Adaptive Network Alignment with Unsupervised and Multi-order Convolutional Networks

Huynh Thanh Trung[1], Tong Van Vinh[2], Nguyen Thanh Tam[3], Hongzhi Yin[4*],
Matthias Weidlich[5], Nguyen Quoc Viet Hung[1]

[1]*Griffith University,* [2]*Hanoi University of Science and Technology (HUST),* [3]*Ho Chi Minh City University of Technology (HUTECH),*
[4]*The University of Queensland,* [5]*Humboldt-Universität zu Berlin*

***Abstract*—Network alignment is the problem of pairing nodes between two graphs such that the paired nodes are structurally and semantically similar. A well-known application of network alignment is to identify which accounts in different social networks belong to the same person. Existing alignment techniques, however, lack scalability, cannot incorporate multi-dimensional information without training data, and are limited in the consistency constraints enforced by an alignment. In this paper, we propose a fully unsupervised network alignment framework based on a multi-order embedding model. The model learns the embeddings of each node using a graph convolutional neural representation, which we prove to satisfy consistency constraints. We further design a data augmentation method and a refinement mechanism to make the model adaptive to consistency violations and noise. Extensive experiments on real and synthetic datasets show that our model outperforms state-of-the-art alignment techniques. We also demonstrate the robustness of our model against adversarial conditions, such as structural noises, attribute noises, graph size imbalance, and hyper-parameter sensitivity.**

***Index Terms*—network alignment, network embedding, GCN**

I. Introduction

Networks, aka graphs, are structures that naturally capture relations between entities in data domains and information systems. Many applications, however, analyse not one, but multiple networks, e.g., for data discovery [11], social network analytics [29], knowledge graph reconciliation [40], and pattern matching in protein networks [25]. Network alignment, the task of pairing nodes between two isomorphic or near-isomorphic networks such that the paired nodes are similar w.r.t. network structure and attribute information, is the foundation to extract valuable knowledge in these applications [34]. For example, in social networks, the detection of different accounts (e.g., Facebook, Twitter) of the same user facilitates personalized advertisement, friend suggestion, and content recommendation [39]. In bioinformatics, aligning protein networks may reveal new patterns of protein-protein interactions, such as cross-species gene prioritization [25].

Network alignment faces challenges related to efficiency, the richness of incorporated information, and the strictness of alignment constraints. Network alignment is often formulated as a maximum bipartite matching problem. Yet, many of its variants, such as the maximum common subgraph problem, are NP-hard [2]. Hence, many approaches resort to a matrix factorization formulation, e.g., IsoRank [32], NetAlign [2], UniAlign [21], FINAL [39], and REGAL [16]. Such spectral methods fail to deal with very large networks, since the required computational effort grows quickly with the network size (e.g., cubic growths for FINAL [39]).

Moreover, networks often comprise heterogeneous information including network structure and node features. While different types of information may be useful for network alignment, the lack of a common modality imposes challenges [33]. Information integration may be guided by path-based constraints and proximity rules [29]. Yet, such approaches are domain-specific and require manual user efforts.

Concerning the constraints used for alignment, most techniques employ some strict notion of structural consistency, e.g., the relation between pairs of nodes shall always be maintained across two aligned networks. However, such strict constraints are sensitive to network perturbations. For example, a person may have more connections in one social network (e.g., Facebook) than in others (e.g., LinkedIn, YouTube). Moreover, node pairings can also be one-to-many, especially for networks with different sizes. In that case, alignment heuristics that employ strict notions of structural consistency, such as a one-to-one constraint [25], are no longer applicable.

