathbf{r}_{l_1}(a)$ and the relative position of node x to anchor link l_2 is $\mathbf{r}_{l_2}(x)$. In Eq. (12), the item $(\mathbf{I}-(1-\beta)\hat{\mathbf{W}}_1)^{-1}$ is similar to $(\mathbf{I}-\alpha\hat{\mathbf{W}})^{-1}$ in Eq. (2). It represents all normalized paths from the anchor link to a node. Since (a, x) is a ground-truth aligned node pair in two isomorphic graphs and (l_1, l_2) is anchor link (also a ground-truth aligned node pair). So, all paths from l_1 to a are same as those from l_2 to x. As a result, $\mathbf{r}_{l_1}(a) = \mathbf{r}_{l_2}(x)$, which satisfies the condition in Proposition 1. After getting all RWR vectors for $\mathcal{G}_1$ and $\mathcal{G}_2$, we build two RWR score matrices $\mathbf{R}_1$ of size $n_1 \times |\mathcal{L}|$ and $\mathbf{R}_2$ of size $n_2 \times |\mathcal{L}|$. Then, we normalize $\mathbf{R}_1$ and $\mathbf{R}_2$ w.r.t. each row as $\hat{\mathbf{W}}_{\mathbf{R}_1} \in \mathbb{R}^{n_1 \times |\mathcal{L}|}$ and $\hat{\mathbf{W}}_{\mathbf{R}_2} \in \mathbb{R}^{n_2 \times |\mathcal{L}|}$, whose value in each position represents the relative position of a node to an anchor link.

To address the limitation that scores from different anchor links are simply added together with equal weights in the consistency optimization based methods, BRIGHT-U uses a *shared* linear layer to adjust the weights of scores from different anchor links and to keep node embeddings of $\mathcal{G}_1$ and $\mathcal{G}_2$ in the same embedding space during the training:

$$\mathbf{E}_1 = \mathbf{E}_1^{RWR} = LINEAR(\hat{\mathbf{W}}_{\mathbf{R}_1}) \tag{13}$$

where $\mathbf{E}_1^{RWR} \in \mathbb{R}^{n_1 \times k}$ denotes the final embedding matrix of $\mathcal{G}_1$ and k is a fixed dimension. By utilizing RWR and a shared linear layer, BRIGHT-U successfully leverages node embeddings of two networks in the unified space and relaxes the restrictive propagation limitation.

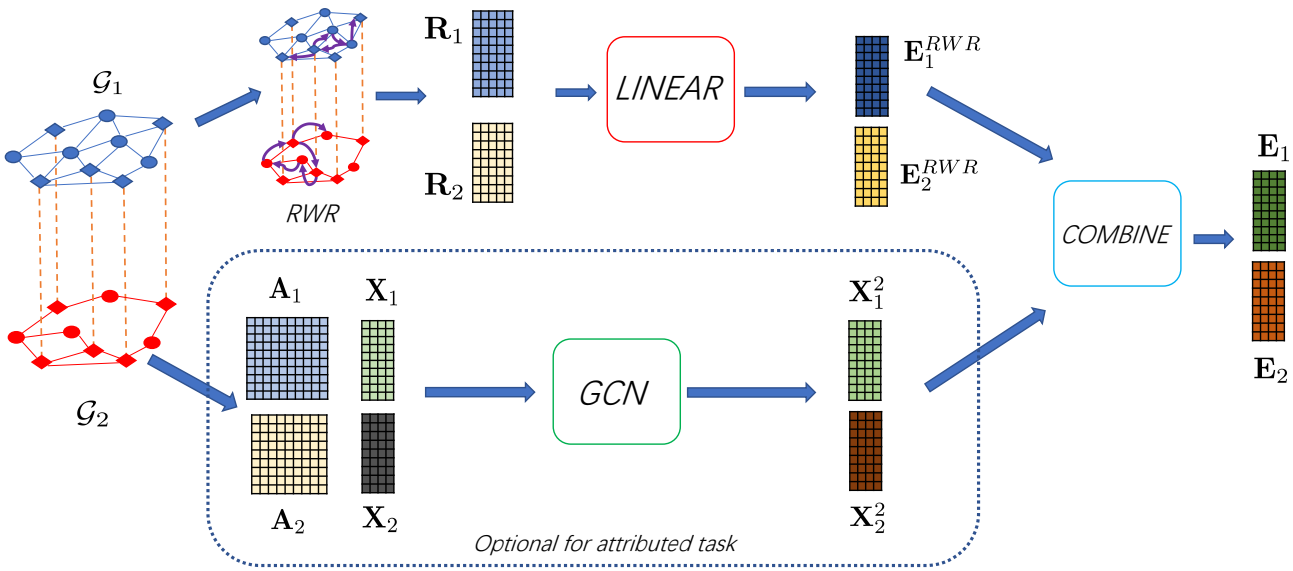


Figure 5: Framework of BRIGHT . $\mathcal{G}_1$ has 10 nodes, $\mathcal{G}_2$ has 9 nodes and 5 anchor links exist between $\mathcal{G}_1$ and $\mathcal{G}_2$. The dimension of node attributes and node embeddings are both 4.

