ORIGIN: Non-Rigid Network Alignment

Si Zhang*, Hanghang Tong*, Jiejun Xu†, Yifan Hu‡, and Ross Maciejewski§
*University of Illinois at Urbana-Champaign, {sizhang2, htong}@illinois.edu
†HRL Laboratories, jxu@hrl.com
‡Yahoo Labs, yifanhu@yahoo.com
§Arizona State University, rmacieje@asu.edu

Abstract—**Network alignment is a fundamental task in many high-impact applications. Most of the existing approaches either explicitly or implicitly consider the alignment matrix as a linear transformation to map one network to another, and might overlook the complicated alignment relationship across networks. On the other hand, node representation learning based alignment methods are hampered by the incomparability among the node representations of different networks. In this paper, we propose a unified semi-supervised deep model (ORIGIN) that simultaneously finds the non-rigid network alignment and learns node representations in multiple networks in a mutually beneficial way. The key idea is to learn node representations by the effective graph convolutional networks, which subsequently enable us to formulate network alignment as a point set alignment problem. The proposed method offers two distinctive advantages. First (*node representations*), unlike the existing graph convolutional networks that aggregate the node information within a single network, we can effectively aggregate the auxiliary information from multiple sources, achieving *far-reaching* node representations. Second (*network alignment*), guided by the high-quality node representations, our proposed non-rigid point set alignment approach overcomes the bottleneck of the linear transformation assumption. We conduct extensive experiments that demonstrate the proposed non-rigid alignment method is (1) effective, outperforming both the state-of-the-art linear transformation-based methods and node representation based methods, and (2) efficient, with a comparable computational time between the proposed multi-network representation learning component and its single-network counterpart.**

Index Terms—**network alignment, non-rigid, graph convolutional networks**

I. Introduction

Multiple networks naturally appear in many areas, ranging from social networks on various platforms, protein-protein interaction networks of different species, transaction networks at different financial institutes to the knowledge graphs constructed by different knowledge bases and the sensored networks derived from multimodal sensors (e.g., Lidar). Network alignment which aims to find the node correspondence across multiple networks is a fundamental task to integrate multiple networks into a worldview of what the input data represents. Consequently, it has drawn much attention in numerous applications. For example, by identifying the overlapping entities across multiple incomplete knowledge graphs, a unified knowledge graph can be constructed to aid knowledge completion [1]. Since many adversarial activities (e.g., smuggling) in different contexts are often covert in multiple domains, they are not quite detectable in each of the input networks alone. Network alignment can integrate these isolated networks and amplify the deceptive adversarial activities, so that they are more detectable in the composite network [2].

Despite the extensive works on network alignment, many of them explicitly or implicitly consider the alignment matrix as a linear transformation matrix. For example, many traditional graph matching based methods attempt to solve the Koopmans-Beckmann's quadratic assignment problem [3] or its relaxations, i.e., to maximize $\text{Tr}(\mathbf{S}^T\mathbf{A}_1\mathbf{S}\mathbf{A}_2) + \text{Tr}(\mathbf{H}^T\mathbf{S})$ where $\mathbf{S}, \mathbf{H}$ are the alignment matrix and prior cross-network node similarity matrix respectively, and $\mathbf{A}_1, \mathbf{A}_2$ are the adjacency matrices. Existing works following this path include *Umeyama* [4], *BigAlign* [5], *NetAlign* [6] and *FINAL* [7]. Since $\text{Tr}(\mathbf{S}^T\mathbf{A}_1\mathbf{S}\mathbf{A}_2) = \sum_{i,j}(\mathbf{S}^T\mathbf{A}_1)_{i,j}(\mathbf{A}_2\mathbf{S}^T)_{i,j}$, it can be alternatively viewed as first generating node feature vectors by linear transformations based on the adjacency matrices themselves (i.e., by $\mathbf{S}^T\mathbf{A}_1$ and $\mathbf{A}_2\mathbf{S}^T$ respectively) and then maximizing the inner product similarities between the generated feature vectors (e.g., the i-th row of $\mathbf{S}^T\mathbf{A}_1$ and $\mathbf{A}_2\mathbf{S}^T$). However, these methods bear some fundamental limitations, including (1) the alignment matrix $\mathbf{S}$ is leveraged as a linear transformation matrix and thus might oversimplify the complicated alignment relationships across networks; and (2) the generated feature vectors of nodes are high-dimensional and may fall short in their representation power. To mitigate these issues, instead of directly solving for the alignment matrix, *IONE* [8] and *PALE* [9] learn the low-dimensional node embedding vectors in different networks, based on which the alignment can be further inferred. Nevertheless, these methods ignore the node attributes that are often accompanied in real-world networks. Besides, these approaches suffer from the obstacle that node representations could be arbitrarily and imperfectly rotated and/or translated across different networks so that they might not be directly comparable.

To tackle these limitations, we propose to go beyond the linear transformation assumption, and hypothesize that network alignment and node representation learning are mutually beneficial with each other due to the following reasons. First, *network alignment helps node representation learning*. Intuitively, if nodes are aligned across networks, the structural and attribute information of the nodes in one network can be integrated with the nodes in the other network as the auxiliary information, leading to a *far-reaching* representation learning strategy. Second, *node representation learning helps network alignment*. With the premise that node representations are of high qualities, finding node alignment in the non-Euclidean space (i.e., directly across networks) can be translated to the point set alignment problem[1] in the Euclidean space. This naturally renders the possibility of unveiling the alignment of nodes by inferring the *non-rigid* transformations which are expected to lead to the more accurate alignment.

Armed with these hypotheses, we propose to solve the non-rigid network alignment problem by simultaneously learning node representations across multiple networks. The key ideas are two-fold. First, to learn node representations of multiple networks, we design a new convolutional operator that can aggregate the *multi-sourced* information. Second, in order to align node representations, we propose a semi-supervised multi-view non-rigid point set alignment algorithm that first learns the point set transformation function and then infers the alignment based on the transformed node representations. The main contributions are summarized as follows.

- **Problem Definition.** To our best knowledge, we are the first to address the non-rigid network alignment problem.
- **Model and Algorithms.** We propose a semi-supervised model ORIGIN which is able to simultaneously (1) learn node representations of different networks by multi-graph convolutional networks (*Multi-GCN*) and (2) unveil the non-rigid network alignment across multiple networks.
- **Evaluations.** Extensive experiments on real-world networks demonstrate that our proposed alignment approach (1) outperforms both the existing linear transformation based methods and node representation based methods, and (2) is efficient for node representation learning with a comparable computational time between the proposed Multi-GCN and its single-network counterpart.

II. Problem Definition

Table 1 summarizes the main symbols and notations used throughout the paper. We use the bold uppercase letters to denote matrices (e.g., $\mathbf{X}$), bold lowercase letters (e.g., $\mathbf{x}$) for vectors and letters not in bold for scalars (e.g., α). We use $\mathbf{X}(u,v)$ to denote the entry at the intersection of the u-th row and v-th column of the matrix $\mathbf{X}$. Besides, we express $\mathbf{x}_u = \mathbf{X}(u,:)$ as the u-th row of $\mathbf{X}$ and $\mathbf{X}(:,v)$ as the v-th column of $\mathbf{X}$. We denote the transpose of matrix $\mathbf{X}$ as $\mathbf{X}^T$ and the trace of matrix $\mathbf{X}$ as $\text{Tr}(\mathbf{X})$.

A. Non-Rigid Network Alignment Problem

The concept of *non-rigid transformation* is rooted in the point set alignment problem to align the 2D or 3D point sets such that one point set can be maximally overlapped with another point set [10]. Unlike the linear or affine transformations which are restricted to some explicitly expressed transformation functions, non-rigid transformation has more flexibility to unveil the complicated alignment among point sets as it does not require any specific form of the transformation functions.

[1]We consider node representations as the point sets in the Euclidean space.

TABLE I: Symbols and Notations

Symbols	Definition
$\mathcal{G}_1$, $\mathcal{G}_2$	the input undirected networks
$\mathbf{A}_1, \mathbf{A}_2$	the adjacency matrices of $\mathcal{G}_1$ and $\mathcal{G}_2$
$\mathbf{X}^0, \mathbf{Y}^0$	the node attribute matrices of $\mathcal{G}_1$ and $\mathcal{G}_2$
$\mathbf{H}$	an optional $n_1 \times n_2$ cross-network node similarity matrix
$\mathbf{I}$	an identity matrix
$\mathbf{S}$	the output $n_1 \times n_2$ alignment matrix
$\mathbf{S}_1, \mathbf{S}_2$	sampled alignment matrices
d	the dimension of the output node representations
$\tilde{\mathbf{X}}, \tilde{\mathbf{Y}}$	the node representation matrices by Inter-GCNs
$\mathbf{X}, \mathbf{Y}$	the node representation matrices by Multi-GCN
$\mathbf{f}$	the non-rigid transformation function on $\mathbf{X}$
$\mathbf{Z}$	the transformed node representation matrix of $\mathcal{G}_1$ by $\mathbf{f}$
κ^1, κ^2	the kernel function for the point view and graph view
$\kappa, \mathbf{K}$	the combined kernel function and its kernel matrix
$\mathcal{L}^+$	the labeled node pairs which are aligned a priori
n_1, n_2	# of nodes in $\mathcal{G}_1, \mathcal{G}_2$
$\alpha, \alpha_1, \alpha_2, \lambda$	the regularization parameters

Inspired by this, we translate the network alignment problem to the point set alignment problem. That is, given the input networks that are known to be the non-Euclidean data [11], we aim to (1) represent the nodes in the Euclidean space so that they can be naturally viewed as point sets, and (2) align nodes in different networks (i.e., different point sets) by inferring the non-rigid transformation among them. Formally, the non-rigid network alignment problem is defined as below.