To address the outlined challenges, this paper proposes an embedding-based network alignment model. In essence, our idea is to embed nodes of two networks into multi-order feature vectors and then align the networks based on the similarity of their node embeddings. We realize this idea through the design of a graph convolutional network (GCN) to learn the embeddings. More specifically, we summarize our contributions as follows:

- We propose **GAlign**, a framework for unsupervised network alignment without any prior knowledge on the relation of the networks (aka anchor links). Since this framework is grounded in multi-dimensional embeddings, rich network information (network structure, node features) can be represented, regardless of its modality.
- We develop the theoretical background for exploiting the multi-order nature of GCN for network alignment. We show that the consistency constraints used separately by existing techniques can be unified in the same GCN model. This enables us to incorporate these constraints directly during node embedding, thereby supporting expressive constraints beyond strict structural consistency.
- Using this theory, we design a specific GCN model for

*Corresponding author

2375-026X/20/$31.00 ©2020 IEEE
DOI 10.1109/ICDE48307.2020.00015

network alignment. Unlike existing methods [23, 24, 41] that separate embedding and alignment, we learn embeddings jointly for the networks. Hence, reconciliation of their embedding spaces is no longer needed.
- We propose an augmented learning process to make the model adaptive to noise (e.g. violations of consistency constraints), which are common in real-world networks. The process includes the design of a perturbation-based data augmentation and a noise-adaptive loss function.

We evaluate our proposal in extensive experiments on real and synthetic networks. Our framework not only achieves superior performance compared to baseline techniques but also exhibits robustness to noise and network size imbalance.

The remainder is organized as follows. §II introduces a model and problem statement for network alignment. §III provides an overview of our approach. §IV discusses the theoretical background for the design of our specific GCN embedding model. §V describes how to transform the original networks into multi-order embeddings. §VI shows how to compute and refine the alignment matrix from the unsupervised embeddings. Evaluation results are given in §VII, before §VIII reviews the related work and §IX concludes the paper.

II. Model and Problem Statement

A. Attributed network

An attributed network (or graph) is a data structure that can be represented as $G = (V, A, F)$, where V is a set of nodes. $A \in \{0,1\}^{n\times n}$ is an adjacency matrix representing connections/edges in a graph; i.e. $A(u,v) = 1$ if there is an edge from u to v, and $A(u,v) = 0$ otherwise. $F \in \mathbb{R}^{n\times m}$ is a node attribute matrix, assuming the number of nodes is n and each node has an m-dimensional attribute. Each row in the attribute matrix F encodes the semantics of the respective node (e.g., age and marital status of users in social networks or category of protein in protein-protein interaction networks). Note that these are attributes that originate from the application domain. They are not synthetic features produced by embedding techniques or hand-crafted by experts. Hence, they do not include any network topology information.

TABLE I: Notation Summary

Symbols	Definition
$G = (V, A, F)$	an attributed network
$\hat{A}$	adjacency matrix with self loop
$\hat{\mathbf{N}}(v)$	set of neighbors of node v include itself
$\hat{D}$	diagonal degree matrix with $\hat{D}_{i,i} = \sum_j \hat{A}_{i,j}$
k	number of GCN layers
$d^{(l)}$	embedding dimension at l-th layer
$W^{(l)}$	weight matrix at layer l
$H^{(l)}$	node embeddings at layer l
S	alignment matrix

B. Network alignment

Network alignment is the task of identifying corresponding nodes between two different networks. Without loss of generality, we select one network as the source network, G_s, and the other one as the target network, G_t. For each node in the source network, we aim to recognize, if there are any, its counterparts in the target network.

Alignment matrix. Network alignment techniques calculate an alignment matrix $\mathbf{S} \in \mathbb{R}^{n_1\times n_2}$, where n_1 and n_2 are the numbers of nodes in G_s and G_t, respectively. $\mathbf{S}(v, v')$ represents the matching degree between $v \in V_s$ and $v' \in V_t$.

Anchor links. Given two networks $G_s = (V_s, A_s, F_s)$ and $G_t = (V_t, A_t, F_t)$, a node pair (v, v'), with $v \in V_s, v' \in V_t$, is an anchor link, if v is aligned to v'. We refer to v and v' as anchor nodes. The goal of network alignment is to produce all potential anchor links, given that little or no ground-truth information on anchor links is available.