4.3 BRIGHT-A

For attributed network alignment task, as shown in Figure 5, we add a shared two-layer graph convolution network (GCN) module to capture the attribute information. GCN embedding matrices $\mathbf{X}_1^2$, $\mathbf{X}_2^2$ will be concatenated with $\mathbf{E}_1^{RWR}$, $\mathbf{E}_2^{RWR}$ and pass through a shared linear combination layer to obtain final embedding matrices $\mathbf{E}_1$ and $\mathbf{E}_2$. The first layer and second layer of GCN for $\mathcal{G}_1$:

$$\mathbf{X}_1^1 = \sigma(\hat{\mathbf{A}}_1\mathbf{X}_1\mathbf{W}^1) \tag{14}$$

$$\mathbf{X}_1^2 = \sigma(\hat{\mathbf{A}}_1\mathbf{X}_1^1\mathbf{W}^2) \tag{15}$$

where $\hat{\mathbf{A}}_1 = \mathbf{D}_1^{-\frac{1}{2}}\mathbf{A}_1\mathbf{D}_1^{-\frac{1}{2}}$, σ is the activation function, $\mathbf{W}^1$ and $\mathbf{W}^2$ are parameter matrices of the first and second layers of GCN. Then, $\mathbf{X}_1^2$ and the RWR embedding matrix from $\mathbf{E}_1^{RWR}$ will be concatenated and fed into a *combination* layer:

$$\mathbf{E}_1 = COMBINE([\mathbf{E}_1^{RWR} \| \mathbf{X}_1^2]) \tag{16}$$

The final embedding matrices $\mathbf{E}_1$ and $\mathbf{E}_2$ are still in a unified embedding space because all modules involved in the training process are shared for $\mathcal{G}_1$ and $\mathcal{G}_2$ with same parameters.

4.4 BRIGHT Model Training

When optimizing the model, We use the marginal ranking loss, which was originated in the entity linking community [30]. With $\mathbf{E}_1$ and $\mathbf{E}_2$, we have

$$\mathcal{J}_i = \frac{1}{|\mathcal{L}|}\sum_{l\in\mathcal{L}}\frac{1}{|U_{l_i}|}\sum_{u\in U_{l_i}}\max\{0, \gamma + d(l_1, l_2) - d(l_i, u)\} \tag{17}$$

where $\mathcal{L}$ is the anchor link set, γ is the margin parameter, $i \in \{1, 2\}$ is to discriminate two networks, l_1 and l_2 are corresponding anchor nodes for anchor link l in $\mathcal{G}_1$ and $\mathcal{G}_2$, U_{l_i} is the negative pair sampling set for node l_i , $u \in U_{l_i}$ is the negative sample pair in $\mathcal{G}_{3-i}$ for l_i in $\mathcal{G}_i$. For example, if $i = 1$, it means that U_{l_i} is the negative pair sampling set from $\mathcal{G}_2$. We use L1 norm as $d(\cdot, \cdot)$—the distance of node embeddings. Specially, our negative sampling strategy is sort-then-select. To improve the quality of negative sampling and the alignment performance, we sample nodes in $\mathcal{G}_2$ with closest embeddings to the anchor node l_1's embedding in $\mathcal{G}_1$ as U_{l_1} in the next epoch. If n_2 is too large, a bi-level negative sampling strategy can be adopted. We first randomly sample a fixed number of negative nodes and then use the same sort-then-select strategy to conduct the second level negative sampling. The final loss for BRIGHT is:

$$\mathcal{J} = \mathcal{J}_1 + \mathcal{J}_2 \tag{18}$$

After BRIGHT converges, we compute the value $\mathrm{S}(x, a)$ in alignment matrix $\mathbf{S}$ as follows:

$$\mathrm{S}(x, a) = e^{-d(x,a)} \tag{19}$$

where $d(x, a)$ is the L1 norm distance between node a in $\mathcal{G}_1$ and node x in $\mathcal{G}_2$.

