Enhancing Network Alignment through Multi-Scale Information Fusion

Zhihao Chen[1], Pengfei Jiao[1], Yinghui Wang[2], Huijun Tang[1*], Jilin Zhang[1]
[1]Hangzhou Dianzi University, Hangzhou, China
[2]Tianjin University, Tianjin, China
{zhchen, pjiao, tanghuijune, jilin.zhang}@hdu.edu.cn, wangyinghui@tju.edu.cn

***Abstract*—The network alignment task aims to identify the correspondence of the same nodes across networks, which plays a crucial role in many fields. However, most existing methods focus solely on consistency assumptions at the local structure information of nodes, meaning that the alignment of neighboring node pairs tends to be consistent with each other, while ignoring information from other scales, such as higher-order information. This can easily lead to over-smoothing issues. In this paper, we propose a novel multi-scale embedding-based network alignment method in this paper. Specifically, we capture three types of node embedding information, including first-order embedding, multi-order embedding, and higher-order embedding, to obtain rich information. Then, we enhance the multi-order and higher-order embedding information of nodes by applying a stable-pair based fine-tuning method. Finally, we use a fusion mechanism based on importance weights to combine the alignment matrices obtained from these three types of node embedding information and obtain the optimal alignment matrix. We conduct extensive experiments on three real-world datasets to verify the effectiveness of our proposed model.**

***Index Terms*—network alignment, multi-scale, node embedding**

I. Introduction

In the era of information explosion, information in many fields can be expressed through network structures, such as social networks of different online platforms and academic citation networks. Some nodes exist in different social networks at the same time. Generally, such nodes are called anchor nodes, and the process of finding the corresponding anchor node relationship (*i.e.,* anchor link [1], [2]) is called network alignment. In real-world life, network alignment plays an important role in multiple networks. For example, when a person is active on multiple online social networks, network alignment can connect users from different platforms. This allows us to understand their social characteristics and interests, enabling cross-domain recommendations [3].

In recent years, there are many solutions for network alignment. Network alignment methods are mainly divided into structured-based methods [4]–[6] and embedding-based methods [7]–[9]. The previous structured-based methods are mostly based on the consistency assumptions [6], [10] that anchor nodes of different networks should have a consistent connection structure. But most methods only consider the neighbor nodes directly connected to the anchor nodes.

* Corresponding Author

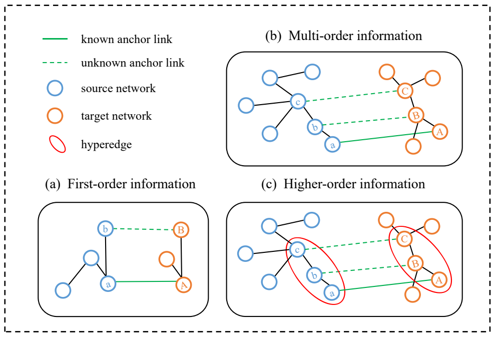
known anchor link
unknown anchor link
source network
target network
hyperedge
(b) Multi-order information
(a) First-order information
(c) Higher-order information

Fig. 1. Examples of our motivation.

Directly optimizing this alignment consistency might lead to local over-smoothing problems. Inspired by the recent network embedding-based methods, some methods use network embedding technology to learn the node information and use the known anchor node as the supervisory information to learn the corresponding relationship between nodes. For example, DANA [7] uses Generative Adversarial Networks to capture node information in the two networks, respectively and learns a mapping function to perform the nearest neighbor node alignment.

Although existing methods have achieved good results, they still suffer from several problems. Firstly, most current methods solely focus on information from a single scale of nodes and conduct network alignment by enhancing the similarity of anchor nodes [8], [11], while disregarding information from other scales, such as higher-order information. Secondly, most network alignment methods are based on the principle of consistency assumptions [6], [12], which requires the structure of overlapping parts between two graphs to be similar. However, in real-world scenarios, the structure between different graphs may not always be consistent. Some anchor nodes adhere to the principle of consistency, while others deviate from it. Consequently, this method can easily lead to over-smoothing issues [13]. Finally, the majority of potential anchor nodes are dispersed in the network as higher-order neighbors of known anchor nodes [14], posing a challenge in obtaining identifiable information about the topological structure of distant known anchor nodes.

To solve the above challenges, in this paper, we propose a novel multi-scale embedding-based network alignment method. Our main idea is to capture three types of node embedding information: first-order information, multi-order information, and higher-order information. As shown in Fig. 1(a), the anchor nodes a and A have potential anchor nodes b and B as their first-order neighbors. Moreover, the local structures of potential anchor nodes in different networks exhibit similarities. Therefore, first-order information aids in aligning potential anchor nodes that adhere to the principle of consistency assumptions. For the learning of first-order embedding information, we refer to the skip-gram [15], [16] model idea. Multi-order information is used to capture network semantic information. As shown in Fig. 1(b), there exists a disparity in the local structure of potential anchor nodes c and C. To address this issue, we integrated both the low-order and higher-order information of nodes to subtract the interference caused by network semantics. This process enables us to obtain globally invariant features of nodes that are independent of the network. Ultimately, this approach assists in aligning potential anchor nodes that deviate from consistency assumptions. For the learning of multi-order embedding information, we use the GCN [17] model to learn, which is beneficial for nodes to capture the multi-order structure and feature information.

Hypergraphs [18] can model complex higher-order relationships, *i.e.*, hyperedges can connect more than two nodes, which is difficult to achieve in simple graphs. For example, the constructed hypergraph based on the n-hop neighbor method represents the circle of friends to some extent. As depicted in Fig. 1(c), the anchor node a in the source network and the potential anchor nodes b and c are located within the same hyperedge. Similarly, in the target network, the anchor node A and the potential anchor nodes B and C also reside in the same hyperedge. Potential anchor nodes c and C can capture information from remote anchor nodes a and A. By leveraging the hypergraph autoencoder [19], [20], the node information is enriched with hyperedge information, enabling effective alignment of potential anchor nodes that deviate from the consistency assumptions.

Then, in order to align potential anchor nodes that do not comply with the principle of consistency assumption, we enhance the multi-order and higher-order embedding information of nodes by applying stable-pair based fine-tuning method. Finally, we use a fusion mechanism based on importance weights to fuse the alignment matrices obtained from three types of node embedding information to obtain the optimal alignment matrix. We summarize the contributions of this paper as follows:

- we propose a novel multi-scale embedding-based network alignment method that integrates first-order, multi-order, and higher-order rich network information, allowing for the full utilization of network structure and node feature information.
- We propose a stabled-pair based fine-tuning method, where we identify stable pairs in both multi-order and higher-order node embeddings, and subsequently increase the weights of these stable pairs to enhance network alignment effectiveness.
- We have conducted extensive experimental verification on three real-world datasets to demonstrate the effectiveness of our proposed model in network alignment tasks.