Problem Definition (Alignment matrix computation) *Given two attributed networks $G_s = (V_s, A_s, F_s)$ and $G_t = (V_t, A_t, F_t)$, the problem of network alignment is to calculate an alignment matrix $\mathbf{S}$ where $\mathbf{S}(v, v')$ represents the matching degree between a node $v \in V_s$ and $v' \in V_t$.*

This above formulation has two advantages. (i) it enables *scalable* calculation, since anchor links can be inferred from the alignment matrix through ranking rules [16, 20]; and (ii) it provides *flexibility* since node pairings can be one-to-many, which is important for differently sized networks [18, 25, 34].

C. Alignment constraints

Existing works observed two types of constraints that should hold for anchor links for good alignment performance [16, 39].

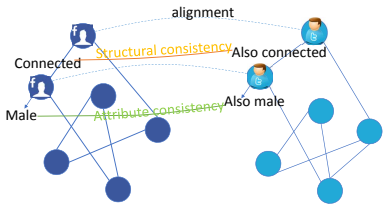


Fig. 1: Structural and attribute consistency example

Structural consistency. This constraint is often referred as the *homophily rule*: If two nodes have close relation in the topology in one network, their anchor nodes in the other network shall maintain this relation [24]. For example, if two persons are friends in one social network, they are likely to connect in another social network (Fig. 1). Mathematically, if v, u are close neighbours in G_s and (u, u') and (v, v') are anchor links, then v', u' shall be close in G_t.

Attribute consistency. This constraint states that corresponding nodes shall share the same attribute values [16, 39]. For example, the profile information (e.g., age, email address) of a single person in different social networks shall be similar (Fig. 1). Formally, if (v, v') is an anchor link, then it should hold that $F_s(v) = F_t(v')$ (implying $|F_s(v)| = |F_t(v')|$).

Most existing alignment techniques cannot satisfy these two consistency constraints at the same time, though, due to their differences in modality [39]. While there are a few notable exceptions, they still consider both types of consistencies in separate steps [16]. In this paper, we show theoretically

and empirically that the properties of GCN can be used to enforce structural consistency and attribute consistency simultaneously. Moreover, we show that, in practice, these consistency constraints may sometimes be violated, so that the model shall be made robust against minor violations.

III. Approach Overview

A. Motivation

From characteristics of real-world datasets [19] and empirical results of the state-of-the-art [16, 21, 39], we argue that a high-quality network alignment solution should respect:

(R1) *Consistency:* Structural consistency and attribute consistency shall be respected since these constraints help to find true anchor links [16, 39]. False positives could hamper downstream applications, such as personalized advertisement and friend suggestion. For our setting (§II-B), this also eliminates false negatives since the output must compute an anchor for every node.

(R2) *Adaptivity:* Performance of alignment techniques often degrades when consistency constraints are violated. In practice, this is often the case, e.g., topology perturbations that violate structural consistency are common in social networks. Attribute noise is common as well, e.g. social network users can register multiple accounts with different or missing attribute values due to typos, asynchronicity, laziness, privacy, or platform incompatibility.

(R3) *Unsupervised:* Many existing alignment techniques require large training data. However, such ground-truth information is rarely available in real-world networks due to high labelling costs [29]. For instance, checking users' background and pairing their accounts manually in social networks is very time-consuming. Hence, unsupervised network alignment is desirable.

Several challenges need to be overcome to satisfy these guidelines. First, learning of an alignment model has to alleviate the presence of structural and attribute noise, although the target network is a permuted version of the source network in ideal cases [21]. Second, existing embedding-based techniques often vectorize the two networks independently and create even more noises since the vectorization is inherently imperfect [4]. They fix the issue by an additional vector-space reconciliation step, which is however still an approximate process [35]. Third, unsupervised learning is typically more prone to noise than supervised learning [36]. However, data augmentation methods to achieve such adaptability over unlabelled graph data the context of network alignment is still missing.