4.5 Time Complexity

The running time of BRIGHT can be split into three parts:

- Each iteration of RWR (Eq. (11)) has the time complexity $O(m_i|\mathcal{L}|)$, where m_i is the edge number of $\mathcal{G}_i$ and $\mathcal{L}$ is the anchor link set.
- The negative sampling process in $\mathcal{G}_i$ includes calculating embedding distance and finding $|U_l|$ negative samples with smallest distances for anchor link l. Its time complexity is $O(|\mathcal{L}|kn_i + n_i|\mathcal{L}||U_l|)$, where k is the node embedding dimension, n_i is the number of nodes in $\mathcal{G}_i$ and U_l is the negative sample set for anchor link $l \in \mathcal{L}$.

- For one linear layer or one GCN layer, we only consider the forward propagation same as other deep learning methods [11]. The time complexities for network $\mathcal{G}_i$ are $O(n_i|\mathcal{L}|k)$ and $O(m_i d_f k)$ respectively, where d_f is the input node feature dimension.

5 EXPERIMENT

We apply the proposed BRIGHT to network alignment task under different circumstances. We evaluate it in the following aspects:

- Q1. How effective is our proposed model BRIGHT in both plain network alignment task and attributed network alignment task?
- Q2. To what extent does our model benefit from the RWR embedding module?
- Q3. In the plain network alignment scenario, when the number of anchor links decreases quickly, how can BRIGHT-U avoid the space disparity issue and maintain a relatively high performance compared to other embedding based methods?
- Q4. How can BRIGHT be extended to the unsupervised scenario by generating candidate anchor links for the model?

5.1 Experimental Setup

5.1.1 Dataset description. Our method is evaluated mainly on five datasets and four scenarios,which are distinguished by task categories. The datasets statics are summarized in Table 2 and the brief description of each dataset is presented as following:

Table 2: Datasets Summary.

Categories	Networks	# of Nodes	# of Edges	# of Attributes
Plain Networks	*Foursquare*	5,313	54,233	—
	Twitter	5,120	130,575	—
	ACM	9,916	44,808	—
	DBLP	9,872	39,561	—
Attributed Networks	*ACM(A)*	9,916	44,808	17
	DBLP(A)	9,872	39,561	17
	Cora-1	2,708	5,806	1,433
	Cora-2	2,708	4,547	1,433

- *Foursquare*: A location-based online social network. The original dataset is collected by [33]. Nodes represent 5,313 users and edges represent 54,233 follower/followee relationships. This dataset is used in plain network alignment task with *Twitter* dataset.
- *Twitter*: A popular microblogging social network, which is also collected by [33]. Nodes are 5,120 users and edges are 130,575 follower/followee relationships.
- *DBLP*: A dataset collected in 2016 [26] and it contains 3,272,991 papers with a list of authors and venues. It is transformed from the original paper citation network into a co-authorship network, which contains 9,872 authors and 39,561 co-authorships. With *ACM* dataset, this dataset is used in both plain network alignment task and attributed network alignment task. With the numbers of papers published by an author in 17 venues functioning as attributes, we denote this dataset in the attributed network alignment task as *DBLP(A)*.
- *ACM*: A dataset also collected in 2016 [26]. It contains 2,381,688 papers with authors and their corresponding venues. It is transformed into a co-authorship network with 9,916 authors and 44,808 co-authorships. Similar to the *DBLP* dataset, we denote the attributed version dataset as *ACM(A)*.
- *Cora*: A citation network where nodes represent documents and edges represent the citations among documents. Each node has a bag-of-words binary feature vector [31].

With these datasets, we construct four scenarios of network alignment distinguished by network categories: ***S1-S2*** are plain network alignment task and ***S3-S4*** are attributed network alignment task.

- ***S1. DBLP vs. ACM*** 6,325 common authors exist across two co-author networks, which are used as the ground-truth. ***S1*** aims to find author correspondences across *ACM* and *DBLP*.
- ***S2. Foursquare vs. Twitter*** There exist 1,609 common users which are used as the ground-truth. In this scenario, we try to align users in social networks.
- ***S3. DBLP(A) vs. ACM(A)*** This is attributed network alignment task for ***S1***. Node attributes are given in this scenario.
- ***S4. Cora-1 vs. Cora-2*** This is an attributed network alignment task. Two noisy permutation networks-*Cora-1* and *Cora-2* are generated from the cora citation network. We first insert 10% edges into *Cora-1* and remove 15% edges from *Cora-2*. Then, we add 10% noises into node attribute matrices of *Cora-1* and *Cora-2* separately. The goal is to align corresponding papers.

In ***S1-S4***, we use 20% of the ground-truth as the training data and test on the rest of the ground-truth.