II. Preliminaries

We define the symbols lowercase a, bold lowercase $\mathbf{a}$, and bold uppercase $\mathbf{A}$ to represent scale, vector, and matrix, respectively. A network is represented as $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V}$ is the set of nodes and $\mathcal{E}$ is the set of edges. In this paper, we mainly focus on the alignment between the two types of graphs, *i.e.*, simple graph and hypergraph. The source network is $\mathcal{G}^s_{simple}$ or $\mathcal{G}^s_{hyper}$ and the target network is $\mathcal{G}^t_{simple}$ or $\mathcal{G}^t_{hyper}$, superscripts s and t refer to symbols related to the source network and the target network, respectively. superscripts $simple$ and $hyper$ refer to symbols related to the simple graph and the hypergraph, respectively. To ensure flexibility, we will omit the symbols s, t, $simple$, and $hyper$ as required.

A. Problem Definition

Definition 1 (Network Alignment): Given a source network $\mathcal{G}^s = (\mathcal{V}^s, \mathcal{E}^s)$ and a target network $\mathcal{G}^t = (\mathcal{V}^t, \mathcal{E}^s)$, a few partially known anchor links $\mathcal{A}^{s,t} \subseteq \mathcal{V}^s \times \mathcal{V}^t$. The purpose of the network alignment problem is to find the remaining unknown anchor links $\mathcal{B}^{s,t} = \{(v^s_i, v^t_j) | v^s_i \in \mathcal{V}^s, v^t_j \in \mathcal{V}^t, (v^s_i, v^t_j) \notin \mathcal{A}^{s,t}\}$ between different networks, in which v^s_i and v^t_j represent the same natural entity in the real world.

Definition 2 (Hypergraph): A network is represented as $\mathcal{G}_{hyper} = (\mathcal{V}, \mathcal{E}_{hyper})$, where $\mathcal{V}$ is the set of nodes and $\mathcal{E}_{hyper}$ is the set of hyperedges. For each hyperedge $e \in \mathcal{E}_{hyper}$, e connecting two or more nodes.

III. Method

A. Overview of our Model

In this paper, we solve the network alignment problem through our model framework, which consists of three main components. As illustrated in Fig. 2, the first module is the multi-scale encoding. We leverage node embedding at three different scales to capture first-order, multi-order, and higher-order information. The second module is stable-pair based fine-tuning. We refine the multi-order and higher-order embeddings to enhance the robustness of the node representations. The final module is the adaptive assignment, where we integrate alignment matrices from three types to derive the optimal alignment matrix.

B. Multi-scale Encoding

In order to capture multi-scale information, which includes first-order, multi-order, and higher-order information, we have obtained three types of node embeddings through three modules: node embedding in local structural learning [16], node embedding in GCN-based learning [21], and node embedding in HGCN-based learning [20].

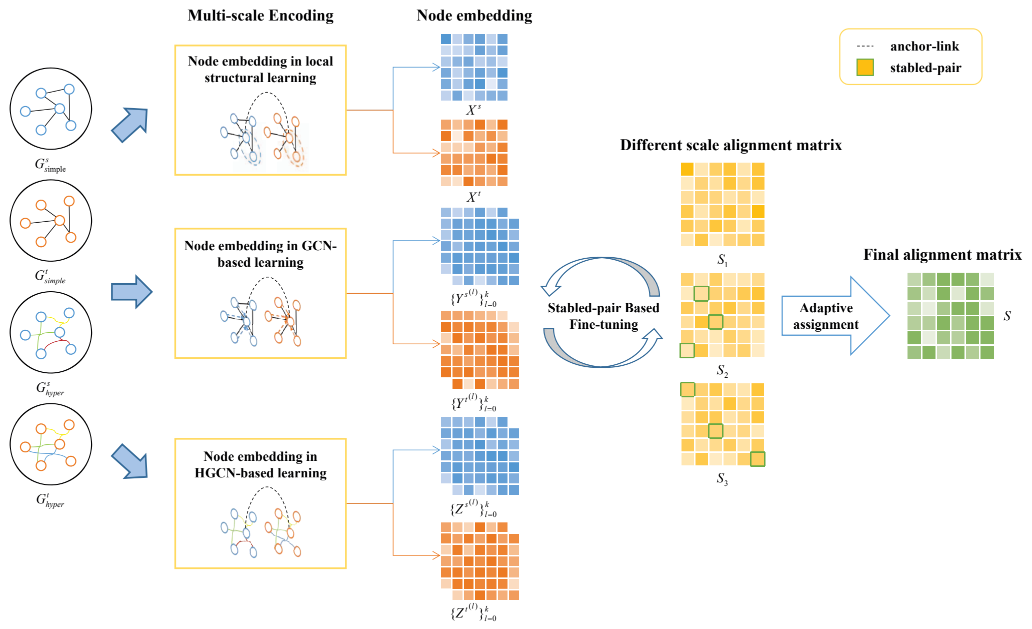

Fig. 2. The illustration of our model.$\mathcal{G}^s_{simple}$ and $\mathcal{G}^t_{simple}$ are simple graph of the source and target networks. $\mathcal{G}^s_{hyper}$ and $\mathcal{G}^t_{hyper}$ are hypergraph of the source and target networks, hyperedges are visually represented by lines of different colors. $\{\mathbf{Y}^{s(l)}\}_{l=0}^k$ and $\{\mathbf{Y}^{t(l)}\}_{l=0}^k$ represent the set of node embedding at different layers, encompassing the learned multi-order information in the source and target networks. $\{\mathbf{Z}^{s(l)}\}_{l=0}^k$ and $\{\mathbf{Z}^{t(l)}\}_{l=0}^k$ represent the set of node embedding at different layers, encompassing the learned higher-order information in the source and target networks. $\mathbf{S}_1$ is the alignment matrix obtained by first-order embedding of nodes, $\mathbf{S}_2$ and $\mathbf{S}_3$ are the alignment matrices obtained by multi-level embedding and high-order embedding, where $\mathbf{S}_2$ and $\mathbf{S}_3$ are alignment matrices obtained by integrating different layers. $\mathbf{S}$ is the final alignment matrix.

1) Node embedding in local structural learning: To learn the first-order information, we maximize the log-likelihood of two direct connected nodes v_i and v_j, i and j represent the node number in the network. The log-likelihood function is

$$L = \sum_{(v_i,v_j)\in\mathcal{E}} \log\left(p\left(v_i,v_j\right)\right) = \sum_{(v_i,v_j)\in\mathcal{E}} \log\left(\sigma\left(x_i,x_j\right)\right) \quad (1)$$

where $\sigma(\cdot)$ denotes a nonlinear activation function, and we use $sigmoid(\cdot)$. x_i and x_j represent node embeddings of v_i and v_j.