B. Solution Sketch

In this work, we go beyond the state-of-the-art by developing an end-to-end network alignment framework, where the source and the target networks are embedded by a GCN-based model while satisfying consistency constraints simultaneously. We provide theoretical motivations (§IV) of using GCN-based model to unify the attribute and structural information of the network nodes by vector representations, where the vector-similarity of two anchor nodes depend on both the topological and semantic properties. Moreover, we design a consistency loss to guarantee that the learnt embeddings satisfying (R1).

Thanks to GCN properties, we show that the network permutation does not affect the embedding of each network node in §IV. Based on this observation, we develop a data augmentation method for graph data by adding structural and attribute noises to the original networks. Then, we design an adaptivity loss, which measures the distance between the embeddings of original network and its augmented versions, to make sure that the alignment can be fully unsupervised (R3) while being robust to noises (R2).

We train the same GCN-based model for both source and target networks by a weight-sharing mechanism. This allows their embeddings to be in a common space without any labelled data, avoiding the reconciliation step and maximising the noise tolerance of the model, satisfying (R3).

C. End-to-end network alignment framework

Fig. 2 presents an overview of our model. We first forward the source and target network through a GCN-based model to embed network nodes to feature spaces, in which embeddings are hidden features capturing all structural and attribute information. With this structure, our model requires the realisation of the following functionalities:

Multi-order embedding. In this step, we design a specific GCN-based model for learning the representations of network nodes. The model contains several layers; each layer learns hidden features at a different order of neighbourhood structure for network nodes (deeper layer, larger neighbourhood). The model is guaranteed to satisfy consistency constraints via a *consistency* loss function. The detailed process of constructing these multi-order representations is described in §V.

Data augmenter. We utilize data augmentation to improve the adaptivity of the model, thus boosting the performance of our unsupervised network alignment. More precisely, we add small perturbations to the original network to simulate structural noises and attribute noises. An *adaptivity* loss function is defined to minimize the difference between the embeddings of the original network and its perturbed versions. Details of this functionality can be found in §V.

Alignment Instantiation. Instead of using only the last-layer embedding as in existing works, we employ the features at all layers to determine the alignment result. More precisely, we first calculate the layer-wise alignment matrix for each GCN layer and then we fuse these alignment matrices by a weighted-sum that characterises the importance of each layer into the final alignment output. The detail is described in §VI.

Alignment Refinement. Before producing the final alignment matrix, we develop an iterative refinement process to avoid the effects of real-world noises. First, we determine the stable nodes (nodes with no noises) by comparing the current embeddings across GCN layers. Then, using the current alignment output as a reference, we adjust the embeddings by amplifying the influence of stable nodes over time. Details are in §VI.

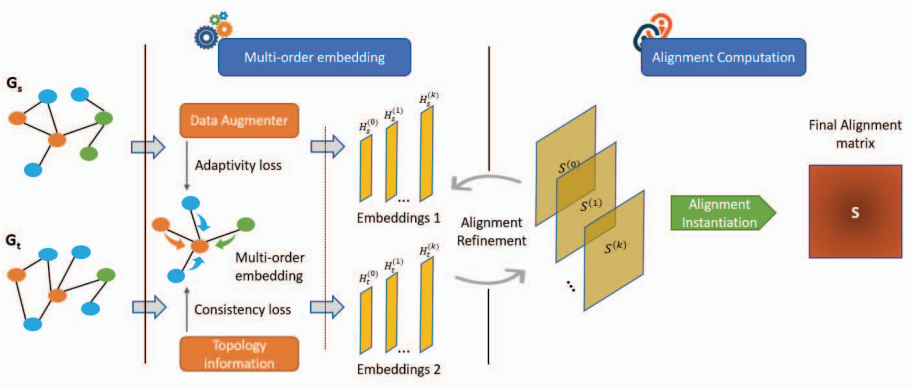


Fig. 2: Overview of GAlign framework

IV. GCN FOR CONSISTENT NETWORK ALIGNMENT

In this section, we provide theoretical motivations to design a specific GCN model for network alignment, in which GCN properties guarantee alignment consistency constraints.