5.1.2 Baselines. We use six recent methods as baselines. Since some methods do not use attribute information and are specially designed for plain network alignment task, we divide baselines into two groups for the sake of fairness. The plain network alignment baseline group includes (1) IONE [16]; (2) CrossMNA [3]; and (3) FINAL-P [34]. The attributed network alignment baseline group includes the remaining three methods:(1) REGAL [8]; (2) FINAL-N [34]; and (3) NetTrans [35]. All baselines are run with their official code. Detailed descriptions of baselines are as the following:
For the plain network alignment task:

- IONE: IONE formulates the edges in the network as follower-ship and followee-ship. IONE preserves the proximity of users with a similar set of followers or followees in the embedding space. In addition, it uses anchor link as a constraint to optimize the embedding.
- CrossMNA: CrossMNA conducts embeddings by minimizing the sum of graph reconstructive loss of different networks and uses a linear transformation between nodes across networks to perform alignment.
- FINAL-P: FINAL-P uses the neighborhood topology consistency to optimize the alignment matrix. It is a typical consistency optimization based network alignment method.

For the attributed network alignment task:

- REGAL: REGAL is an unsupervised method to capture the structure similarity between networks based on cross-network matrix factorization. REGAL applies to both attributed and

Table 3: Performance on Plain and Attributed Network Alignment (Higher is better).

Plain Task	***DBLP vs. ACM***				***Foursquare vs. Twitter***			
Metrics	Hit@1	Hit@10	Hit@30	MRR	Hit@1	Hit@10	Hit@30	MRR
CrossMNA	7.90%	62.53%	79.48%	23.42%	0.00%	3.26%	12.03%	1.48%
IONE	30.91%	74.25%	84.11%	46.26%	4.50%	16.69%	27.80%	8.56%
FINAL-P	19.49%	68.75%	81.23%	35.00%	4.97%	22.22%	32.25%	10.31%
BRIGHT-U	40.45%	81.26%	84.13%	53.85%	6.37%	25.24%	33.54%	13.04%
Attributed Task	***DBLP(A) vs. ACM(A)***				***Cora-1 vs. Cora-2***			
Metrics	Hit@1	Hit@10	Hit@30	MRR	Hit@1	Hit@10	Hit@30	MRR
REGAL	36.26%	60.36%	69.51%	44.92%	45.66%	60.90%	69.21%	51.11%
FINAL-N	38.18%	79.74%	89.07%	52.15%	86.29%	91.32%	91.37%	88.70%
NetTrans	11.84%	84.11%	94.53%	30.11%	27.56%	90.95%	97.51%	49.67%
BRIGHT-A	45.26%	86.76%	92.17%	59.87%	83.85%	99.08%	99.68%	90.41%

plain network alignment. Here, we apply it to the attributed network task.

- FINAL-N: FINAL-N is the attributed version of FINAL-P, which relies on both topology consistency and node/edge attribute consistency to optimize the objective function.
- NetTrans: NetTrans is a GCN based embedding method, which solves the attributed network alignment task from the network transformation view.

5.1.3 Metrics. In the experiment, we adopt two metrics commonly used in the literature to evaluate performance of different methods. The first is Hit@K: given a testing node a from $\mathcal{G}_1$, if the corresponding node x from $\mathcal{G}_2$ is among the top-K most similar list of all nodes in $\mathcal{G}_2$ returned by the algorithm, we treat x as a hit. Hit@K for the test set is the sum of hits divided by the size of test set. The second metrics is Mean Reciprocal Rank (MRR) in entity linking community [24].

5.1.4 Implementation details. Adam [10] optimizer is used with a learning rate 0.0001 to train the model. The dimension of node embeddings in all methods is set as 128. The epoch number for BRIGHT is 250, the size of negative sample set for each anchor link in the training set is 500 and the margin parameter γ is 10. For all baselines, hyper-parameters are set as default in their official code except the dimension of node embeddings for a fair comparison. All the code of BRIGHT is implemented by Pytorch-Geometrics tool [6] and run on a Nvidia GTX 1080-Ti.

5.2 Performance on plain and Attributed Network Alignment

In this subsection, we compare BRIGHT with baselines in scenarios ***S1-S4***. The results of these methods are presented in Table 3. We give separate discussions on plain network alignment and attributed network alignment.