Problem 1: Non-Rigid Network Alignment.

Given: (1) undirected networks $\mathcal{G}_1 = \{\mathcal{V}_1, \mathbf{A}_1, \mathbf{X}^0\}$ and $\mathcal{G}_2 = \{\mathcal{V}_2, \mathbf{A}_2, \mathbf{Y}^0\}$ with $\mathcal{V}_1, \mathcal{V}_2$ as the node sets where $|\mathcal{V}_1| = n_1, |\mathcal{V}_2| = n_2$, $\mathbf{A}_1, \mathbf{A}_2$ as the adjacency matrices and $\mathbf{X}^0, \mathbf{Y}^0$ as the input node attribute matrices of $\mathcal{G}_1, \mathcal{G}_2$ respectively, (2) a set of labeled node pairs $\mathcal{L}^+ = \{(u_{l_i}, v_{l_i}) | i = 1, \cdots, L\}$ where node u_{l_i} in $\mathcal{G}_1$ is aligned with node v_{l_i} in $\mathcal{G}_2$ a priori, (3) an optional prior cross-network node similarity matrix $\mathbf{H}$.

Find: (1) an $n_1 \times n_2$ soft alignment matrix $\mathbf{S}$ where $\mathbf{S}(u,v)$ represents to what extent node u in $\mathcal{G}_1$ is aligned with node v in $\mathcal{G}_2$, and (2) node representation matrices $\mathbf{Z}, \mathbf{Y}$ of $\mathcal{G}_1, \mathcal{G}_2$.

Remarks. If there is no prior knowledge of the cross-network node similarity matrix, we can alternatively construct $\mathbf{H}$ by some heuristics, such as node degree similarity. Though we consider network alignment problem between two input networks in the paper, it is straightforward to generalize our proposed model to handle multiple network alignment. Specifically, after computing the pair-wise alignment between each pair of networks, we can postprocess (e.g., by [12]) to find the alignment among more than two input networks.

B. Preliminary: Graph Convolutional Networks

Graph neural networks have attracted lots of research interests in the recent years. Among others, graph convolutional networks have achieved great success in node representation learning [13], [14]. The main idea of graph convolutional networks lies in generalizing the traditional convolution operators on the Euclidean data (e.g., images) to graphs such that node information can be aggregated based on the graph structure. Here, we briefly review a spatial-based graph convolutional network (namely *GraphSage* [14]) that will be used as a building block in our ORIGIN model to learn node representations for a single network. Given a graph $\mathcal{G}_1$, each node $u \in \mathcal{V}_1$

aggregates hidden representations from its neighborhood $\mathcal{N}_u$ and combines the aggregated representation with its current representation. Formally, it is formulated as

$$\tilde{\mathbf{x}}^t_{\mathcal{N}_u} = \text{AGGREGATE}_t(\{\tilde{\mathbf{x}}^{t-1}_{u'},\ \forall u' \in \mathcal{N}_u\}) \tag{1}$$

$$\tilde{\mathbf{x}}^t_u = \sigma\left([\tilde{\mathbf{x}}^{t-1}_u \| \tilde{\mathbf{x}}_{\mathcal{N}^t_u}]\mathbf{W}^t\right) \tag{2}$$

where $[\cdot\|\cdot]$ represents the concatenation of two vectors and $\sigma(\cdot)$ is the non-linear activation function. $\tilde{\mathbf{x}}^t_u$ is the representation of node u and $\mathbf{W}^t$ denotes the weight matrix at the t-th layer. Note that when $t = 0$, $\tilde{\mathbf{x}}^0_u$ is initialized by the input attributes of node u, i.e., $\tilde{\mathbf{x}}^0_u = \mathbf{X}^0(u,:)$. Moreover, according to [14], the *GraphSage* model can be instantiated by Mean, LSTM and Pooling aggregators. In this paper, we choose the MEAN aggregator due to its high representation power and simplicity. It is notable that *GraphSage* is capable of mini-batch training by uniformly sampling with replacement a fixed size of the neighboring nodes. For an input network $\mathcal{G}_1$, the unsupervised *GraphSage* minimizes the following loss function based on the SkipGram with negative sampling [15].

$$\mathcal{J}_{\mathcal{G}_1}(\tilde{\mathbf{X}}) = \tag{3}$$
$$\sum_{u \in \mathcal{V}_1} \sum_{u' \in C_u} -\log\left(\sigma(\tilde{\mathbf{x}}^T_u \tilde{\mathbf{x}}_{u'})\right) - Q \cdot \mathbb{E}_{u'_n \sim P_n(u')} \log\left(\sigma(-\tilde{\mathbf{x}}^T_u \tilde{\mathbf{x}}_{u'_n})\right)$$

where $u' \in C_u$ represents that node u' co-occurs with u on a fixed-length random walk, Q defines the number of negative samples and $P_n(u')$ is the negative sampling distribution.

III. Proposed Model

In this section, we present ORIGIN, a deep semi-supervised model that can simultaneously learn the node representations and find the non-rigid alignment across the input networks in a symbiotic way. We start by proposing a graph convolutional network model for multiple networks (*Multi-GCN*) to learn *far-reaching* node representations of the input networks. Next, we introduce a multi-view approach to unveiling the non-rigid alignment among the nodes which are represented by the point sets in the Euclidean space, followed by the optimization algorithm to effectively and efficiently learn both node representations and node alignment. The overall framework of ORIGIN is shown in Figure 1.

A. Node Representation Learning for Multiple Networks

A - Aggregation and Combination. Many existing spatial-based graph convolutional networks (e.g., *GraphSage* [14]) essentially define two operators: (1) *Intra-Aggregation* that aggregates node hidden representations (or node attribute information at the first layer) from the neighboring nodes (e.g., Eq. (1)) underlying a single network and (2) *Intra-Combination* that combines the current hidden representation with the resultant aggregated representation of the node as the updated node representation (e.g., Eq. (2)). However, this might not be able to provide sufficiently informative node representations in the multiple networks scenario given that multiple networks might contain some complementary information for each other. To remedy this restrictive situation, in addition to the separate graph convolutional networks for single networks (named as Intra-GCNs), we propose an *Inter-GCN* component that integrates node representations across different networks. The intuition is to view the node alignment as the probabilistic cross-network node similarity and bridge different networks such that nodes in one network can be considered as the *virtual* neighbors of the nodes in the other network if they are likely to be aligned. For example, if node u in $\mathcal{G}_1$ is similar to node v in $\mathcal{G}_2$ (i.e., likely to be aligned), we can view node v as a *virtual* neighbor of node u. In this way, the representation of node v can be used to aggregate the neighboring node representations for node u, and vice versa. Thus, given two separate Intra-GCNs that aggregate node information in the same network and output the node representations $\tilde{\mathbf{x}}_u, \tilde{\mathbf{y}}_v \in \mathbb{R}^d,\ \forall u \in \mathcal{V}_1, v \in \mathcal{V}_2$, we define the cross-network aggregation as

$$\hat{\mathbf{x}}_u = \text{AGGREGATE}_{\text{cross}}(\tilde{\mathbf{x}}_u) = \sum_{v \in \mathcal{V}_2} \mathbf{S}(u,v)\tilde{\mathbf{y}}_v \tag{4}$$

$$\hat{\mathbf{y}}_v = \text{AGGREGATE}_{\text{cross}}(\tilde{\mathbf{y}}_v) = \sum_{u \in \mathcal{V}_1} \mathbf{S}(u,v)\tilde{\mathbf{x}}_u \tag{5}$$

where $\mathbf{S}$ is the alignment matrix and $\mathbf{S}(u,v)$ represents to what extent node u in $\mathcal{G}_1$ and node v in $\mathcal{G}_2$ are aligned.

It is crucial to properly leverage the alignment matrix $\mathbf{S}$ in Eq. (4) and Eq. (5) from the following two perspectives. First (*aggregation efficiency*), the node alignment matrix is often quite dense, leading to an $O(nd)$ time complexity to compute the cross-network aggregation for each node (e.g., $\hat{\mathbf{x}}_u$) which is prohibitive especially for large-scale networks. Second (*aggregation localization*), when $\mathbf{S}$ is dense, Eq. (4) and Eq. (5) aggregate the representations of most or even all of the nodes from one network for the other, i.e., smoothing globally over the networks. This will further make the aggregated representations of different nodes less distinguishable. To overcome these issues, since we are given the labeled node pairs $\mathcal{L}^+ = \{(u_{l_i}, v_{l_i}) | i = 1, \cdots, L\}$ that indicates which nodes are aligned, we set $\mathbf{S}(u_{l_i}, v) = \mathbf{S}(u, v_{l_i}) = 0$ for all $u \in \mathcal{V}_1$ and $v \in \mathcal{V}_2$ except that $\mathbf{S}(u_{l_i}, v_{l_i}) = 1$. Besides, for all the other nodes whose alignment are unknown (e.g., $u \notin \{u_{l_i}, \forall i = 1, \cdots, L\}, v \notin \{v_{l_i}, \forall i = 1, \cdots, L\}$), we propose to downsample the alignment matrix $\mathbf{S}$ as follows. For each node $u \notin \{u_{l_i}, \forall i = 1, \cdots, L\}$, we only preserve the K largest values $\mathbf{S}(u, v_{q_k}),\ k = 1, \cdots, K$ column-wise from $\mathbf{S}(u,:)$ for Eq. (4) and denote the sampled matrix as $\mathbf{S}_1$. We then normalize it such that $\sum_{k=1}^K \mathbf{S}_1(u, v_{q_k}) = 1$. Similarly, we sample K values $\mathbf{S}(u_{p_k}, v)$ for each node $v \notin \{v_{l_i}, \forall i = 1, \cdots, L\}$ row-wise from $\mathbf{S}(:,v)$ and denote it as $\mathbf{S}_2$ where $\sum_{k=1}^K \mathbf{S}_2(u_{p_k}, v) = 1$ after normalization. This sampling and normalization process is summarized in Algorithm 1. The cross-network aggregation is re-written as