A. GCN Model

GCN is a deep neural network model of k layers, denoted as $\langle f, W\rangle$ where $W = \{W^{(1)}, \ldots, W^{(k)}\}$, that allows end-to-end representation learning of graph nodes [5]. Given node attributes and the structure of a network G, GCN encodes simultaneously these two type of information into the hidden features at each layer, which can be used as individual information of the network nodes. Formally, the feed-forward pass is: $f^{(k)}(f^{(k-1)}(\ldots f^{(1)}(G)\ldots))$, where the layer l produces the embedding $H^{(l)} = f^{(l)}(.)$. GCN employs a neighborhood aggregation scheme $f(.)$ across all layers: the hidden features of the previous layer construct the features of the next layer:

$$H^{(l)} = f(G, H^{(l-1)}, W^{(l)}) = \sigma(CH^{(l-1)}W^{(l)}) \quad (1)$$

where $1 \leq l \leq k$, $C = \hat{D}^{-\frac{1}{2}}\hat{A}\hat{D}^{-\frac{1}{2}}$ is the normalized Laplacian matrix, $H^{(l)}$ is the $|V| \times d^{(l)}$ embedding matrix ($H^{(0)} = F$), $d^{(l)}$ is the embedding dimension at layer l ($d^{(0)} = m$), $\hat{A} = A + I_n$ is the adjacency matrix with self-connections; I_n is the identity matrix, $\hat{D}$ is a diagonal matrix that $\hat{D}_{i,i} = \sum_j \hat{A}_{i,j}$, $W^{(l)} \in \mathbb{R}^{d^{(l-1)} \times d^{(l)}}$ is a trainable weight matrix at layer l, and $\sigma(.)$ is an activation function. $ReLU$ is not suitable for alignment task since it does not distinguish negative values from positive values (i.e. $ReLU(-x) = ReLU(x)$). Since activate function has to be bijective, we use $tanh$ to avoid the loss of information.

B. Permutation Immunity

The target network can be modeled as a permuted version of the source network [21]. Given the adjacency matrices of source network A_s and the target network A_t, these works aim to find the permutation matrix P^* that minimize the cost function f_{cost} which captures the consistency constraints:

$$P^* = \min_P f_{cost}(P) = \min_P ||PA_sP^T - A_t||_F \quad (2)$$

However, this approach is prone to consistency noise and restricted to 1-1 alignment. We show that using GCN can achieve similar or better efficiency with more flexibility.

Proposition 1 *Embedding generated by GCN is immune to the permutation factor. Given $A_t = PA_sP^T$ for a particular permutation matrix P, we prove that $H_t^{(l)} = PH_s^{(l)}$, where $H_s^{(l)}$ and $H_t^{(l)}$ are the embedding matrices at the layer l of the model of source and target network, respectively.*

Proof: First, we already have $H_t^{(0)} = PH_s^{(0)}$, as the embeddings at the first layer of network nodes are their original attribute. Suppose that for layer l, $H_t^{(l)} = PH_s^{(l)}$ is true, we will prove that $H_t^{(l+1)} = PH_s^{(l+1)}$. We have:

$$H_t^{(l+1)} = \sigma(\hat{D}_t^{-\frac{1}{2}}\hat{A}_t\hat{D}_t^{-\frac{1}{2}}H_t^{(l)}W^{(l)}) \quad (3)$$

Because of permutation assumption, we have $A_t = PA_sP^T$ and $D_t = PD_sP^T$. The Eq. 3 is then equivalent to:

$$\begin{aligned} H_t^{(l+1)} &= \sigma[(P\hat{D}_sP^T)^{-\frac{1}{2}}(P\hat{A}_sP^T)(P\hat{D}_sP^T)^{-\frac{1}{2}}PH_s^{(l)}W^{(l)}] \\ &= \sigma(P\hat{D}_s^{-\frac{1}{2}}P^TP\hat{A}_sP^TP\hat{D}_s^{-\frac{1}{2}}P^TPH_s^{(l)}W^{(l)}) \quad (4) \end{aligned}$$