5.2.1 Plain network alignment task. In plain tasks (***S1-S2***), BRIGHT-U is compared with three baselines: CrossMNA, IONE and FINAL-P, the results of which are shown in the upper part of Table 3. For ***DBLP vs. ACM***, embedding based method IONE has the best performance among baselines and BRIGHT-U has a 10% improvement in Hit@1 and a 7.5% improvement in MRR over IONE. For ***Foursquare vs. Twitter***, consistency optimization based method FINAL-P is the best baseline and BRIGHT-U has an up to 3% improvement in both Hit@30 and MRR compared to it. According to Table 2, ***Foursquare vs. Twitter*** is denser than ***DBLP vs. ACM***, which may cause node embeddings in IONE difficult to be distinguished. However, BRIGHT-U still outperforms all three baselines in all Hit@Ks and MRR for both two scenarios.

5.2.2 Attributed network alignment task. In attributed tasks (***S3-S4***), first, we can see that with the help of node attributes, FINAL-N has improved FINAL-P's performance by 8% in Hit@30 and by 19% in Hit@1 on ***DBLP(A) vs. ACM(A)***. For this scenario, BRIGHT-A outperforms FINAL-N by 7% in Hit@1, Hit@10 and MRR. NetTrans has better performance in Hit@10 and Hit@30 than the remaining two baselines. While NetTrans beats BRIGHT-A in Hit@30 by 2%, BRIGHT-A has an advantage about 30% in Hit@1 and MRR over NetTrans. For the scenario ***Cora-1 vs. Cora-2***, BRIGHT-A achieves near 100% (99.08% and 99.68%) in Hit@10 and Hit@30. In addition, we can observe that all four methods perform well in this scenario. Notice that **Cora-2** is a noisy permutation of **Cora-1** and the two networks share the similar distributions. The alignment task of such kind of networks (i.e., one network is noisy permutation of the other) is easier than that of networks from different sources.

5.3 Ablation Study for BRIGHT

Table 4: Ablation Study for BRIGHT-U.

plain Task	***DBLP vs. ACM***		***Foursquare vs. Twitter***	
Metrics	Hit@10	MRR	Hit@10	MRR
BRIGHT-U(SPD)	75.75%	48.78%	6.06%	3.07%
BRIGHT-U	81.26%	53.85%	25.24%	13.04%

The proposed model BRIGHT is nimble. That is, the core for BRIGHT-U is RWR and BRIGHT-A only adds a GCN module based on BRIGHT-U. For BRIGHT-U, we replace RWR with the shortest path distance. From Table 4, we can observe that BRIGHT-U with RWR outperforms the shortest path distance version-BRIGHT-U (SPD) in both ***S1-S2***, especially in ***S2*** with an up to 20% improvement in Hit@10. The reason is that RWR is more robust than shortest path distance in dense networks.

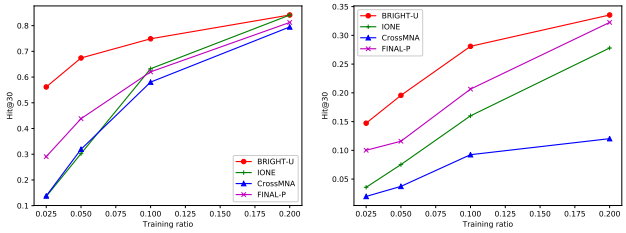
Table 5: Ablation Study for BRIGHT-A.

Attributed Task	*DBLP(A) vs. ACM(A)*		*Cora-1 vs. Cora-2*	
Metrics	Hit@10	MRR	Hit@10	MRR
BRIGHT-A(-RWR)	79.43%	51.61%	99.08%	90.12%
BRIGHT-A(-RWR:3500)	84.31%	58.01%	99.08%	90.12%
BRIGHT-A	86.76%	59.87%	99.08%	90.41%

For BRIGHT-A, we delete the BRIGHT-U/RWR (upper part in Figure 5). As shown in Table 5, BRIGHT-A (-RWR) has almost same performance with BRIGHT-A in ***Cora-1 vs. Cora-2*** scenario, where two networks are the noisy permutation of each other and node attributes contribute most to the alignment process. For scenario ***ACM(A)-DBLP(A)***, BRIGHT-A (-RWR) can not converge in the same epoch number (250) as BRIGHT-A. So, we extend the epoch number to 3500 for BRIGHT-A (-RWR) as BRIGHT-A (-RWR:3500). As shown in Table 5, BRIGHT-A outperforms BRIGHT-A (-RWR:3500) by 2% in Hit@10 and MRR. In BRIGHT-A, node attributes play an important role and RWR helps it to converge faster.