$$\begin{aligned}\hat{\mathbf{x}}_u &= \text{AGGREGATE}_{\text{cross}}(\tilde{\mathbf{x}}_u) = \sum_{k=1}^K \mathbf{S}_1(u, v_{q_k})\tilde{\mathbf{y}}_{v_{q_k}} \\ \hat{\mathbf{y}}_v &= \text{AGGREGATE}_{\text{cross}}(\tilde{\mathbf{y}}_v) = \sum_{k=1}^K \mathbf{S}_2(u_{p_k}, v)\tilde{\mathbf{x}}_{u_{p_k}}\end{aligned} \tag{6}$$

We show an illustrative example in Figure 2 where $K = 2$ and nodes v_{q_1}, v_{q_2} are sampled for cross-network aggregation for $\tilde{\mathbf{x}}_u$ through $\mathbf{S}_1(u, v_{q_1})$ and $\mathbf{S}_1(u, v_{q_2})$ respectively. For

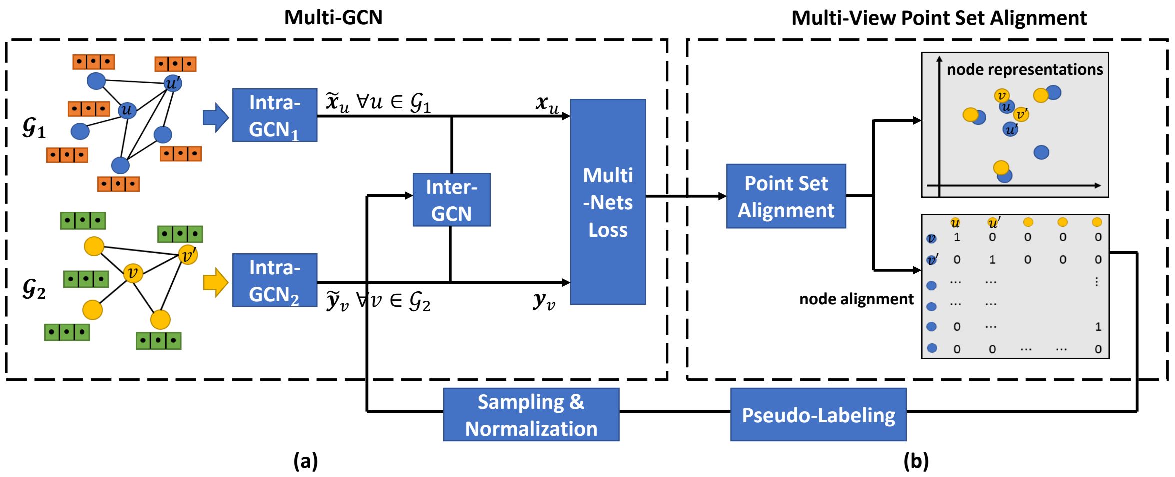


Fig. 1: Illustration of the proposed ORIGIN model. (a) The Multi-GCN module that learns node representations of the input networks by intra-network and cross-network aggregations and combinations. (b) The point set alignment process that first displaces the node representations of $\mathcal{G}_1$ to those of $\mathcal{G}_2$, and then infers node alignment which will be fed back to Multi-GCN.

Algorithm 1 SAMPLE($\mathbf{S}, K$).

```
Input: (1) the current alignment matrix $\mathbf{S}$, (2) the labeled
    node pairs $\mathcal{L}^+$ and (3) the sample size $K$.
Output: sampled alignment matrix $\mathbf{S}_1, \mathbf{S}_2$.
 1: Initialize $\mathbf{S}_1 = \mathbf{S}_2 = \mathbf{S}$;
 2: for $i = 1 \to L$ do
 3:     Set $\mathbf{S}_1(u_{l_i}, v_{l_i}) = \mathbf{S}_2(u_{l_i}, v_{l_i}) = 1$;
 4:     Set $\mathbf{S}_1(u, v_{l_i}) = \mathbf{S}_2(u_{l_i}, v) = 0,\ \forall u \neq u_{l_i}, v \neq v_{l_i}$;
 5: end for
 6: Preserve top-$K$ nonzero elements in $\mathbf{S}_1(u,:)$, $\forall u \notin$
    $\{u_{l_i}, \forall i = 1, \cdots, L\}$;
 7: Preserve top-$K$ nonzero elements in $\mathbf{S}_2(:,v)$, $\forall v \notin$
    $\{v_{l_i}, \forall i = 1, \cdots, L\}$;
 8: Normalize $\mathbf{S}_1(u,:)$, $\forall u = 1, \cdots, n_1$;
 9: Normalize $\mathbf{S}_2(:,v)$, $\forall v = 1, \cdots, n_2$;
10: Output $\mathbf{S}_1$, $\mathbf{S}_2$.
```

cross-network combination, we use

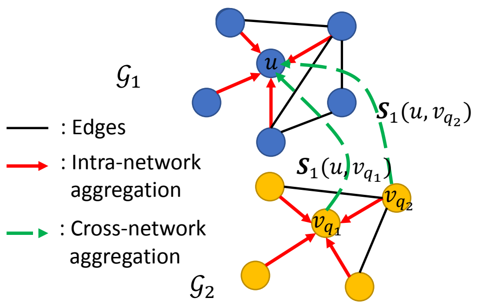
$$\begin{aligned} \mathbf{x}_u &= \text{COMBINE}_{\text{cross}}(\tilde{\mathbf{x}}_u, \hat{\mathbf{x}}_u) = [\tilde{\mathbf{x}}_u \| \hat{\mathbf{x}}_u]\mathbf{W}_{cross} + \mathbf{b}_1 \\ \mathbf{y}_v &= \text{COMBINE}_{\text{cross}}(\tilde{\mathbf{y}}_v, \hat{\mathbf{y}}_v) = [\tilde{\mathbf{y}}_v \| \hat{\mathbf{y}}_v]\mathbf{W}_{cross} + \mathbf{b}_2 \end{aligned} \tag{7}$$

where $\mathbf{x}_u, \mathbf{y}_v \in \mathbb{R}^d$ are the output node representations by the proposed Multi-GCN model. The weight matrix $\mathbf{W}_{\text{cross}} \in \mathbb{R}^{2d \times d}$ is shared in both equations as it basically measures how to combine the representations learned by Intra-GCNs and by Inter-GCN for cross-network combinations.

B - Loss Functions. To learn the *far-reaching* node representations that can simultaneously maintain the local structural information within a single network and the cross-network representation consistency, we aim to minimize the loss function

$$\mathcal{J}_{GCN} = \mathcal{J}_{\mathcal{G}_1}(\mathbf{X}) + \mathcal{J}_{\mathcal{G}_2}(\mathbf{Y}) + \lambda \mathcal{J}_{\text{cross}}(\mathbf{X}, \mathbf{Y}) \tag{8}$$

where $\mathcal{J}_{\mathcal{G}_1}(\mathbf{X}), \mathcal{J}_{\mathcal{G}_2}(\mathbf{Y})$ have the same formula as Eq. (3), but are minimized over $\mathbf{X}, \mathbf{Y}$ respectively instead of $\tilde{\mathbf{X}}, \tilde{\mathbf{Y}}$. Minimizing $\mathcal{J}_{\mathcal{G}_1}(\mathbf{X})$, $\mathcal{J}_{\mathcal{G}_2}(\mathbf{Y})$ encourages the representations of the nearby nodes to be similar and the representations of disparate nodes to be dissimilar. Besides, to impose the consistency of the node representations between different networks, we minimize the distance between the node representations of one network and those computed by the aggregations from the other network via alignment. Specifically, given the node representations $\mathbf{x}_u$, $\forall u \in \mathcal{G}_1$ and $\mathbf{y}_v$, $\forall v \in \mathcal{G}_2$, we aim to minimize the following disagreement loss.