Since only a small number of edges are known, in order to obtain reliable node embedding, we adopt a negative sampling method. Finally, we maximize the following objective function:

$$\begin{aligned} L_{one}^{(\cdot)} = &\sum_{(v_i,v_j)\in\mathcal{E}} \left(\log\left(\sigma\left(x_i,x_j\right)\right)\right. \\ &+ \sum_{k=1}^{K} \mathbb{E}_{k'\sim P_{neg}}\left(1-\log\left(\sigma\left(x_i,x_k\right)\right)\right). \end{aligned} \quad (2)$$

where K is the number of negative samples of node. $(\cdot)$ represents the source network s or the target network t. P_{neg} is the probability distribution of negative sample [15] generation, which $P_{neg} \sim d_v^{3/4}$.

Next, we need to learn a matching function Θ. Given known anchor links set $(v_i^s, v_j^t) \in \mathcal{A}^{s,t}$ as supervisory information, we can optimise the loss function:

$$L_{one}^{align} = \sum_{(v_i^s,v_j^t)\in\mathcal{A}^{s,t}} ||\Theta(\mathbf{X}_i^s) - \mathbf{X}_j^t||_F \quad (3)$$

where $||\cdot||_F$ is the Frobenius norm, which is a distance metric between the predicted latent factor and the training latent factor and $\mathbf{X}_i^s$ and $\mathbf{X}_j^t$ is the embedding of two anchor nodes pair v_i^s and v_j^t. Matching function Θ utilizes a Multi-Layer Perceptron (MLP) [16] to effectively capture the intricate non-linear correlation between the source and target network. The final loss function is

$$L_{one} = (L_{one}^{align} + L_{one}^{s} + L_{one}^{t}) \quad (4)$$

2) Node embedding in GCN-based learning: In order to subtract the interference caused by network semantics and obtain globally invariant features of nodes that are independent of the network, we suggest leveraging the widely adopted Graph Convolutional Network (GCN) framework for node embedding learning.

The Graph Convolutional Network (GCN) is a deep neural network model that consists of l layers. Each hidden layer in GCN employs neighbor aggregation rules to effectively capture both structural and attribute information. We employ a multi-layer Graph Convolutional Network (GCN) that follows the subsequent layer-wise propagation rule:

$$\mathbf{Y}^{(l)} = \sigma\left(\hat{\mathbf{D}}^{-\frac{1}{2}}\hat{\mathbf{A}}\hat{\mathbf{D}}^{-\frac{1}{2}}\mathbf{Y}^{(l-1)}\mathbf{W}^{(l)}\right) \quad (5)$$

Where, $\hat{\mathbf{A}} = \mathbf{A} + \mathbf{I}$ is the self-loop adjacency matrix. $\mathbf{I}$ is the identity matrix, $\hat{\mathbf{D}}$ is degree matrix of $\hat{\mathbf{A}}$, where $\hat{\mathbf{D}}_{ii} = \sum_j \hat{\mathbf{A}}_{ij}$. $\mathbf{W}^{(l)}$ is trainable parameter matrix. $\sigma(\cdot)$ denotes a nonlinear activation function, such as $ReLU(\cdot) = \max(0, \cdot)$. $\mathbf{Y}^{(l-1)}$ is the embedding in the previous layer, $\mathbf{Y}^{(0)} = \mathbf{F}$, $\mathbf{F}$ is node feature matrix.

However, many existing methods rely on the deepest node embeddings obtained from GCN, which primarily capture local graph topology information [6]. A GCN with l layers only considers information from the l-order neighborhood. As the number of layers in GCNs increases, the node representations tend to converge to similar values, leading to the issue of oversmoothing. Furthermore, recent studies have highlighted the significance of global topology patterns in achieving precise alignment, particularly when the assumption of local structural consistency is not applicable.

Therefore, this motivates us to capture global invariant features of nodes as well. To tackle this challenge, we adopt a method where we learn node embeddings from all layers $\{\mathbf{Y}^{(l)}\}_{l=0}^{k}$. This enables us to capture network information at both lower-order and higer-order information, addressing the aforementioned concern.

The loss function comprises two main components: the mutil-layer network reconstruction loss and the mutil-layer network alignment loss.

Mutil-layer network reconstruction loss. The objective of this loss term is to preserve the structural information of the network by minimizing the distance between the hierarchical embeddings of neighboring nodes. The loss function is

$$L_{simple}^{(\cdot)\ (l)} = ||\hat{\mathbf{D}}^{-\frac{1}{2}}\hat{\mathbf{A}}\hat{\mathbf{D}}^{-\frac{1}{2}} - \mathbf{Y}^{(l)}\mathbf{Y}^{(l)T}||_F \tag{6}$$

where $||\cdot||_F$ is the Frobenius norm, $(\cdot)$ represents the source network s or target network t, and $\hat{\mathbf{D}}^{-\frac{1}{2}}\hat{\mathbf{A}}\hat{\mathbf{D}}^{-\frac{1}{2}}$ is Laplacian matrix and $\mathbf{Y}^{(l)}\mathbf{Y}^{(l)T}$ is the generated reconstructed representation.

Mutil-layer network alignment loss. By utilizing anchor node pairs as supervisory information and minimizing the embedding of these anchor nodes, this loss function guarantees that the embeddings of anchor pairs in each layer exhibit similarity.

$$L_{simple}^{align\ (l)} = \sum_{(v_i^s, v_j^t) \in \mathcal{A}^{s,t}} ||\mathbf{Y}_i^{s(l)} - \mathbf{Y}_j^{t(l)}||_F \tag{7}$$

where $\mathbf{Y}_i^{s(l)}$ and $\mathbf{Y}_j^{t(l)}$ is the embedding of two anchor nodes pair v_i^s and v_j^t in each layer.

The final loss function is

$$L_{simple} = \sum_{l=0}^{k}(L_{simple}^{align\ (l)} + L_{simple}^{s\ (l)} + L_{simple}^{t\ (l)}) \tag{8}$$

3) Node embedding in HGCN-based learning: Hypergraphs [22], [23] have demonstrated significant potential in representing higher-order relations among entities. In contrast to previous approaches of hypergraph convolutional networks, we firmly believe that hyperedges carry valuable information and should be effectively utilized. Hence, in our method, we not only perform convolution operations on nodes but also on hyperedges. By utilizing the hypergraph autoencoder, nodes can learn both their individual information and the super edge information, resulting in enhanced network alignment capabilities.