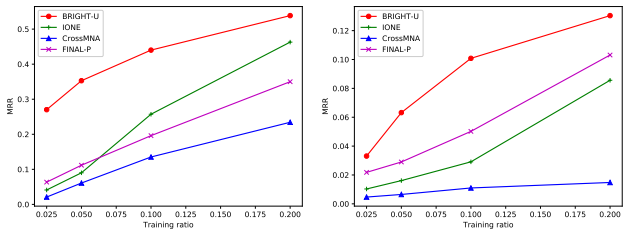
5.4 Plain Network Alignment with Small Training Ratio

In general, the difficulty of network alignment task increases with less provided input information. For example, the plain network alignment task is more difficult than attributed network alignment task. Therefore, given two plain networks from different sources, if the number of anchor links decreases sharply, the difficulty of alignment will increase. In addition, embedding based methods will suffer from the severe *space disparity issue* with very few anchor links. In this subsection, we make the anchor links number decreases quickly by setting the training ratio from 0.2 to 0.025 ([0.025, 0.05, 0.1, 0.2]). As shown in Figure 6 and Figure 7, comparing to consistency optimization based method FINAL-P, the performances of two embedding based alogorithms (i.e., IONE and CrossMNA) decline quickly when the training ratio decreases, which demonstrates the severe space disparity issue they are faced with. However, even with a very small training ratio setting (0.025), BRIGHT-U still maintains a relatively high performance, which indicates that BRIGHT-U successfully mitigates the space disparity issue .



(a) Hit@30 with small training ratio on ***S1***(b) Hit@30 with small training ratio on ***S2***.

Figure 6: Hit@30 with small training ratio on plain network alignment task.

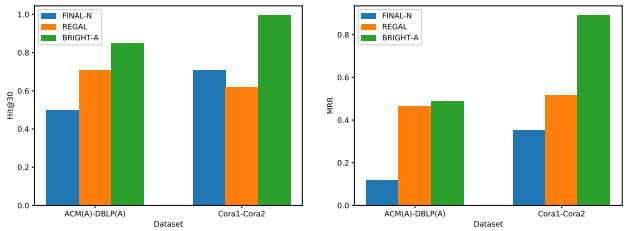


(a) MRR with small training ratio on ***S1***. (b) MRR with small training ratio on ***S2***.

Figure 7: MRR with small training ratio on plain network alignment task.

5.5 Study on Unsupervised Attributed Network Alignment

Both BRIGHT-U and BRIGHT-A are designed for semi-supervised network alignment task. We further extend BRIGHT to the unsupervised network alignment task in this subsection. In particular, we use the attribute similarity to generate *candidate anchor links* and feed these *candidate anchor links* into BRIGHT-A. The detail of this step is as follows. After calculating attribute similarity, we select the most similar k pairs of node across networks ($k = 0.1 \min\{n_1, n_2\}$) where n_1, n_2 are sizes of two networks. We compares BRIGHT-A with two baselines, including REGAL and FINAL. The former is an unsupervised method, and the latter provides a specific way to operate in the unsupervised setting. As shown in Figure 8, BRIGHT-A outperforms unsupervised FINAL and REGAL in both Hit@30 and MRR.



(a) Hit@30 (b) MRR

Figure 8: Performance on unsupervised attributed network alignment task.

6 RELATED WORKS

6.1 Network Representation Learning

Network embedding is applied in many applications like category characterization [37] or temporal prediction [17]. DeepWalk [21] is the first to use random walk to conduct representation learning in graph. Node2vec [7] introduces an interpolation strategy when performing random walks. LINE [25] takes both the first order and the second order proximity into consideration. SPARC [37] is a For

heterogeneous networks, Graph Neural Network (GNN) has been studied, which can integrate attribute information in the learning process. According to different aggregation methods, GNN includes Graph Convolution Network [11], Graph Attention Network [29], and many more. Personalized Pagerank [1] is utilized in GNN [12] to sample better nodes to aggregate information. PGNN [32] is designed for discriminating nodes with same structure in different positions by randomly selecting some nodes as anchors. All of the above network representation learning methods primarily focus on a single network and directly adopting them in the network alignment task bring the space disparity issue.

6.2 Network Alignment

Network alignment is applied in many domains such as database schema matching [19], bioinformatics [9, 15, 18, 22] and computer vision [4]. Network alignment was initially formulated as Koopmans-Beckmann's quadratic assignment problem [13]. Based on the consistency assumption that one network is noisy permutation of the other, topological graph alignment is treated as graph matching problem in [28]. With attribute information, FINAL [34] uses attribute and structure consistency to formulate the network assignment problem as a convex quadratic optimization problem, which is generalized to high order in [16, 20]. REGAL [8] is a matrix factorization method by building cross-network similarity matrix. As network representation learning has become popular, [33] proposes a network embedding based method which uses anchor link as regularization. With the assistance of following relationship, IONE [16] is specially designed for social network alignment. CrossMNA [3] handles the alignment between a set of networks, where the space disparity issues is handled by a linear transformation to transform the embeddings from different networks. For attributed network alignment, Origin [36] uses the non-rigid point-set registration. ADMIRING [38] is an adversarial method. NetTrans [35] leverages graph convolutional network for the alignment task from the network transformation view.