$\mathcal{G}_1$
$\mathcal{G}_2$
: Edges
: Intra-network aggregation
: Cross-network aggregation
u
v_{q_1}
v_{q_2}
$\mathbf{S}_1(u, v_{q_2})$
$\mathbf{S}_1(u, v_{q_1})$

Fig. 2: An illustrative example of the aggregations in the multi-GCN model ($K = 2$). Red solid lines represent the aggregations from node neighborhood within a single network in Intra-GCN while green dashed lines represent the cross-network aggregations in Inter-GCN.

$$\begin{aligned} \mathcal{J}_{\text{cross}}(\mathbf{X}, \mathbf{Y}) = &\sum_{u \in \mathcal{V}_1} \|\mathbf{x}_u - \sum_{k=1}^{K} \mathbf{S}_1(u, v_{q_k})\mathbf{y}_{v_{q_k}}\|_2^2 \\ &+ \sum_{v \in \mathcal{V}_2} \|\mathbf{y}_v - \sum_{k=1}^{K} \mathbf{S}_2(u_{p_k}, v)\mathbf{x}_{u_{p_k}}\|_2^2 \end{aligned} \tag{9}$$

Note that for $(u_{l_i}, v_{l_i}) \in \mathcal{L}^+$, since $\mathbf{S}_1(u_{l_i}, v_{l_i}) = 1$ is the only nonzero entry in the row of u_{l_i}, the first line in Eq. (9) is equivalent to $\|\mathbf{x}_{u_{l_i}} - \mathbf{y}_{v_{l_i}}\|_2^2$, enforcing the representation of node u_{l_i} in $\mathcal{G}_1$ to be close to that of its aligned node v_{l_i} in $\mathcal{G}_2$. However, one computational issue that resides in the Eq. (9) is that the node representations of all the other unlabeled nodes are nested with each other. For example, the first line needs the node representations $\mathbf{y}_{v_{q_k}}$, $\forall k = 1, \cdots, K$ which requires to explicitly calculate all node representations of $\mathcal{G}_2$ (i.e., $\mathbf{Y}$) in each iteration in the worst case. This is impractical especially when one wants to use stochastic training methods. Thus, we further simplify the terms in Eq. (9). For brevity, we only take $\sum_{k=1}^{K} \mathbf{S}_1(u, v_{q_k})\mathbf{y}_{v_{q_k}}$ as an example, and rewrite it as below.

$$\begin{aligned}
\sum_{k=1}^{K} \mathbf{S}_1(u, v_{q_k})\mathbf{y}_{v_{q_k}} &= \sum_{k=1}^{K} \mathbf{S}_1(u, v_{q_k}) \left([\tilde{\mathbf{y}}_{v_{q_k}} \| \hat{\mathbf{y}}_{v_{q_k}}]\mathbf{W}_{\text{cross}} + \mathbf{b}_2 \right) \\
&= \sum_{k=1}^{K} \sum_{k'=1}^{K} \mathbf{S}_1(u, v_{q_k})\mathbf{S}_2(u_{p_{k'}}, v_{q_k})\tilde{\mathbf{x}}_{u_{p_{k'}}}\mathbf{W}_{c_2} \\
&+ \sum_{k=1}^{K} \mathbf{S}_1(u, v_{q_k}) \left(\tilde{\mathbf{y}}_{v_{q_k}}\mathbf{W}_{c_1} + \mathbf{b}_2 \right) \\
&= \sum_{k=1}^{K} \mathbf{S}_1(u, v_{q_k})(\tilde{\mathbf{y}}_{v_{q_k}}\mathbf{W}_{c_1} + \mathbf{b}_2) + \sum_{k'=1} (\mathbf{S}_1\mathbf{S}_2^T)(u, u_{p_{k'}})\tilde{\mathbf{x}}_{u_{p_{k'}}}\mathbf{W}_{c_2} \\
&= \left[\sum_{k=1}^{K} \mathbf{S}_1(u, v_{q_k})\tilde{\mathbf{y}}_{v_{q_k}} \| \sum_{k'=1} (\mathbf{S}_1\mathbf{S}_2^T)(u, u_{p_{k'}})\tilde{\mathbf{x}}_{u_{p_{k'}}} \right] \mathbf{W}_{\text{cross}} + \mathbf{b}_2
\end{aligned}$$

where $\mathbf{W}_{c_1}$ and $\mathbf{W}_{c_2}$ are the first and second d rows of $\mathbf{W}_{\text{cross}}$ respectively. Based on the above equations, we can rethink of the computation for the aggregated representation of $\mathbf{x}_u$ (i.e., $\sum_{k=1}^{K} \mathbf{S}_1(u, v_{q_k})\mathbf{y}_{v_{q_k}}$) as two steps. First, we concatenate (1) the result of the cross-network aggregation $\hat{\mathbf{x}}_u$ and (2) the result (denoted by $\bar{\mathbf{x}}_u$) of the aggregation among the nodes $u_{p_{k'}}$ from the same network $\mathcal{G}_1$ weighted by $(\mathbf{S}_1\mathbf{S}_2^T)(u, u_{p_{k'}})$. Then, it is equivalent to

$$\sum_{k=1}^{K} \mathbf{S}_1(u, v_{q_k})\mathbf{y}_{v_{q_k}} = [\hat{\mathbf{x}}_u \| \bar{\mathbf{x}}_u]\mathbf{W}_{\text{cross}} + \mathbf{b}_2 \tag{10}$$

Nevertheless, as $\mathbf{S}_1\mathbf{S}_2^T$ is likely to be a dense matrix, the aggregations for $\bar{\mathbf{x}}_u$ gather the information globally from most of the other nodes, leading to a less emblematic $\bar{\mathbf{x}}_u$ and a higher computational cost. To address this issue, we only preserve the nonzero $(\mathbf{S}_1\mathbf{S}_2^T)(u, u_{p_{k'}})$, $\forall u_{p_{k'}} \in C_u$ (i.e., nodes that co-occur with node u in some random walks). In this way, the extra node representations that are needed only include $\tilde{\mathbf{x}}_{u_{p_{k'}}}$ which can be efficiently calculated by feeding node $u_{p_{k'}} \in C_u$ to the Intra-GCN$_1$ module. Accordingly, we can simplify the second term in Eq. (9) similarly.

B. Multi-View Point Set Alignment

After we obtain the node representations learned by the proposed Multi-GCN model, it seems that we can simply learn whether node u in $\mathcal{G}_1$ is aligned with node v in $\mathcal{G}_2$ based on their node representations $\mathbf{x}_u$ and $\mathbf{y}_v$. However, as the Multi-GCN model only preserves the structural consistency within the same networks (i.e., Eq. (3)) and the consistency between node representations and their aggregated representations from the other network (i.e., Eq. (9)), node representations $\mathbf{x}_u$ and $\mathbf{y}_v$ are still likely not to be close with each other in the Euclidean space even if node u and node v are supposed to be aligned. This limitation could result in a suboptimal alignment or even totally mislead the alignment. To mitigate this limitation, we propose to translate the network alignment problem over the nodes to a non-rigid point set alignment (PSA) problem where each point is represented by the representation of the corresponding node in the Euclidean space. Specifically, given a set of labeled node alignment $\mathcal{L}^+ = \{(u_{l_i}, v_{l_i})|i = 1, \cdots, L\}$, we want to displace each point $\mathbf{x}_{u_{l_i}}$ towards its aligned point $\mathbf{y}_{v_{l_i}}$ by some non-rigid vector-valued transformation function $\mathbf{f} \in \mathbb{R}^d$ such that the maximum point-to-point overlappings can be achieved. Mathematically, this can be formulated as a functional minimization problem.

$$\min_{\mathbf{f}} \sum_{i=1}^{L} \|\mathbf{x}_{u_{l_i}} + \frac{1}{2}\mathbf{f}(\mathbf{x}_{u_{l_i}}) - \mathbf{y}_{v_{l_i}}\|_2^2 \tag{11}$$

However, Eq. (11) is an ill-posed problem without any constraints imposed on $\mathbf{f}$. Instead, we model the above optimization problem by requiring the non-rigid function $\mathbf{f}$ to lie within a specific functional space, namely a reproducing kernel Hilbert space (RKHS) denoted by $\mathcal{H}$. Then we have

$$\min_{\mathbf{f}} \sum_{i=1}^{L} \|\mathbf{x}_{u_{l_i}} + \frac{1}{2}\mathbf{f}(\mathbf{x}_{u_{l_i}}) - \mathbf{y}_{v_{l_i}}\|_2^2 + \alpha\|\mathbf{f}\|_{\mathcal{H}}^2 \tag{12}$$

where $\|\mathbf{f}\|_{\mathcal{H}}^2$ is the RKHS norm of $\mathbf{f}$ in $\mathcal{H}$ and α is the regularization parameter. In addition, since each point (e.g., $\mathbf{x}_{u_{l_i}}$) intrinsically has two interpretations (or views): points in the Euclidean space (i.e., node representations) and the corresponding nodes of the networks in the non-Euclidean graph space, we further consider to divide $\mathcal{H}$ into two RKHS $\mathcal{H}^1, \mathcal{H}^2$ by $\mathcal{H} = \mathcal{H}^1 \oplus \mathcal{H}^2$ such that

$$\mathcal{H} = \left\{ \mathbf{f} | \mathbf{f}(\mathbf{x}) = \mathbf{f}^1(\mathbf{x}) + \mathbf{f}^2(\mathbf{x}), \mathbf{f}^1 \in \mathcal{H}^1, \mathbf{f}^2 \in \mathcal{H}^2 \right\}$$

and the RKHS norm $\|\mathbf{f}\|_{\mathcal{H}}^2$ can be re-written as

$$\|\mathbf{f}\|_{\mathcal{H}}^2 = \tag{13}$$

$$\min_{\substack{\mathbf{f}=\mathbf{f}^1+\mathbf{f}^2 \\ \mathbf{f}^1 \in \mathcal{H}^1 \\ \mathbf{f}^2 \in \mathcal{H}^2}} \alpha_1\|\mathbf{f}^1\|_{\mathcal{H}^1}^2 + \alpha_2\|\mathbf{f}^2\|_{\mathcal{H}^2}^2 + \mu \sum_{j=1}^{n_1-L} \left[\mathbf{f}^1(\mathbf{x}_{u_{r_j}}) - \mathbf{f}^2(\mathbf{x}_{u_{r_j}}) \right]^2$$