Encoder of Nodes. For the encoder of nodes, we utilize l-layer Hypergraph Convolutional Network (HGCN) to learn node embedding, and the formulation [24] as follows:

$$\mathbf{Z}^{(l)} = \sigma\left(\mathbf{E}^{-\frac{1}{2}}\mathbf{H}\mathbf{\Omega}\mathbf{B}^{-1}\mathbf{H}^T\mathbf{E}^{-\frac{1}{2}}\mathbf{Z}^{(l-1)}\mathbf{T}^{(l)}\right) \tag{9}$$

where $\mathbf{Z}^{(l)}$ represents the node embedding learned of lth-layer HGCN, $\mathbf{Z}^{(0)} = \mathbf{F}$. $\mathbf{H}$ is an incidence matrix of hypergraph represented by node-hyperedge, $\mathbf{\Omega}$ is a diagonal positive weights matrix of hyperedge. Specifically, in scenarios where there is no prior information available regarding the importance of hyperedges, $\mathbf{\Omega}$ is set as a identity matrix. $\mathbf{E}$ is the diagonal node degree matrix with $\mathbf{E}_{ii} = \sum_j \mathbf{\Omega}_{jj}\mathbf{H}_{ij}$ and $\mathbf{B}$ is the diagonal hyperedge degree matrix with $\mathbf{B}_{jj} = \sum_i \mathbf{H}_{ij}$. $\mathbf{T}^{(l)}$ is trainable parameter matrix. Therefore, by performing multiple hypergraph convolution operations, we obtain the embedding of nodes on each layer.

Encoder of Hyperedges. For the encoder of hyperedges, to enable filtering on the hyperedges, we can consider the nodes and hyperedges in the hypergraph $\mathbf{H}$ as the hyperedges and nodes, respectively, in a new hypergraph $\mathbf{H}^T$. We utilize l-layer Hypergraph Convolutional Network (HGCN) to learn hyperedge embedding, and the formulation [18], [24] as follows:

$$\mathbf{P}^{(l)} = \sigma\left(\mathbf{L}^{-\frac{1}{2}}\mathbf{H}^T\mathbf{\Phi}\mathbf{J}^{-1}\mathbf{H}\mathbf{L}^{-\frac{1}{2}}\mathbf{P}^{(l-1)}\mathbf{U}^{(l)}\right) \tag{10}$$

where $\mathbf{P}^{(l)}$ represents the hyperedge embedding learned of lth-layer HGCN, $\mathbf{P}^{(0)} = \mathbf{V}$. $\mathbf{V}$ is attribute matrix of hyperedges. Since hyperedge data often lacks features, we set $\mathbf{V}$ as the identity matrix. $\mathbf{\Phi}$ is a diagonal positive weights matrix of nodes. Specifically, in scenarios where there is no prior information available regarding the importance of nodes, $\mathbf{\Phi}$ is set as a one-hot matrix. $\mathbf{L}$ is the diagonal hyperedge degree matrix with $\mathbf{L}_{jj} = \sum_i \mathbf{\Phi}_{ii}\mathbf{H}_{ji}$ and $\mathbf{J}$ is the diagonal node degree matrix with $\mathbf{J}_{ii} = \sum_j \mathbf{H}_{ji}$. $\mathbf{U}^{(l)}$ is trainable parameter matrix. Therefore, by performing multiple hypergraph convolution operations, we obtain the embedding of hyperedges on each layer.

Decoder. In the decoder, our aim is to reconstruct the hypergraph $\mathbf{H}$ by restoring the relationships between nodes and hyperedges. To achieve this, we strive to decode the potential representations of nodes and hyperedge data to closely resemble the original hypergraph. and the reconstructed incidence matrix $\tilde{\mathbf{H}}^{(l)}$ is

$$\tilde{\mathbf{H}}^{(l)} = sigmoid(\mathbf{Z}^{(l)}\mathbf{P}^{(l)\top}) \tag{11}$$

For a network, we have a hypergraph reconstruction loss function in each layer as follows:

$$L_{hyper}^{(\cdot)\ (l)} = ||\tilde{\mathbf{H}}^{(l)} - \mathbf{H}||_F \tag{12}$$

where $(\cdot)$ represents the source network s or target network t.

To capture high-order network information from both lower and higher layers, we learn the node embeddings from different layers $\{\mathbf{Z}^{(l)}\}_{l=0}^{k}$. This obtains higher-order information from different layers together.

Mutil-layer network alignment. To ensure that the node embeddings of two networks reside in a similar embedding space for network alignment purposes, we minimize the embedding distance between anchor node pairs. The formula is as follows:

$$L_{hyper}^{align\ (l)} = \sum_{(v_i^s, v_j^t)\in\mathcal{A}^{s,t}} ||\mathbf{Z}_i^{s(l)} - \mathbf{Z}_j^{t(l)}||_F \tag{13}$$

where $\mathbf{Z}_i^{s(l)}$ and $\mathbf{Z}_j^{t(l)}$ is the embedding of two anchor nodes pair v_i^s and v_j^t in each layer.

The final loss function is

$$L_{hyper} = \sum_{l=0}^{k}((1-\alpha)L_{hyper}^{align\ (l)} + \alpha(L_{hyper}^{s\ (l)} + L_{hyper}^{t\ (l)})) \tag{14}$$

C. Stabled-pair Based Fine-tuning

We perform fine-tuning on the learned multi-order node embeddings and high-order embeddings. This process involves further refinement and adjustment to optimize the quality and accuracy of the embeddings.

Alignment Computed. The alignment matrix is utilized to establish the corresponding relationship between nodes in two or more networks. The calculation of the alignment matrix for the first-order embedding of nodes is as follows:

$$\mathbf{S}_1 = \mathbf{X}^s\mathbf{X}^{t^T} \tag{15}$$

The calculation of the alignment matrix for the multi-order embedding of nodes in each layer is as follows:

$$\mathbf{S}_2{}^{(l)} = \mathbf{Y}^{s(l)}\mathbf{Y}^{t(l)^T} \tag{16}$$

Therefore, the final alignment matrix obtained through multi-order embedding is as follows:

$$\mathbf{S}_2 = \sum_{l=0}^{k}\mathbf{Y}^{s(l)}\mathbf{Y}^{t(l)^T} \tag{17}$$

Similarly, the alignment matrix for the high-order embeddings of nodes in each layers and final alignment matrix can be obtained as

$$\mathbf{S}_3{}^{(l)} = \mathbf{Z}^{s(l)}\mathbf{Z}^{t(l)^T} \tag{18}$$

$$\mathbf{S}_3 = \sum_{l=0}^{k}\mathbf{Z}^{s(l)}\mathbf{Z}^{t(l)^T} \tag{19}$$

Stabled-pair. According to our assumption, if a pair of anchor nodes have similar embeddings in any layer of the model, then they are considered stable. Mathematically, a stabled-pair $< v_i^s, v_j^t >$ satisfies the following conditions:

$$\operatorname{argmax}\mathbf{S}^{(l-1)}\left(v_i^j, u\right) = \operatorname{argmax}\mathbf{S}^{(l)}\left(v_i^j, u\right) = v_j^t \tag{20}$$

In this context, a stable pair refers to a scenario where a source network node v_i^s has the highest score in the alignment matrix across all layers, corresponding to a target network node v_j^t.