7 CONCLUSION

In this paper, we study network alignment problem. We focus two categories of methods and reveal two main limitations, including (1) the restrictive propagation limitation; and (2) the space disparity issue behind embedding based methods. We further bridge them by showing that embedding based methods are essentially the *relaxed* version of the consistency optimization based methods. Based on the analysis, we propose an embedding based family of network alignment algorithms named BRIGHT, including BRIGHT-U for plain network alignment and BRIGHT-A for attributed network alignment. Extensive experiments demonstrate that BRIGHT-U can outperform the state-of-the-arts by an up to 10% improvement in Hit@10 and BRIGHT-A can align the Cora network and its noisy permutation near perfectly.

8 ACKNOWLEDGEMENT

This work is supported by National Science Foundation under grant No. 1947135, and by the NSF Program on Fairness in AI in collaboration with Amazon under award No. 1939725. The content of the information in this document does not necessarily reflect the position or the policy of the Government or Amazon, and no official endorsement should be inferred. The U.S. Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation here on.

REFERENCES

[1] Bahman Bahmani, Abdur Chowdhury, and Ashish Goel. 2010. Fast incremental and personalized pagerank. *arXiv preprint arXiv:1006.2880* (2010).
[2] Xuezhi Cao and Yong Yu. 2016. Joint user modeling across aligned heterogeneous sites. In *Proceedings of the 10th ACM Conference on Recommender Systems*. 83–90.
[3] Xiaokai Chu, Xinxin Fan, Di Yao, Zhihua Zhu, Jianhui Huang, and Jingping Bi. 2019. Cross-Network Embedding for Multi-Network Alignment. In *The World Wide Web Conference (WWW '19)*. ACM, New York, NY, USA, 273–284. https://doi.org/10.1145/3308558.3313499
[4] Donatello Conte, Pasquale Foggia, Carlo Sansone, and Mario Vento. 2004. Thirty years of graph matching in pattern recognition. *International journal of pattern recognition and artificial intelligence* 18, 03 (2004), 265–298.
[5] Fazle E Faisal, Lei Meng, Joseph Crawford, and Tijana Milenković. 2015. The post-genomic era of biological network alignment. *EURASIP Journal on Bioinformatics and Systems Biology* 2015, 1 (2015), 3.
[6] Matthias Fey and Jan E. Lenssen. 2019. Fast Graph Representation Learning with PyTorch Geometric. In *ICLR Workshop on Representation Learning on Graphs and Manifolds*.
[7] Aditya Grover and Jure Leskovec. 2016. node2vec: Scalable Feature Learning for Networks. *CoRR* abs/1607.00653 (2016). arXiv:1607.00653 http://arxiv.org/abs/1607.00653
[8] Mark Heimann, Haoming Shen, Tara Safavi, and Danai Koutra. 2018. REGAL: Representation Learning-based Graph Alignment. In *Proceedings of the 27th ACM International Conference on Information and Knowledge Management (CIKM '18)*. ACM, New York, NY, USA, 117–126. https://doi.org/10.1145/3269206.3271788
[9] Ehsan Kazemi, Hamed Hassani, Matthias Grossglauser, and Hassan Pezeshgi Modarres. 2016. PROPER: global protein interaction network alignment through percolation matching. *BMC bioinformatics* 17, 1 (2016), 527.
[10] Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* (2014).
[11] Thomas N Kipf and Max Welling. 2016. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907* (2016).
[12] Johannes Klicpera, Aleksandar Bojchevski, and Stephan Günnemann. 2018. Predict then propagate: Graph neural networks meet personalized pagerank. *arXiv preprint arXiv:1810.05997* (2018).
[13] Tjalling C. Koopmans and Martin Beckmann. 1957. Assignment Problems and the Location of Economic Activities. *Econometrica* 25, 1 (1957), 53–76. http://www.jstor.org/stable/1907742
[14] Danai Koutra, Hanghang Tong, and David Lubensky. 2013. Big-align: Fast bipartite graph alignment. In *2013 IEEE 13th International Conference on Data Mining*. IEEE, 389–398.
[15] Oleksii Kuchaiev, Tijana Milenković, Vesna Memišević, Wayne Hayes, and Nataša Pržulj. 2010. Topological network alignment uncovers biological function and phylogeny. *Journal of the Royal Society Interface* 7, 50 (2010), 1341–1354.
[16] Li Liu, William K Cheung, Xin Li, and Lejian Liao. [n.d.]. Aligning Users across Social Networks Using Network Embedding.
[17] Zhining Liu, Dawei Zhou, Yada Zhu, Jinjie Gu, and Jingrui He. 2020. Towards fine-grained temporal network representation via time-reinforced random walk. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 34. 4973–4980.
[18] Noël Malod-Dognin and Nataša Pržulj. 2015. L-GRAAL: Lagrangian graphlet-based network aligner. *Bioinformatics* 31, 13 (2015), 2182–2189.
[19] Sergey Melnik, Hector Garcia-Molina, and Erhard Rahm. [n.d.]. Similarity flooding: A versatile graph matching algorithm and its application to schema matching. In *Proceedings 18th International Conference on Data Engineering*. IEEE, 117–128.
[20] Shahin Mohammadi, David F. Gleich, Tamara G. Kolda, and Ananth Grama. 2017. Triangular Alignment (TAME): A Tensor-Based Approach for Higher-Order Network Alignment. *IEEE/ACM Trans. Comput. Biology Bioinform.* 14, 6 (2017), 1446–1458.
[21] Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. 2014. Deepwalk: Online learning of social representations. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*. 701–710.
[22] Rohit Singh, Jinbo Xu, and Bonnie Berger. 2007. Pairwise global alignment of protein interaction networks by matching neighborhood topology. In *Annual International Conference on Research in Computational Molecular Biology*. Springer, 16–31.
[23] Rohit Singh, Jinbo Xu, and Bonnie Berger. 2008. Global alignment of multiple protein interaction networks with application to functional orthology detection. *Proceedings of the National Academy of Sciences* 105, 35 (2008), 12763–12768.
[24] Zequn Sun, Chengming Wang, Wei Hu, Muhao Chen, Jian Dai, Wei Zhang, and Yuzhong Qu. 2020. Knowledge graph alignment network with gated multi-hop