where $\mathbf{f}^1(\mathbf{x})$, $\mathbf{f}^2(\mathbf{x})$ are two transformation functions in the RKHS $\mathcal{H}^1, \mathcal{H}^2$ corresponding to the point view and graph view, respectively. Besides, $\mathcal{U} = \{u_{r_j}|j = 1, \cdots, n_1 - L\}$ represents the unlabeled nodes in $\mathcal{G}_1$ whose alignment with the nodes in $\mathcal{G}_2$ are not labeled. Intuitively, the term $\left[\mathbf{f}^1(\mathbf{x}_{u_{r_j}}) - \mathbf{f}^2(\mathbf{x}_{u_{r_j}}) \right]^2$ regularizes the transformation functions $\mathbf{f}^1, \mathbf{f}^2$ over the unlabeled node u_{r_j} to be consistent in two different views (i.e., moving $\mathbf{x}_{u_{r_j}}$ coherently in two views). According to [16], let $\mathcal{H}^1, \mathcal{H}^2$ be with the reproducing kernels κ^1, κ^2, then the RKHS $\mathcal{H}$ is with the reproducing kernel:

$$\kappa(u, u') = \phi(u, u') - \mu\mathbf{a}_u\mathbf{\Psi}\mathbf{a}_{u'}^T \tag{14}$$

Here, $\phi(u, u') = \alpha_1^{-1}\kappa^1(u, u') + \alpha_2^{-1}\kappa^2(u, u')$ and $\mathbf{a}_u = \alpha_1^{-1}\mathbf{k}_{u\mathcal{U}}^1 - \alpha_2^{-1}\mathbf{k}_{u\mathcal{U}}^2$ where $\mathbf{k}_{u\mathcal{U}}^s = [\kappa^s(u, u_{r_j}), u_{r_j} \in \mathcal{U}]$ for $s = 1, 2$ is a row vector measuring the kernel values between u and all the other unlabeled nodes in $\mathcal{U}$. Besides, $\mathbf{\Psi}$ is a positive definite matrix computed by $\mathbf{\Psi} = (\mathbf{I} + \mu\mathbf{\Phi})^{-1}$ where $\mathbf{\Phi}$ denotes the kernel matrix of $\phi(u, u')$ over all the unlabeled nodes. Furthermore, by the Representer Theorem [17], the solution to Eq. (13) is a function that

$$\mathbf{f}(\mathbf{x}_u) = \mathbf{K}(u, \mathcal{I})\mathbf{\Gamma} \tag{15}$$

where $\mathcal{I} = \{u_{l_i}|i = 1, \cdots, L\}$ includes the indices of the labeled nodes in $\mathcal{G}_1$, $\mathbf{K}$ represents the kernel matrix corre-

Algorithm 2 Non-rigid Network Alignment (ORIGIN).

```
Input: (1) undirected networks $\mathcal{G}_1 = \{\mathcal{V}_1, \mathbf{A}_1, \mathbf{X}^0\}$ and $\mathcal{G}_2 = \{\mathcal{V}_2, \mathbf{A}_2, \mathbf{Y}^0\}$, (2) a set of labeled cross-network node pairs $\mathcal{L}^+$ that are aligned, (3) the prior node similarity matrix $\mathbf{H}$, (4) the parameters $\alpha, \alpha_1, \alpha_2, \lambda, K, \rho$, (5) the total number of iterations $\text{iter}_{\text{max}}$.
Output: (1) the alignment matrix $\mathbf{S}$ between $\mathcal{G}_1, \mathcal{G}_2$, and (2) node representations $\mathbf{Z}, \mathbf{Y}$ of $\mathcal{G}_1, \mathcal{G}_2$.
1: Initialize the alignment matrix $\mathbf{S}$ by $\mathbf{H}$;
2: Compute the kernel matrix $\mathbf{K}^2 = \mathbf{I} + \mathbf{D}_1^{-\frac{1}{2}}\mathbf{A}_1\mathbf{D}_1^{-\frac{1}{2}}$;
3: for iter $= 1 \to \text{iter}_{\text{max}}$ do
4:     Compute $\mathbf{S}_1, \mathbf{S}_2$ by SAMPLE($\mathbf{S}$, $K$);
5:     Generate minibatches $\mathcal{B}^1 = \{\mathcal{B}_1^1, \cdots, \mathcal{B}_B^1\}, \mathcal{B}^2 = \{\mathcal{B}_1^2, \cdots, \mathcal{B}_B^2\}$ for $\mathcal{G}_1, \mathcal{G}_2$ by GraphSage models;
6:     for $b = 1 \to B$ do
7:         Generate $\tilde{\mathbf{x}}_u, \tilde{\mathbf{y}}_v$ for $u \in \mathcal{B}_b^1, v \in \mathcal{B}_b^2$ by GraphSage;
8:         Compute $\hat{\mathbf{x}}_u, \hat{\mathbf{y}}_v$ for $u \in \mathcal{B}_b^1, v \in \mathcal{B}_b^2$ by Eq. (6);
9:         Compute $\mathbf{x}_u, \mathbf{y}_v$ for $u \in \mathcal{B}_b^1, v \in \mathcal{B}_b^2$ by Eq. (7);
10:        Update hidden layer parameters by optimizing $\mathcal{J}_{\text{GCN}}$;
11:    end for
12:    Compute $\mathbf{X}, \mathbf{Y}$ by hidden layer parameters;
13:    Compute $\mathbf{K}^1$ by $\mathbf{K}^1(u, u') = \exp(-\|\mathbf{x}_u - \mathbf{x}_{u'}\|_2^2)$;
14:    Compute $\mathbf{K}$ by Eq. (14);
15:    Compute $\mathbf{\Gamma}$ by Eq. (19);
16:    Compute $\mathbf{Z} = \mathbf{X} + \frac{1}{2}\mathbf{K}(:, \mathcal{I})\mathbf{\Gamma}$ for $\mathcal{G}_1$;
17:    Update alignment $\mathbf{S}$ by $\mathbf{S}(u, v) = \exp(-\|\mathbf{z}_u - \mathbf{y}_v\|_2^2)$;
18:    Generate pseudo labels based on $\mathbf{S}$;
19:    Anneal $K \leftarrow K/\rho$;
20: end for
21: Output the alignment matrix $\mathbf{S}$;
22: Output node representations $\mathbf{Z}, \mathbf{Y}$ of $\mathcal{G}_1, \mathcal{G}_2$ respectively.
```

sponding to the kernel κ and $\mathbf{\Gamma}$ includes the coefficients to be solved. By substituting Eq. (15) to Eq. (12), we have

$$\min_{\mathbf{\Gamma}} \ \mathcal{J}_{\text{PSA}} = \sum_{i=1}^{L} \|\mathbf{x}_{u_{l_i}} + \frac{1}{2}\mathbf{K}(u_{l_i}, \mathcal{I})\mathbf{\Gamma} - \mathbf{y}_{v_{l_i}}\|_2^2 + \alpha \text{Tr}(\mathbf{\Gamma}^T \mathbf{K}_{\mathcal{I}} \mathbf{\Gamma}) \tag{16}$$

where $\mathbf{K}_{\mathcal{I}} = \mathbf{K}(\mathcal{I}, \mathcal{I})$.

To construct the kernel κ based on Eq. (14), we use the Gaussian RBF kernel $\kappa^1(u, u') = \exp(-\|\mathbf{x}_u - \mathbf{x}_{u'}\|_2^2)$ for the point view. In terms of the graph view, many existing kernels have been proposed, such as diffusion kernel [18], p-step random walk kernel [19]. In this paper, we choose the 1-step random walk kernel due to its computational simplicity. In particular, the kernel matrix corresponding to the kernel κ^2 is formulated as $\mathbf{K}^2 = 2\mathbf{I} - \tilde{\mathbf{L}} = \mathbf{I} + \mathbf{D}_1^{-\frac{1}{2}}\mathbf{A}_1\mathbf{D}_1^{-\frac{1}{2}}$ where $\mathbf{D}_1$ is the diagonal degree matrix of $\mathbf{A}_1$. By optimizing Eq. (16) , we can solve for $\mathbf{\Gamma}$ and compute the transformed node representations of $\mathcal{G}_1$ by $\mathbf{z}_u = \mathbf{x}_u + \frac{1}{2}\mathbf{K}(u, \mathcal{I})\mathbf{\Gamma}$. Such transformed node representations will then be compared with $\mathbf{Y}$ to infer the alignment with the nodes in $\mathcal{G}_2$. Specifically, we calculate the cross-network node similarity between node u in $\mathcal{G}_1$ and node v in $\mathcal{G}_2$ as the alignment matrix by $\mathbf{S}(u, v) = \exp(-\|\mathbf{z}_u - \mathbf{y}_v\|_2^2)$, which measures the confidence of aligning the two nodes.