Aggregate weight fine-tuning. In fact, assigning larger aggregation weights to stable pair means that their adjacent potential anchor nodes can receive more similar information. The more stable pairs of nodes there are around potential anchor nodes, the easier it becomes to identify them.

For multi-order embedding of nodes, we propose an improved GCN framework. Given two reinforcement factor vectors $\mathbf{q}^s$ and $\mathbf{q}^t$, with all values set to 1. If $v_i^s \in \mathcal{V}^s$ and $v_j^t \in \mathcal{V}^t$ are identified as stabled-pair, the corresponding reinforcement factors are updated as follows:

$$\mathbf{q}^s(v_i^s) = \beta\cdot\mathbf{q}^s(v_i^s) \tag{21}$$

$$\mathbf{q}^t(v_j^t) = \beta\cdot\mathbf{q}^t(v_j^t) \tag{22}$$

where β>1 is the reinforcement rate. The new aggregation rule of GCN is as follows:

$$\mathbf{Y}^{(l)} = \sigma\left(\hat{\mathbf{R}}^{-\frac{1}{2}}\hat{\mathbf{A}}\hat{\mathbf{R}}^{-\frac{1}{2}}\mathbf{Y}^{(l-1)}\mathbf{W}^{(l)}\right) \tag{23}$$

where $\hat{\mathbf{R}} = \hat{\mathbf{D}}\mathbf{Q}$ and $\mathbf{Q}$ is a diagonal reinforcement matrix in which $\mathbf{Q}_{ii} = \mathbf{q}_i$.

One iteration can consist of four steps: (a) Identifying stable pairs from the alignment matrix in each layer. (b) Updating the aggregation coefficients for each stable pair in each layer. (c) Generating new node embeddings for each layer. (d) Computing new alignment matrices for each layer. We keep track of the sum of the top-1 alignment scores for each refined alignment matrix, *i.e.*, $score = \sum_{v\in\mathcal{V}} max(\mathbf{S}(v))$. Finally, we return the alignment matrix with the highest score. Similarly, the same optimization applies to high-order embedding of nodes.

D. Adaptive assignment

1) Final alignment matrix: We combine the alignment matrix obtained from multi-scale node information to derive the final adjacency matrix. The final adjacency matrix is defined as

$$\mathbf{S} = \theta_1\mathbf{S}_1 + \theta_2\mathbf{S}_2 + \theta_3\mathbf{S}_3 \tag{24}$$

where $\mathbf{S}_1$ represents the alignment matrix from the first-order node information, $\mathbf{S}_2$ represents the alignment matrix from the multi-order node information, and $\mathbf{S}_3$ represents the alignment matrix from the higher-order node information. θ_1, θ_2 and θ_3 represent the trainable importance coefficient.

2) Adaptive assignment for final alignment matrix: We utilize network augmentation to automatically optimize the importance coefficient.

Network Augmentation. Given a source or target network $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, the adjacency matrix of this network is $\mathbf{A}$. We generate an augmented network $\mathcal{G}_p = (\mathcal{V}_p, \mathcal{E}_p)$ with the adjacency matrix $\mathbf{A}_p$ using the following procedure:

$$\mathbf{A}_p = \mathbf{P}\mathbf{A}\mathbf{P}^T \tag{25}$$

TABLE I
STATISTIC OF THE NETWORKS

Networks	# nodes	# edges	# attributes
Douban Online	3906	8164	538
Douban Offline	1118	1511	538
Allmovie	6011	124709	14
Imdb	5713	119073	14
ACM	9872	39561	17
DBLP	9916	44808	17

where $\mathbf{P}$ is a random permutation matrix, and $\mathbf{P}_{ij} = 1$ indicates that node v_i^s from the original network corresponds to node v_j^t in the augmented network; otherwise, $\mathbf{P}_{ij} = 0$.

Importance coefficient fusion mechanism. We leverage the augmentation mechanism to enable the model to capture the importance of different alignment matrices. Given the original graph and its augmented version, a good model should maintain similar embeddings between the two versions and accurately align the network nodes. We optimize the following function to learn the importance coefficient:

$$\underset{\theta}{\operatorname{argmax}} \sum_{(v_i^{\mathcal{G}}, v_j^{\mathcal{G}'}) \in \mathbf{T}} \mathbf{S}_{ij} \tag{26}$$

where $\mathcal{G}'$ represents the augmented version of the original network $\mathcal{G}$. The objective is to maximize the alignment matrix score for all anchor links $(v_i^{\mathcal{G}}, v_j^{\mathcal{G}'})$ between $\mathcal{G}$ and $\mathcal{G}'$. $\mathbf{T}$ is the set of anchor links for two networks $\mathcal{G}$ and $\mathcal{G}'$.

IV. EXPERIMENT

In this section, we have conducted extensive experiments on our proposed model as well as the baseline methods using three real-world datasets. Our experiments have yielded state-of-the-art results, surpassing existing approaches.

A. Experimental Settings

1) Dataset: We select three widely used real-world benchmark datasets. The details of the three datasets are summarized in Table I.

2) Baselines: We compare our model with six state-of-the-art network alignment baseline methods that are considered to be the most advanced in the field, including **GAlign** [25], **NAME** [26], **WAlign** [11], **NeXtAlign** [27], **DANA** [7] and **SLOTAlign** [28].

3) Evaluation Metrics: Two quantitative metrics are employed to thoroughly evaluate and compare the efficacy of various alignment approaches. We use the evaluation metric $Hits@k$ [27] and MAP (Mean Average Precision) [16] to measure the performance of model network alignment, $Hits@k$ calculates the percentage of the nodes in $\mathcal{V}^t$ whose groundtruth alignment results in $\mathcal{V}^s$ is in the top-k candidates, which is defined as

$$Hits@k = \frac{1}{N_{test}} \sum_{i=1}^{N_{test}} \mathbf{1}_i \{success@k\} \tag{27}$$

where $\mathbf{1}_i \{success@k\} = 1$ if hit. And N_{test} is the size of the anchor link test set.