neighborhood aggregation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 34. 222–229.

[25] Jian Tang, Meng Qu, Mingzhe Wang, Ming Zhang, Jun Yan, and Qiaozhu Mei. 2015. Line: Large-scale information network embedding. In *Proceedings of the 24th international conference on world wide web*. 1067–1077.

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

[27] Hanghang Tong, Christos Faloutsos, and Jia-Yu Pan. 2006. Fast random walk with restart and its applications. In *Sixth international conference on data mining (ICDM'06)*. IEEE, 613–622.

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

[29] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. 2017. Graph Attention Networks. arXiv:stat.ML/1710.10903

[30] Zhichun Wang, Qingsong Lv, Xiaohan Lan, and Yu Zhang. 2018. Cross-lingual knowledge graph alignment via graph convolutional networks. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*. 349–357.

[31] Zhilin Yang, William W. Cohen, and Ruslan Salakhutdinov. 2016. Revisiting Semi-Supervised Learning with Graph Embeddings. *CoRR* abs/1603.08861 (2016). arXiv:1603.08861 http://arxiv.org/abs/1603.08861

[32] Jiaxuan You, Rex Ying, and Jure Leskovec. 2019. Position-aware Graph Neural Networks. *CoRR* abs/1906.04817 (2019). arXiv:1906.04817 http://arxiv.org/abs/1906.04817

[33] Jiawei Zhang and S Yu Philip. 2015. Integrated anchor and social link predictions across social networks. In *Twenty-Fourth International Joint Conference on Artificial Intelligence*.

[34] Si Zhang and Hanghang Tong. 2016. Final: Fast attributed network alignment. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 1345–1354.

[35] Si Zhang, Hanghang Tong, Yinglong Xia, Liang Xiong, and Jiejun Xu. 2020. NetTrans: Neural Cross-Network Transformation. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining (KDD '20)*. Association for Computing Machinery, New York, NY, USA, 986–996. https://doi.org/10.1145/3394486.3403141

[36] Si Zhang, Hanghang Tong, Jiejun Xu, Yifan Hu, and Ross Maciejewski. 2019. Origin: Non-rigid network alignment. In *2019 IEEE International Conference on Big Data (Big Data)*. IEEE, 998–1007.

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

[38] Qinghai Zhou, Liangyue Li, Nan Cao, Lei Ying, and Hanghang Tong. 2019. ADMIRING: Adversarial multi-network mining. In *2019 IEEE International Conference on Data Mining (ICDM)*. IEEE, 1522–1527.