C. Optimization Algorithm

The overall loss function of the proposed model is

$$\mathcal{J} = \underbrace{\mathcal{J}_{\mathcal{G}_1}(\mathbf{X}) + \mathcal{J}_{\mathcal{G}_2}(\mathbf{Y}) + \lambda \mathcal{J}_{\text{cross}}(\mathbf{X}, \mathbf{Y})}_{\text{Multi-GCN}} + \underbrace{\mathcal{J}_{\text{PSA}}}_{\text{point set alignment}} \tag{17}$$

To minimize the above loss function, it is straightforward to simultaneously learn all the parameters in an alternating manner. However, the main drawbacks of this approach include: (1) if the node representations of $\mathcal{G}_1$ and $\mathcal{G}_2$ are not representative enough, the inferred alignment could be suboptimal or even misleading, and (2) at the initial stages, as the alignment matrix $\mathbf{S}$ is expected to be imprecise, the learning of node representations is very likely to be misled (e.g., in Eq. (6)). Even worse, these drawbacks could make the learning algorithm diverge.

Instead, we propose the optimization algorithm, where each training cycle is divided into two stages. In the first stage, we train the proposed Multi-GCN model to learn node representations with the current alignment matrix $\mathbf{S}$. In this paper, we use the GraphSage models with Mean aggregators [14] for Intra-GCN$_1$ and Intra-GCN$_2$ and hence the Multi-GCN model can be optimized by mini-batched stochastic gradient descent (SGD). We note that other GCN models can also be also used as the Intra-GCNs (e.g., [20]). In the second stage, we learn the parameter $\mathbf{\Gamma}$ by solving the optimization problem Eq. (16). Specifically, we compute the gradient of Eq. (16) w.r.t. $\mathbf{\Gamma}$ as

$$\frac{\partial \mathcal{J}_{\text{PSA}}}{\partial \mathbf{\Gamma}} = \frac{1}{2}\mathbf{K}_{\mathcal{I}}^T\mathbf{K}_{\mathcal{I}}\mathbf{\Gamma} + \mathbf{K}_{\mathcal{I}}^T(\mathbf{X}(\mathcal{I}, :) - \mathbf{Y}(\mathcal{I}, :)) + 2\alpha\mathbf{K}_{\mathcal{I}}\mathbf{\Gamma} \tag{18}$$

Then we can obtain the solution of $\mathbf{\Gamma}$ by gradient descent.

$$\mathbf{\Gamma} = \mathbf{\Gamma} - \eta \frac{\partial \mathcal{J}_{\text{PSA}}}{\partial \mathbf{\Gamma}} \tag{19}$$

where η is the learning rate. Note that the time complexity of Eq. (19) in each iteration is $O(|\mathcal{L}^+|^2 d)$ which is sub-quadratic w.r.t. the number of nodes n. After that, we calculate the displaced node representations of $\mathcal{G}_1$ by $\mathbf{Z} = \mathbf{X} + \frac{1}{2}\mathbf{K}(:, \mathcal{I})\mathbf{\Gamma}$, followed by computing the cross-network node similarity matrix as the alignment matrix $\mathbf{S}$. Next, the alignment matrix $\mathbf{S}$ will be fed back to Multi-GCN after the pseudo-labeling and sampling steps. To generate the pseudo-labels indicating which node in $\mathcal{G}_1$ is aligned with which node in $\mathcal{G}_2$, we first conduct a greedy matching process on $\mathbf{S}$ to obtain all one-to-one node alignment [12], and then leverage the alignment consistency proposed in [7] to select the confident alignment as the pseudo labels. To be specific, note that the alignment consistency assumes if two nodes are aligned, their corresponding close neighbors are likely to be aligned. In this way, we heuristically select the alignment between node u and node v obtained by greedy matching as the pseudo alignment label, if nodes (u, v) are the respective neighbors of nodes $(u_{l_i}, v_{l_i}) \in \mathcal{L}^+$. The overall optimization algorithm is summarized in Algorithm 2. Note that as the model is trained, matrix $\mathbf{S}$ should indicate more accurate node alignment. For this reason, we reduce the sample size K used in Algorithm 1 by ρ (Line 19).

TABLE II: Statistics of the networks.

Category	Network	# of Nodes	# of Edges	# of Attributes
Citation	Cora	2,708	5,429	1,433
Citation	Citeseer	3,327	4,732	3,703
Social	Foursquare	5,313	54,233	5
Social	Twitter	5,120	130,575	5

IV. Experiment

In this section, we present experimental results of the proposed model ORIGIN. We evaluate in the following aspects:

- *Effectiveness*: How accurate is our algorithm to align networks and how robust is our algorithm to the parameters?
- *Efficiency*: How fast is our algorithm?

A. Experimental Setup

Datasets. We evaluate the proposed model on four real-world networks. The statistics of all datasets are summarized in Table II.

- *Citation networks*: We consider two citation networks: Cora and Citeseer[2]. Each node represents a document and each edge indicates a citation link between two documents. We use the bag-of-words representations of the documents as the node attributes. For both networks, we convert the originally directional edges to undirected to make the networks undirected.
- *Social networks*: We consider two social networks including Foursquare and Twitter [21]. Each node in the networks represents a user and each edge represents the friendship between two users. For each node in the networks, we compute the degree, the number of edges in the egonet, PageRank score, betweenness and closeness centralities and use them as node attributes. We then convert them to undirected networks.

We build the following three scenarios to evaluate the alignment performance. In each scenario, we randomly select 50%, 30% and 20% cross-network node pairs from the ground-truth as labeled alignment $\mathcal{L}^+$.

- *Cora-1 vs. Cora-2*. Given the Cora network (denoted by $\mathcal{G}_1 = \{\mathcal{V}_1, \mathbf{A}_1, \mathbf{X}^0\}$), we first generate a random permutation matrix $\mathbf{P}$ which is used as the ground-truth alignment. We treat the permutated matrix $\mathbf{A}_2 = \mathbf{P}^T\mathbf{A}_1\mathbf{P}$ and $\mathbf{Y}^0 = \mathbf{P}^T\mathbf{X}^0$ as the adjacency matrix and node attributes of the second network $\mathcal{G}_2$. We also randomly remove 5%, 15% edges from two networks and add 10%, 15% noise on the node attributes of two networks, respectively. We compute node degree similarity for $\mathbf{H}$.
- *Citeseer-1 vs. Citeseer-2*. This scenario is built similarly.
- *Foursquare vs. Twitter*. In this scenario, we aim to align nodes in Foursquare and Twitter networks. There are 1,609 common nodes between two networks which are used as the ground-truth to evaluate the alignment. Besides, we compute the degree similarity matrix as $\mathbf{H}$.

Baseline Methods. We compare our method ORIGIN with the following network alignment algorithms, including (1) *SageAlign* that learns node representations by two separate GraphSage models [14], followed by the proposed point set alignment algorithm, (2) *FINAL-N* [7], [22], (3) *FINAL-P* which is a non-attributed variant of *FINAL-N*, (4) *REGAL* [23], (5) *IONE* [8], and (6) *PriorSim* which aligns the nodes directly based on the prior node similarity matrix $\mathbf{H}$. To make a fair comparisons, we set $\mathbf{H}(u_{l_i},:) = \mathbf{H}(:,v_{l_i}) = 0$ except that $\mathbf{H}(u_{l_i},v_{l_i}) = 1,\ \forall i = 1,\cdots,L$ for *FINAL-N, FINAL-P* and *PriorSim* to incorporate the label information. For *REGAL*, we slightly modify the original released code by setting the constructed node similarity matrix to be factorized by $\mathbf{H}_{\text{REGAL}}(u_{l_i},v_{l_i}) = 1,\ \forall i = 1,\cdots,L$.

Hyperparameters. For all the effectiveness evaluations, we set $\lambda = 0.1$, $K = 20$, $\alpha = 0.1$, $\alpha_1 = 0.01$, $\alpha_2 = 1$, $\rho = 1.2$. We use 0.001 as the learning rate. We keep the default values for all the hyperparameters used in the baseline methods.

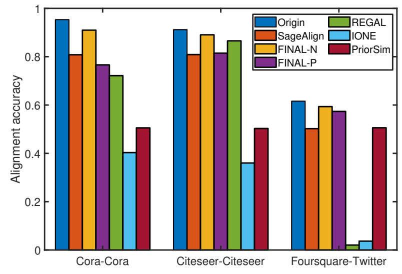
Machines. The proposed method ORIGIN is implemented in Tensorflow [24] with Adam optimizer. We use one Nvidia Titan X with 12G RAM as GPU. The CPU-based methods are performed with four 3.6GHz Intel Cores and 32G RAM.

B. Effectiveness Results

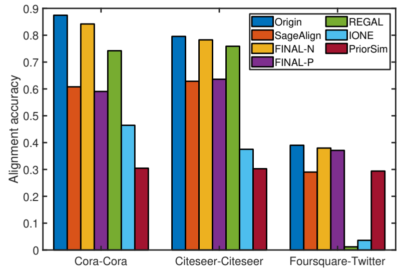
We first evaluate the alignment accuracy of the proposed ORIGIN compared with the baseline methods. To compute the alignment accuracy, we conduct a greedy matching [12] as the post-processing step to obtain the one-to-one node mapping between two networks, followed by calculating the percentage of *all* ground-truths that can be correctly aligned as the alignment accuracy. The results are summarized in Figure 3. We have the following observations. First, our proposed method ORIGIN outperforms all the baseline methods. Specifically, our method can achieve an up to 5% improvement in terms of the alignment accuracy compared with *FINAL-N*, a strong baseline for attributed network alignment. Recall that *FINAL-N* can be viewed as a method based on the linear transformation to maximize a variant of Koopmans-Beckmann's QAP. This demonstrates the benefits of the non-rigid network alignment. Besides, our method can achieve better alignment results than other node representation-based methods, namely *REGAL* and *IONE*. Second, our method achieves an at least 15% accuracy improvement compared with *SageAlign*, a variant of ORIGIN. This shows that the node representations of multiple networks jointly learned with the proposed Multi-GCN are more powerful for the specific alignment task. In the meanwhile, *SageAlign* itself also demonstrates the effectiveness of the proposed point set alignment algorithm to align node representations in the Euclidean space. Finally, as the amount of labeled alignment decreases, our method consistently outperforms other baseline methods.