Another evaluation metric MAP is defined as

$$MAP = \frac{1}{N_{test}} \sum_{i=1}^{N_{test}} \frac{1}{r_i} \tag{28}$$

where r_i is the rank of ith true anchor node's alignment score in the corresponding row of $\mathbf{S}$.

We conduct ten-time experiments for each model and present their average results $Hits@k$ and MAP. To facilitate recording, we magnify the results by 100 times.

4) Parameter settings: In the negative sampling, the number of negative samples is set to 10. Both the GCN and HGCN layers l are set to 2. The hyperparameter a is set to 0.01. The embbdding dimensions used during the training process are set to 200, and the reinforcement rate β is set to 1.1. The parameters in all baseline methods are set to default values.

B. Effectiveness Analysis

We use 20% training data to train our model and other baselines respectively. The network alignment performance is shown in Table II. Compared with the most advanced methods, the average improvement of our method is about 3% by $Hits@10$. The embedding-based alignment method (*i.e.*, GAlign and WAlign) eliminates the problem of over smoothing to some extent by introducing negative alignment pairs and NeXtAlign introduces negative sampling to alleviate the over smoothing problem. They only consider information on a single scale. DANA makes the representation of nodes more task-specific by suppressing the unique characteristics of the network, which destroys the unique structure of the graph. NAME fuses multiple node embedding information, but lacks enhancement of individual information. SLOTAlign integrates intra-graph structure modeling and cross-graph optimal transport alignment for network alignment. However, there may be deviations in the fused node embeddings obtained from multi-view structure learning.

Additionally, we conducted experiments by varying the training ratio on three datasets to assess the impact of known anchor links. As show in Fig. 3, we can see that the alignment performance of our model changes little in different training ratios, which indicates its robustness. This is particularly important considering that anchor links are often sparse in real-world scenarios. Hence, our model is a competitive choice when working with limited known anchor links.

C. Ablation Study

To isolate and assess the individual contributions of different components in our proposed model, we constructed several variants and evaluated their performance on three real datasets.

- **Our-F:** remove the first-order information learning module of the node, *i.e.* node embedding in local structural learning.
- **Our-M:** remove the multi-order information learning module of the node, *i.e.* node embedding in GCN-based learning.

TABLE II
RESULTS OF 20% TRAINING DATA

Methods	Douban Online-Offline			Allmovie-Imdb			ACM-DBLP		
	Hits@5	Hits@10	MAP	Hits@5	Hits@10	MAP	Hits@5	Hits@10	MAP
NAME	<u>75.64</u>	<u>84.80</u>	<u>61.27</u>	<u>93.53</u>	<u>95.00</u>	90.57	75.45	86.25	58.35
GAlign	67.98	77.46	55.26	84.31	87.28	80.05	79.32	85.41	67.39
WAlign	43.11	50.36	33.34	76.66	81.33	68.68	<u>85.20</u>	<u>90.12</u>	73.61
SLOTAlign	72.81	77.28	60.48	92.72	93.10	<u>91.45</u>	83.79	87.62	<u>73.68</u>
NextAlign	54.41	64.25	36.20	77.97	80.92	71.37	78.68	86.54	59.90
DANA	43.58	54.92	29.76	81.98	85.56	75.04	68.27	76.94	47.00
Our	**79.89**	**88.72**	**63.39**	**95.17**	**96.04**	**92.59**	**86.98**	**93.30**	**73.88**

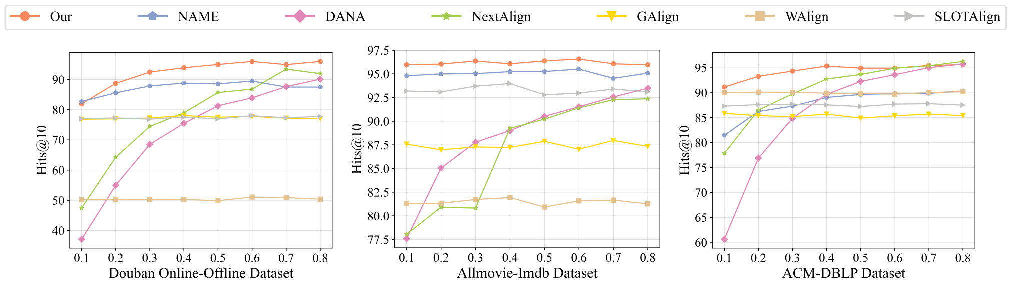

Fig. 3. Performance of our method and baselines under different training ratios

TABLE III
ABLATION STUDY

Methods	Douban Online-Offline			Allmovie-Imdb			ACM-DBLP		
	Hits@5	Hits@10	MAP	Hits@5	Hits@10	MAP	Hits@5	Hits@10	MAP
Our-F	73.07	81.12	59.38	89.25	91.11	86.54	80.65	86.11	71.65
Our-M	76.09	85.14	61.70	94.61	95.68	92.13	83.42	91.03	67.43
Our-H	75.20	85.36	58.37	93.91	95.19	89.83	79.68	88.18	63.88
Our-FT	79.44	87.26	63.06	93.60	94.98	90.44	85.99	92.71	71.74
Our	**79.89**	**88.72**	**63.39**	**95.17**	**96.04**	**92.59**	**86.98**	**93.30**	**73.88**

- **Our-H:** remove the higher-order information learning module of the node, *i.e.* node embedding in HGCN-based learning.
- **Our-FT:** remove the fine-tuning operation, *i.e.* stabled-pair based fine-tuning.

As shown in Table III, our method achieved the best alignment performance compared to the four variants. Specifically, when compared to three variants (Our-F, Our-M, Our-H) across different datasets, our model exhibits noticeable improvements in network alignment performance metrics such as $Hits@5$, $Hits@10$ and MAP. Through this analysis, we can conclude that the incorporation of first-order information, multi-order information, and higher-order information significantly enhances the alignment performance of the network. When it comes to stabled-pair based fine-tuning, our model exhibits an improvement of approximately 1% in $Hits@10$ compared to the Our-FT variant. From this observation, it is evident that our fine-tuning operations on node multi-order embeddings and node higher-order embeddings have played a significant role.

D. Parameter Sensitivity Analysis

In this subsection, we study the influence of different parameter changes on the model effect.

1) Embedding dimension: We conduct experiments on three real-world datasets with different node embedding dimensions of GCN layers and HGCN layers, and the results are shown in the left subfigure in Fig. 4. It can be observed that as the embedding dimension ranges from 50 to 300, the network alignment performance shows an initial significant improvement, followed by a gradual slowdown in improvement. And when embedding dimension is greater than 300, the network alignment performance becomes slightly improved or decreased. Generally, we should avoid selecting a large number of dimensions since it does not lead to significant performance improvement, while increasing both time and space complexity.