As P is an orthogonal matrix, $PP^T = P^TP = I$. We can rewrite Eq. 4 as: $H_t^{(l+1)} = \sigma[P(\hat{D}_s^{-\frac{1}{2}}\hat{A}_s\hat{D}_s^{-\frac{1}{2}}H_s^{(l)}W^{(l)})] = P\sigma(\hat{D}_s^{-\frac{1}{2}}\hat{A}_s\hat{D}_s^{-\frac{1}{2}}H_s^{(l)}W^{(l)}) = PH_s^{(l+1)}$ ■

C. From consistency to embedding

We prove that GCN is not only immune to permutation but also naturally supports the structural and attribute consistency constraints. In other words, two anchor nodes which share the same attribute and topology information (i.e. node degree, neighbours) will have same embeddings at every layer.

Proposition 2 *If two nodes $v \in G_s$, $v' \in G_t$ have same degree ($deg(v) = deg(v')$) and there exists a matching $M = \{(t, t')|t \in \hat{\mathbf{N}}_s(v), t' \in \hat{\mathbf{N}}_t(v')\}$ between the neighbor sets of two nodes such that for every matched pairs (t, t'):*

- *Degree is equal: $deg(t) = deg(t')$*
- *Embedding at the layer l is equal: $H_s^{(l)}(t) = H_t^{(l)}(t')$*

Then embeddings of v and v' at layer $l+1$ are the same, which means: $H_s^{(l+1)}(t) = H_t^{(l+1)}(t')$.

Proof: At layer $(l+1)$ the embedding of v and v' is:

$$H_s^{(l+1)}(v) = \sigma\big(g(v) \sum_{t \in \hat{N}_s(v)} g(t) H_s^{(l)}(t) W^{(l)}\big) \quad (5)$$

$$H_t^{(l+1)}(v') = \sigma\big(g(v') \sum_{t' \in \hat{N}_s(v')} g(t') H_t^{(l)}(t') W^{(l)}\big) \quad (6)$$

where $g(t) = deg(t)^{-\frac{1}{2}}$. Since $H_s^{(l)}(t) = H_t^{(l)}(t')$ and $g(t) = g(t')$, we have $H_s^{(l+1)}(v) = H_t^{(l+1)}(v')$. ■

V. Multi-order Embedding

In this section, we design a specific GCN model to embed the nodes of the source and target networks into a unified vector space, in which the embeddings of true anchor nodes are similar, allowing us to retrieve the alignment result efficiently.

A. GCN-based embedding model

The model contains k layers; each layer learns hidden features at a different order of neighbourhood structure for each network node. Starting from the lowest embedding $H^{(0)}$ that presents the node attributes, the model iteratively aggregate the hidden feature of every node in one layer with features of its adjacent nodes to produce the hidden feature for this node in the next layer $H^{(1)}$, and so on (see Eq. 1). This message-passing scheme allows the model to unify attribute information with topology information at different orders [8].

Existing works use only the embeddings in the final layer $H^{(k)}$ of GCN for representing the network nodes [4]. However, there is a trade-off: deeper layers contain more structural information but prone to structural noises and vice-versa (shallower layers focus on more attribute consistency but prone to attribute noises). Indeed, the hidden features $H^{(l)}$ can be considered as the collective information of l-hop neighbourhood of nodes [5]; and thus, a larger neighbourhood would hinder individual information of each node.

For this reason, we leverage embeddings at all layers. Formally, each node v is associated with a set of embeddings $\{H^{(0)}(v), \ldots, H^{(k)}(v)\}$, or so-called multi-order features. Using all layers leverage the structural information locally and globally (from short-range to long-range neighbourhood). However, the number of GCN layers k is still a hyperparameter and we show how to choose a suitable k in the experiments.