To further verify if our method can correctly align unlabeled nodes, we compare the precision@30 score with the baseline methods. We define the precision@30 as follows. For any unlabeled node u in $\mathcal{G}_1$, if the correct alignment (say node v in $\mathcal{G}_2$) belongs to the top-30 most similar nodes to node u, we say there is a *hit*. We calculate the precision@30 by precision@30=(# of hits)/(# of unlabeled nodes). As Figure 4 shows, our method achieves higher precision@30 scores than

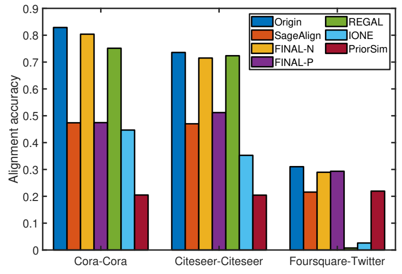
[2]https://linqs.soe.ucsc.edu/data



(a) 50% labeled alignment.

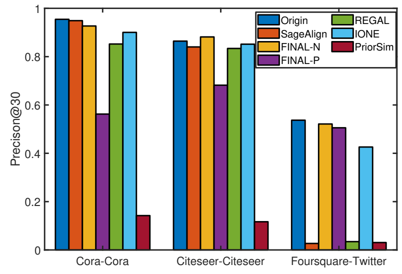


(b) 30% labeled alignment.

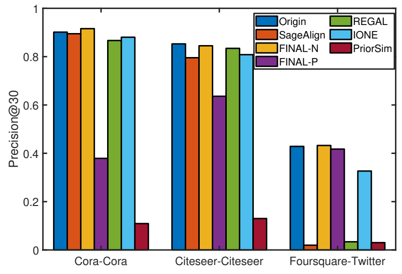


(c) 20% labeled alignment.

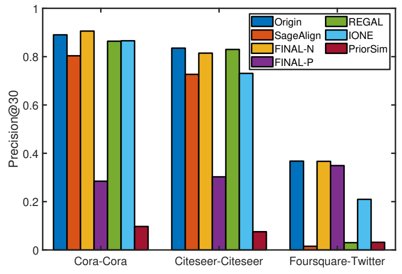
Fig. 3: Alignment accuracy with different amount of labeled alignment. (Best viewed in color.)



(a) 50% labeled alignment.



(b) 30% labeled alignment.



(c) 20% labeled alignment.

Fig. 4: Precision@30 with different amount of labeled alignment. (Best viewed in color.)

baseline methods in most cases (i.e., our method is slightly lower than *FINAL-N* only in a few cases).

We also visualize in Figure 5 the node representations of networks cora-1 and cora-2 in 2-D space by t-SNE and demonstrate the benefits of point set alignment. Here, we use different colors as different node classes, which are known a priori as additional information, to help indicate the node correspondence. For example, nodes in purple in network cora-1 are expected to be aligned with the purple nodes in network cora-2. By comparing Figure 5 (a) (i.e., $\mathbf{X}$) and Figure 5 (c) (i.e., $\mathbf{Y}$), we observe that despite minimizing the distances among node representations across networks by Eq. (9), nodes that ought to be aligned may still be far away from each other in the Euclidean space, such as the nodes in purple of two networks. This incomparability among node representations could further mislead the node alignment. In contrast, by our proposed non-rigid point set alignment, node representations of network cora-1 can be moved towards those of network cora-2. Consequently, the displaced node representations $\mathbf{Z}$ shown in Figure 5 (b) are closer to the representations $\mathbf{Y}$ of their aligned counterparts in network cora-2.

Moreover, we conduct a parameter study on cora-1 and cora-2 dataset about how the importance of different views used for point set alignment (i.e., α_1, α_2) influences the alignment accuracy. As Figure 6 shows, the alignment accuracy is stable over a wide range of α_1, α_2 ($0.01 \leq \alpha_1, \alpha_2 \leq 10$). Meanwhile, we observe that when $\alpha_2 > \alpha_1$, the proposed method achieves higher alignment accuracies than those when $\alpha_2 \leq \alpha_1$. This indicates the graph view indeed benefits point set alignment by calibrating the alignment based on a single point view.

C. Efficiency Results

We evaluate the efficiency of our proposed Multi-GCN for three alignment scenarios in terms of the running time per mini-batch. We fix each mini-batch with size 200 and compare our method (Multi-GCN) with its single-network counterpart (GraphSage). In particular, we measure the time cost of Multi-GCN minimizing Eq. (8) and the cost of GraphSage on two networks minimizing the loss Eq. (3). As shown in Figure 7, the time cost of the proposed Multi-GCN is very close to the GraphSage counterpart. This suggests that the extra computational cost for Inter-GCN, which brings the alignment improvement as shown above, is actually quite light.

V. Related Work

Here, we briefly review the related works on network alignment, node representation learning and point set alignment.

A. Network Alignment

Many traditional graph matching based methods formulate the problem into a Koopmans-Beckmann's quadratic assignment problem (KBQAP) [3]. However, it is very hard to directly solve KBQAP in an exact way. The early attempts [4], [25] to approximate the problem propose to use the spectral relaxation, i.e., relaxing the assignment matrix to an orthogonal matrix. Bayati et. al formulate the network alignment problem by relaxing the assignment matrix in KBQAP to a doubly stochastic matrix [6]. Recently, Zhang et. al adopt the attribute consistency to calibrate the structure-based alignment and formulate the attributed network alignment problem as a convex quadratic problem [7]. Note that its formulation can be also derived into a relaxed KBQAP. These methods can be viewed as using linear transformations to represent the networks and then aligning those node representations in the Euclidean space. However, the linear transformation assumption might oversimplify the complicated alignment relationship. Other existing network alignment methods include multilevel network alignment [26], high-order network alignment [27] and incomplete network alignment [28].

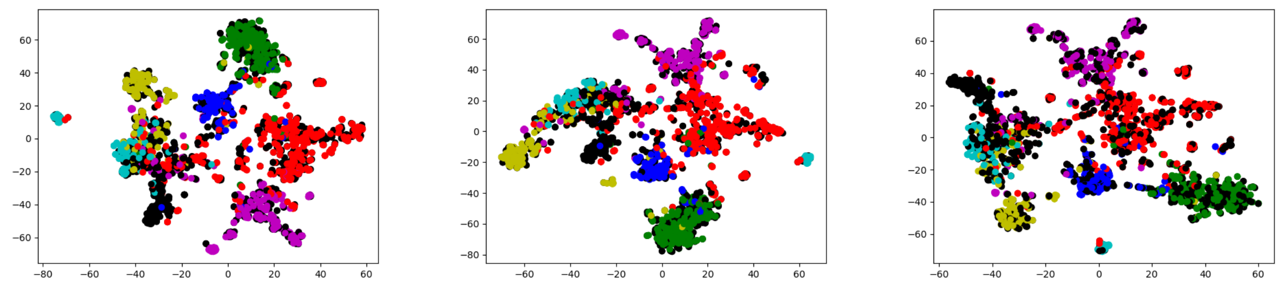

(a) Cora-1 node representations (i.e. $\mathbf{X}$). (b) Displaced cora-1 representations (i.e. $\mathbf{Z}$). (c) Cora-2 node representations (i.e. $\mathbf{Y}$).

Fig. 5: 2-D t-SNE visualization of node representations of networks cora-1 and cora-2. (Best viewed in color.)

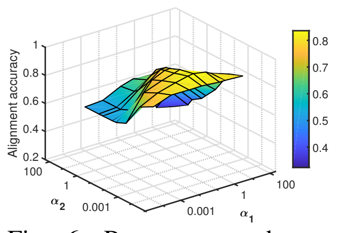


Fig. 6: Parameter study on α_1, α_2

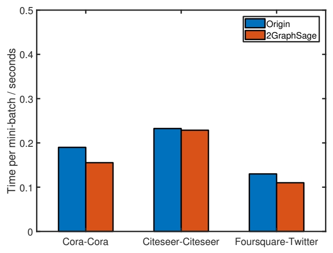


Fig. 7: Running time per mini-batch with size of 200.

Thanks to the recent advances in the network representation learning, several works are proposed to solve the alignment problem in the Euclidean space instead of directly aligning the networks in the non-Euclidean space. Among others, [8], [9] propose network embedding based alignment methods that simultaneously preserve the node proximity within a single network and require the aligned nodes to coincide in the embedding space. These methods typically only leverage the structural information of the input networks. To encode node attributes, Heimann et. al propose a matrix factorization method over some carefully constructed cross-network node similarity matrix [23]. However, most of these methods enforce the node representations of the known aligned node pairs to be close with each other, yet overlook the fact that the node representations of other unlabeled node pairs might still be incomparable due to the imperfect rotations.