2) Constructing hypergraphs: We choose to construct a hypergraph through k-hop neighbors of graph vertices. In

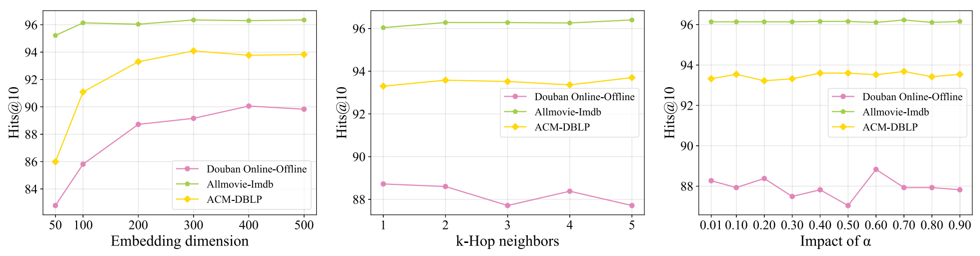

Fig. 4. Performance of our method with different hyperparameters (left: Embedding dimension; center: k-Hop neighbors; right: Impact of α).

order to reduce complexity, we chose to experiment in the neighborhood range of 1 to 5 hops. As shown in the center subfigure in Fig. 4, we have noticed that the effectiveness of network alignment is significantly enhanced when the value of k is between 1 and 2, with the exception of the Douban Online-Offline dataset. However, when the value of k is within the range of 2 to 5, the network alignment performance shows only marginal improvements or slight decreases. It can be inferred that incorporating moderate higher-order information is beneficial for improving performance.

3) Size of α: It can be seen from the right subfigure in Fig. 4, In terms of overall results, the Allmovie-Imdb and ACM-DBLP datasets show insensitivity to α, with no significant changes in alignment performance as α varies. The performance of the Douban online and offline datasets exhibits a relatively stable pattern, with the best results achieved when the parameter α is set to 0.6. This further substantiates the significant robustness of our proposed model.

V. Related Work

A. Network Alignment

The network alignment problem refers to the task of identifying the corresponding relationships between nodes in distinct networks. Network alignment research is mainly divided into structured-based network alignment method [4], [6], [29] and embedding-based network alignment method [7], [8], [30]. The early structured-based network alignment methods involved formulating the maximum common subgraph for different networks, which was addressed by comparing the topological similarity between the two networks. However, these solutions suffered from poor efficiency and limited versatility.

Therefore, in recent years, most studies are based on embedding-based network alignment methods, which are mainly divided into three categories. The first category [12], [29], [31] is to embed different networks into their own potential space, and then learn the network mapping function; The second category [8], [32], [33] is to embed different networks into a common potential space, and use similarity for network alignment; The third category [7], [30], [34] is to combine the Generative Adversarial Networks to eliminate features that are useless for network alignment tasks.

We introduces a novel model that integrates multiple types of information, including first-order information, multi-order information, and higher-order information. By leveraging these diverse sources of information, our model aims to enhance node embedding and effectively align potential anchor nodes that deviate from the assumptions consistency.

B. Hypergraph Learning

A hypergraph [18], [24], [35]–[38] is a graph where an edge can connect two or more nodes. Hypergraph learning has received widespread attention due to its advantages in representing high-order correlations, and has achieved successful performance improvements in various fields, such as recommendation systems [39], social networks [40], image ranking [41] and computer vision [42].

Hypergraph neural networks have been developed for effective hypergraph representation. Hypergraph neural network (HGNN) [18] uses the hypergraph structure to encode the correlation of higher-order data, and extends the convolution operation to the hypergraph learning process based on the hypergraph Laplacian operator and truncated Chebyshev polynomials. HyperGCN [36] proposes the generalized hypergraph Laplacian and explores adding the hyperedge information through mediators.

In this paper, we use the hypergraph autoencoder to capture the imformation across nodes and hyperedges. By enriching the node information with hyperedge information, our model enables the effective alignment of potential anchor nodes that may deviate from the consistency assumption.

VI. Conclusion

In this paper, we propose a novel multi-scale embedding-based network alignment method. To obtain comprehensive information, we captured three types of node embeddings and highlighted the significance of different dimensional information. Then, we enhance the multi-order and high-order embedding information of nodes by applying stable-pair based fine-tuning method. Finally, we use a fusion mechanism based on importance weights to fuse the alignment matrices obtained from three types of node embedding information to obtain the optimal alignment matrix. We have carried out extensive experiments on three real-world datasets, and the experimental results prove the effectiveness of our method.

VII. Acknowledgment

This work was supported in part by the Fundamental Research Funds for the Provincial Universities of Zhejiang Grant

GK229909299001-008 and Grant GK239909299001-028, in part by the Zhejiang Laboratory Open Research Project under Grant K2022QA0AB01, and in part by the National Natural Science Foundation of China under Grant 62372146.

References

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

[2] Y. Zhang, J. Tang, Z. Yang, J. Pei, and P. S. Yu, "Cosnet: Connecting heterogeneous social networks with local and global consistency," in *Proceedings of the 21th ACM SIGKDD international conference on knowledge discovery and data mining*, 2015, pp. 1485–1494.

[3] L. Hu, J. Cao, G. Xu, L. Cao, Z. Gu, and C. Zhu, "Personalized recommendation via cross-domain triadic factorization," in *Proceedings of the 22nd international conference on World Wide Web*, 2013, pp. 595–606.

[4] M. Bayati, D. F. Gleich, A. Saberi, and Y. Wang, "Message-passing algorithms for sparse network alignment," *ACM Transactions on Knowledge Discovery from Data (TKDD)*, vol. 7, no. 1, pp. 1–31, 2013.

[5] O. Kuchaiev, T. Milenković, V. Memišević, W. Hayes, and N. Pržulj, "Topological network alignment uncovers biological function and phylogeny," *Journal of the Royal Society Interface*, vol. 7, no. 50, pp. 1341–1354, 2010.

[6] 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*, 2016, pp. 1345–1354.

[7] H. Hong, X. Li, Y. Pan, and I. W. Tsang, "Domain-adversarial network alignment," *IEEE Transactions on Knowledge & Data Engineering*, vol. 34, no. 07, pp. 3211–3224, 2022.

[8] L. Liu, W. K. Cheung, X. Li, and L. Liao, "Aligning users across social networks using network embedding." in *Ijcai*, vol. 16, 2016, pp. 1774–80.

[9] J.-D. Park, C. Tran, W.-Y. Shin, and X. Cao, "On the power of gradual network alignment using dual-perception similarities," *arXiv preprint arXiv:2201.10945*, 2022.

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

[11] J. Gao, X. Huang, and J. Li, "Unsupervised graph alignment with wasserstein distance discriminator," in *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, 2021, pp. 426–435.