B. Consistency loss

To make the embedding satisfy the consistency constraints (see R1), we design a *consistency* loss function that encourages the nodes that have similar neighbourhood structure to have similar embeddings while making those of unrelated nodes highly distinctive from each other. On the one hand, lower-order embeddings (shallower layers) capture local information of a node such as neighbourhood structure. However, since there could be many nodes with the same local information in a network, the lower-order embeddings of different nodes are often similar even if they are far away from each other or, making the alignment difficult. On the other hand, high-order embeddings might collapse the embedding distribution. Embedding at deeper layers can be seen as the result of averaging large number of nodes into one vector, leading to similar embeddings due to the law of large number. Nevertheless, taking neighbours within a large number of hops means the information of many nodes can appear in each other's context, pulling their feature vectors closer in the embedding space. Balancing this trade-off, we compute the loss function from the embeddings at all layers to complement each other:

$$J_c(G) = \sum_{l \in [1 \ldots k]} ||\hat{D}^{-\frac{1}{2}} \hat{A} \hat{D}^{-\frac{1}{2}} - H^{(l)} {H^{(l)}}^T||_F \quad (7)$$

where $||.||_F$ denotes the Frobenius norm. We use normalized Laplacian matrix instead of adjacency matrix with the aim to enrich the embeddings with more topology information, which avoids collapsing the embedding space.

C. Adaptivity loss

Perturbation-based Network Augmentation. We utilize data augmentation to enforce the adaptivity of the model with the noises. Intuitively, violations of consistency constraints are only caused by local differences (noises) in the network topology or node attributes. For example, two friends in a social network might not be friends in another social network. We follow a perturbation-based augmentation approach [36] to simulate structural and attribute violations. The original network is injected with small perturbations by adding or removing edges randomly (structural noises) and changing the node attributes randomly (attribute noises). This allows the model to be robust for various types of noises, as they can appear in an unpredictable manner, by enforcing the model to perform good alignments for all noisy versions.

Formally, starting from an original network $G = (V, A, F)$ (source or target network), we produce an augmented network $G_p = (V_p, A_p, F_p)$ with adjacency matrix A_p by:

$$A_p = PAP^T \quad (8)$$

where P is a random permutation matrix, with $P_{ij} = 1$ means that node i from the original network corresponds to node j in the augmented network, otherwise $P_{ij} = 0$.

On the one hand, we add structural noises to $\mathbf{G}_p$ by removing or adding edges with probability p_s (via element-wise multiplication of adjacency matrix with a zero-mask matrix). On the other hand, we add attribute noises to a random node v depending on the type of attributes. For binary attributes, we randomly change the position of non-zero entries of each attribute vector F_i with probability p_a. For real-value attributes, we adjust each element F_{ij} in each attribute vector F_i by a random amount in the range $[0, p_a * F_{ij}]$.

Noise-adaptive loss function. Intuitively, the source network should align with both the target network and its perturbed versions, and vice-versa. This helps the alignment to be robust to consistency violations (see R2). To this end, we design a *noise-adaptive* loss function that minimizes the difference

between the multi-order features of the network nodes before and after perturbation. Formally, the adaptivity loss is:

$$J_a(G, G^*) = \sum_{v \in G, v^* \in G^*} \sum_{l \in 1..k} \sigma_<(||H^{(l)}(v) - H^{(l)}(v^*)||_F) \tag{9}$$

where v and v^* are the corresponding nodes in original network G and augmented network G^*. Here, we apply a confidence-based mechanism to mask out the cases when the perturbation becomes uncontrollable. In particular, $\sigma_<(x)$ is an activation function, which is equal to x if x is less than a perturbation threshold and 0 otherwise. This is because large perturbation could destroy the neighbourhood structure of a node; and thus, create undesirable embedding differences that might affect the sanity of the whole model.

D. Put It Altogether

Finally, we combine two loss functions as follow:

$$J(G) = \gamma J_c(G) + (1 - \gamma) \sum_{G^*} J_a(G, G^*) \tag{10}$$

where γ is a hyper-parameter used to balance the losses.

Alg. 1 illustrates the whole multi-order feature construction process. The core idea of our training algorithm is the weight-sharing mechanism, in which the GCNs of the source network, the target network, and the augmented networks use the same weight matrix $W^{(l)}$ at every layer l. Otherwise, the learnt embeddings of different networks will end-up in different embedding spaces, breaking the alignment purposes in the first place. Moreover, the weight-sharing mechanism guarantees that the consistency constraints are satisfied, as proven in §IV.