B. Node Representation Learning

Following the idea of representation learning, especially word embedding [15], many network embedding methods have been proposed over years. To name a few, *DeepWalk* conducts the fixed-length random walks to build the contexts of each node and then applies the SkipGram model to learn the node embedding [29]. *Node2vec* [30] proposes a combination of BFS and DFS to construct node contexts. Similar works include [31], [32], [33]. Recently, many graph neural network models have been proposed to learn node representations by aggregating node information from the neighborhood. In general, based on the types of aggregators, graph neural networks can be categorized into graph convolutional networks [13], [14], graph attention networks [34], [35], gated graph neural networks [36] and jumping knowledge networks [20]. Among others, graph convolutional networks (GCN) mainly have two categories: spectral-based GCN (e.g., [13]) and spatial-based GCN (e.g., [14]). For more details of graph neural networks, one can refer to [37], [38]. Most of them only aggregate node information within a single network, and thus might miss the potential benefits of the auxiliary node information.

C. Point Set Alignment

Point set alignment aims to estimate the correspondences between two point sets in the Euclidean space and has been widely applied in computer vision and pattern recognition. Generally speaking, point set alignment can be categorized into rigid and and non-rigid transformation based methods. Rigid transformations include the linear transformation and affine transformations. One of the well-established rigid alignment method is the Iterative Closest Point (*ICP*) [39] which iteratively uses the nearest neighbors to estimate the rigid transformation. For non-rigid transformations, Myronenko et. al propose to view one point set as the centroids of a Gaussian mixture model and the other as the data [10]. Moreover, [40] encodes the local contexts within the point sets into the alignment procedure. Nevertheless, it is still nascent to align point sets representing networks as the node representations of networks may not have a good geometric property.

VI. Conclusion

The multi-sourced real-world networks from numerous domains have galvanized the research in network alignment. Most of the existing graph matching based methods explicitly or implicitly consider the alignment matrix as a linear transformation matrix, whereas the node representation based methods are limited by the incomparability issues among the node representations. In this paper, we study the non-rigid network alignment problem and propose a semi-supervised deep model ORIGIN which jointly learns the *far-reaching* node representations and finds node alignment across multiple networks. We first propose a Multi-GCN model that can aggregate the auxiliary node information across different networks to aid node representation learning. We translate the network alignment problem to the point set alignment problem and propose a non-rigid alignment method based on multi-view learning. We perform extensive experiments that demonstrate the effectiveness of the proposed ORIGIN in finding accurate alignment and efficiency in node representation learning.

VII. Acknowledgement

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

References

[1] R. Trivedi, B. Sisman, J. Ma, C. Faloutsos, H. Zha, and X. L. Dong, "Linknbed: Multi-graph representation learning with entity linkage," *arXiv preprint arXiv:1807.08447*, 2018.

[2] J. Xu, H. Tong, T.-C. Lu, J. He, and N. Bliss, "Gta 3 2018: Workshop on graph techniques for adversarial activity analytics," in *Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining*. ACM, 2018, pp. 803–803.

[3] T. C. Koopmans and M. Beckmann, "Assignment problems and the location of economic activities," *Econometrica: journal of the Econometric Society*, pp. 53–76, 1957.

[4] S. Umeyama, "An eigendecomposition approach to weighted graph matching problems," *IEEE transactions on pattern analysis and machine intelligence*, vol. 10, no. 5, pp. 695–703, 1988.

[5] D. Koutra, H. Tong, and D. Lubensky, "Big-align: Fast bipartite graph alignment," in *ICDM*. IEEE, 2013.

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

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

[8] L. Liu, W. K. Cheung, X. Li, and L. Liao, "Aligning users across social networks using network embedding." in *IJCAI*, 2016.

[9] T. Man, H. Shen, S. Liu, X. Jin, and X. Cheng, "Predict anchor links across social networks via an embedding approach." in *IJCAI*, 2016.

[10] A. Myronenko and X. Song, "Point set registration: Coherent point drift," *IEEE transactions on pattern analysis and machine intelligence*, 2010.

[11] D. I. Shuman, S. K. Narang, P. Frossard, A. Ortega, and P. Vandergheynst, "The emerging field of signal processing on graphs: Extending high-dimensional data analysis to networks and other irregular domains," *IEEE Signal Processing Magazine*, vol. 30, no. 3, pp. 83–98, 2013.

[12] R. Singh, J. Xu, and B. Berger, "Global alignment of multiple protein interaction networks with application to functional orthology detection," *Proceedings of the National Academy of Sciences*, 2008.

[13] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," *arXiv preprint arXiv:1609.02907*, 2016.

[14] W. Hamilton, Z. Ying, and J. Leskovec, "Inductive representation learning on large graphs," in *Advances in Neural Information Processing Systems*, 2017, pp. 1024–1034.

[15] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," in *Advances in neural information processing systems*, 2013, pp. 3111–3119.

[16] V. Sindhwani and D. S. Rosenberg, "An rkhs for multi-view learning and manifold co-regularization," in *Proceedings of the 25th international conference on Machine learning*. ACM, 2008, pp. 976–983.

[17] B. Schölkopf, R. Herbrich, and A. J. Smola, "A generalized representer theorem," in *International conference on computational learning theory*. Springer, 2001, pp. 416–426.

[18] R. I. Kondor and J. Lafferty, "Diffusion kernels on graphs and other discrete structures," in *Proceedings of the 19th international conference on machine learning*, vol. 2002, 2002, pp. 315–322.

[19] A. J. Smola and R. Kondor, "Kernels and regularization on graphs," in *Learning theory and kernel machines*. Springer, 2003, pp. 144–158.

[20] K. Xu, C. Li, Y. Tian, T. Sonobe, K.-i. Kawarabayashi, and S. Jegelka, "Representation learning on graphs with jumping knowledge networks," *arXiv preprint arXiv:1806.03536*, 2018.

[21] J. Zhang and S. Y. Philip, "Integrated anchor and social link predictions across social networks," in *Twenty-Fourth International Joint Conference on Artificial Intelligence*, 2015.

[22] S. Zhang and H. Tong, "Attributed network alignment: Problem definitions and fast solutions," *IEEE Transactions on Knowledge and Data Engineering*, 2018.

[23] M. Heimann, H. Shen, T. Safavi, and D. Koutra, "Regal: Representation learning-based graph alignment," in *CIKM*. ACM, 2018.

[24] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard *et al.*, "Tensorflow: A system for large-scale machine learning," in *12th {USENIX} Symposium on Operating Systems Design and Implementation ({OSDI} 16)*, 2016, pp. 265–283.

[25] C. Ding, T. Li, and M. I. Jordan, "Nonnegative matrix factorization for combinatorial optimization: Spectral clustering, graph matching, and clique finding," in *2008 Eighth IEEE International Conference on Data Mining*. IEEE, 2008, pp. 183–192.

[26] S. Zhang, H. Tong, R. Maciejewski, and T. Eliassi-Rad, "Multilevel network alignment," in *The World Wide Web Conference*. ACM, 2019, pp. 2344–2354.

[27] S. Mohammadi, D. F. Gleich, T. G. Kolda, and A. Grama, "Triangular alignment tame: A tensor-based approach for higher-order network alignment," *IEEE/ACM Transactions on Computational Biology and Bioinformatics (TCBB)*, vol. 14, no. 6, pp. 1446–1458, 2017.

[28] S. Zhang, H. Tong, J. Tang, J. Xu, and W. Fan, "ineat: Incomplete network alignment," in *2017 IEEE International Conference on Data Mining (ICDM)*. IEEE, 2017, pp. 1189–1194.

[29] B. Perozzi, R. Al-Rfou, and S. Skiena, "Deepwalk: Online learning of social representations," in *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2014, pp. 701–710.

[30] A. Grover and J. Leskovec, "node2vec: Scalable feature learning for networks," in *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2016.

[31] J. Tang, M. Qu, M. Wang, M. Zhang, J. Yan, and Q. Mei, "Line: Large-scale information network embedding," in *Proceedings of the 24th international conference on world wide web*. International World Wide Web Conferences Steering Committee, 2015, pp. 1067–1077.

[32] L. F. Ribeiro, P. H. Saverese, and D. R. Figueiredo, "struc2vec: Learning node representations from structural identity," in *KDD*. ACM, 2017.

[33] D. Zhou, J. He, H. Yang, and W. Fan, "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*. ACM, 2018, pp. 2807–2816.

[34] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Lio, and Y. Bengio, "Graph attention networks," *arXiv preprint arXiv:1710.10903*, 2017.

[35] Z. Liu, D. Zhou, and J. He, "Towards explainable representation of time-evolving graphs via spatial-temporal graph attention," in *The 28th ACM International Conference on Information and Knowledge Management*, 2019.

[36] N. Peng, H. Poon, C. Quirk, K. Toutanova, and W.-t. Yih, "Cross-sentence n-ary relation extraction with graph lstms," *arXiv preprint arXiv:1708.03743*, 2017.

[37] J. Zhou, G. Cui, Z. Zhang, C. Yang, Z. Liu, and M. Sun, "Graph neural networks: A review of methods and applications," *arXiv preprint arXiv:1812.08434*, 2018.

[38] S. Zhang, H. Tong, J. Xu, and R. Maciejewski, "Graph convolutional networks: Algorithms, applications and open challenges," in *International Conference on Computational Social Networks*. Springer, 2018, pp. 79–91.

[39] P. J. Besl and N. D. McKay, "Method for registration of 3-d shapes," in *Sensor Fusion IV: Control Paradigms and Data Structures*, vol. 1611. International Society for Optics and Photonics, 1992, pp. 586–607.

[40] G. Wang, Z. Wang, Y. Chen, Q. Zhou, and W. Zhao, "Context-aware gaussian fields for non-rigid point set registration," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 5811–5819.