[12] C. Zheng, L. Pan, and P. Wu, "Jora: Weakly supervised user identity linkage via jointly learning to represent and align," *IEEE Transactions on Neural Networks and Learning Systems*, 2022.

[13] Q. Li, Z. Han, and X.-M. Wu, "Deeper insights into graph convolutional networks for semi-supervised learning," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 32, no. 1, 2018.

[14] L. Liu, P. Chen, X. Li, W. K. Cheung, Y. Zhang, Q. Liu, and G. Wang, "Wl-align: Weisfeiler-lehman relabeling for aligning users across networks via regularized representation learning," *IEEE Transactions on Knowledge and Data Engineering*, 2023.

[15] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," *Advances in neural information processing systems*, vol. 26, 2013.

[16] T. Man, H. Shen, S. Liu, X. Jin, and X. Cheng, "Predict anchor links across social networks via an embedding approach." in *Ijcai*, vol. 16, 2016, pp. 1823–1829.

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

[18] Y. Feng, H. You, Z. Zhang, R. Ji, and Y. Gao, "Hypergraph neural networks," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 33, no. 01, 2019, pp. 3558–3565.

[19] H. Fan, F. Zhang, Y. Wei, Z. Li, C. Zou, Y. Gao, and Q. Dai, "Heterogeneous hypergraph variational autoencoder for link prediction," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 44, no. 8, pp. 4125–4138, 2021.

[20] H. Wu and M. K. Ng, "Hypergraph convolution on nodes-hyperedges network for semi-supervised node classification," *ACM Transactions on Knowledge Discovery from Data (TKDD)*, vol. 16, no. 4, pp. 1–19, 2022.

[21] T. N. Kipf and M. Welling, "Variational graph auto-encoders," *arXiv preprint arXiv:1611.07308*, 2016.

[22] Y. Dong, W. Sawin, and Y. Bengio, "Hnhn: Hypergraph networks with hyperedge neurons," *arXiv preprint arXiv:2006.12278*, 2020.

[23] Q. Dai and Y. Gao, "Neural networks on hypergraph," in *Hypergraph Computation*. Springer, 2023, pp. 121–143.

[24] S. Bai, F. Zhang, and P. H. Torr, "Hypergraph convolution and hypergraph attention," *Pattern Recognition*, vol. 110, p. 107637, 2021.

[25] H. T. Trung, T. Van Vinh, N. T. Tam, H. Yin, M. Weidlich, and N. Q. V. Hung, "Adaptive network alignment with unsupervised and multi-order convolutional networks," in *2020 IEEE 36th International Conference on Data Engineering (ICDE)*. IEEE, 2020, pp. 85–96.

[26] T. T. Huynh, C. T. Duong, T. T. Nguyen, V. Van Tong, A. Sattar, H. Yin, and Q. V. H. Nguyen, "Network alignment with holistic embeddings," *IEEE Transactions on Knowledge and Data Engineering*, 2021.

[27] S. Zhang, H. Tong, L. Jin, Y. Xia, and Y. Guo, "Balancing consistency and disparity in network alignment," in *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, 2021, pp. 2212–2222.

[28] J. Tang, W. Zhang, J. Li, K. Zhao, F. Tsung, and J. Li, "Robust attributed graph alignment via joint structure learning and optimal transport," *arXiv preprint arXiv:2301.12721*, 2023.

[29] F. Zhou, L. Liu, K. Zhang, G. Trajcevski, J. Wu, and T. Zhong, "Deeplink: A deep learning approach for user identity linkage," in *IEEE INFOCOM 2018-IEEE Conference on computer communications*. IEEE, 2018, pp. 1313–1321.

[30] C. Zheng, L. Pan, and P. Wu, "Camu: Cycle-consistent adversarial mapping model for user alignment across social networks," *IEEE Transactions on Cybernetics*, 2021.

[31] Q. Sun, X. Lin, Y. Zhang, W. Zhang, and C. Chen, "Towards higher-order topological consistency for unsupervised network alignment," *arXiv preprint arXiv:2208.12463*, 2022.

[32] X. Du, J. Yan, and H. Zha, "Joint link prediction and network alignment via cross-graph embedding." in *IJCAI*, 2019, pp. 2251–2257.

[33] Y. Wang, W. Wang, Z. Zhen, Q. Peng, P. Jiao, W. Liang, M. Shao, and Y. Sun, "Geometry interaction network alignment," *Neurocomputing*, vol. 501, pp. 618–628, 2022.

[34] C. Chen, W. Xie, T. Xu, Y. Rong, W. Huang, X. Ding, Y. Huang, and J. Huang, "Unsupervised adversarial graph alignment with graph embedding," *arXiv preprint arXiv:1907.00544*, 2019.

[35] D. Zhou, J. Huang, and B. Schölkopf, "Learning with hypergraphs: Clustering, classification, and embedding," *Advances in neural information processing systems*, vol. 19, 2006.

[36] N. Yadati, M. Nimishakavi, P. Yadav, V. Nitin, A. Louis, and P. Talukdar, "Hypergcn: A new method for training graph convolutional networks on hypergraphs," *Advances in neural information processing systems*, vol. 32, 2019.

[37] D. Arya, D. K. Gupta, S. Rudinac, and M. Worring, "Hypersage: Generalizing inductive representation learning on hypergraphs," *arXiv preprint arXiv:2010.04558*, 2020.

[38] T. Wei, Y. You, T. Chen, Y. Shen, J. He, and Z. Wang, "Augmentations in hypergraph contrastive learning: Fabricated and generative," *arXiv preprint arXiv:2210.03801*, 2022.

[39] X. Chen, K. Xiong, Y. Zhang, L. Xia, D. Yin, and J. X. Huang, "Neural feature-aware recommendation with signed hypergraph convolutional network," *ACM Transactions on Information Systems (TOIS)*, vol. 39, no. 1, pp. 1–22, 2020.

[40] R. Zhang, Y. Zou, and J. Ma, "Hyper-sagnn: a self-attention based graph neural network for hypergraphs," *arXiv preprint arXiv:1911.02613*, 2019.

[41] Y. Huang, Q. Liu, S. Zhang, and D. N. Metaxas, "Image retrieval via probabilistic hypergraph ranking," in *2010 IEEE computer society conference on computer vision and pattern recognition*. IEEE, 2010, pp. 3376–3383.

[42] Y. Cai, Z. Zhang, Z. Cai, X. Liu, and X. Jiang, "Hypergraph-structured autoencoder for unsupervised and semisupervised classification of hyperspectral image," *IEEE Geoscience and Remote Sensing Letters*, vol. 19, pp. 1–5, 2021.