Algorithm 1 Augmented learning for multi-order embedding

Input: Source network G_s, target network G_t
Output: Multi-order node embeddings of G_s and G_t
Construct GCN model $\langle f, W \rangle$
Construct augment graphs $\{G^*\}_s$ for G_s
Construct augment graphs $\{G^*\}_t$ for G_t
for $G = (V, A, F)$ in $\{G_s, G_t\} \cup \{G^*\}_s \cup \{G^*\}_t$ **do**
    Initialize layer-0 embeddings $H_G^{(0)} = F$
for some epochs **do**
    for G in $\{G_s, G_t\} \cup \{G^*\}_s \cup \{G^*\}_t$ **do**
        Compute multi-order embeddings $H_G^{(l)} s = f(G, H_G^{(l-1)}, W)$ for $1 \leq l \leq k$ by Eq. 1
    for G in $\{G_s, G_t\}$ **do**
        Compute the loss function $J(G)$ by Eq. 10
        Update parameters W by back propagation
return H_s and H_t

VI. Alignment Computation

In this section, we show how to exploit the learnt multi-order embeddings to compute the alignment matrix effectively.

A. Alignment Instantiation

As aforementioned, the embeddings at different GCN layers represent feature information of each node at different topological order. Shallow layers tend to handle local neighbourhood information, while deeper layers tend to capture global network topology. Combining all layers simultaneously is beneficial as true anchor nodes should have not only similar local features but also similar global features.

However, there are issues to implement this idea. First, the layers could be not equally important. For example, the first layer carries neighbourhood structure of each node, but easily suffer from structural noises. Deeper layers are more robust to structural noises since they consider a larger range of topology information, but might be less useful for aligning individual nodes. Second, the node features of the source network and target network might not have the same embedding space. Existing alignment frameworks often perform an additional reconciliation step, which however still have discrepancy and cause instability of alignment output [23, 24, 35, 41].

Layer-wise alignment matrix. To mitigate these issues, we first define the layer-wise alignment matrix aggregated from the embeddings at each layer $l \in [0, k]$:

$$S^{(l)} = H_s^{(l)} {H_t^{(l)}}^T \tag{11}$$

As the weights of the GCN models are shared between the two networks, their embedding spaces are the same and thus their alignment can be measured directly by the analogy of their embeddings. Each layer-wise alignment matrix represents the alignment scores for all pairs of nodes between the source and target networks. In other words, it captures, for each node, the alignment candidates according to all embedded structural and attribute information at the current layer.

Aggregated alignment matrix. We aggregate all layer-wise alignment matrices into a single alignment matrix:

$$\mathbf{S} = \sum_{l=0}^{k} \theta^{(l)} S^{(l)} \tag{12}$$

where $\theta^{(l)}$ is the importance of layer l and we treat them as hyper-parameters. For one-to-one network alignment setting, the anchor links can be instantiated straightforwardly by taking the top-1 target node (highest alignment score) for each source node. Other alignment settings such as one-to-many can be instantiated as well, but out of the scope of our paper.

B. Alignment Refinement

Another issue when computing alignment from embeddings is the effect of adversarial conditions (or noises, for short), which refers to structural and attribute noises as well as constraint violations and graph size imbalance. Intuitively, nodes with noises will bring instability in their embeddings over neighbouring nodes. As a result, noises will make the embeddings of true anchor nodes different, leading to incorrect alignment scores. However, the vice-versa is not true since the embedding difference is only caused by noises and we do not know the true alignment before-hand. We design an iterative refinement process to detect and mitigate the effects of noises on the aggregation of embeddings into alignment matrix.

Stability of embeddings. The first step is to evaluate the stability of node embeddings. As aforementioned in §IV-B, the target network can be considered as a permuted version of the

source network with the noises added. Since GCN embedding is immune to permutation (Prop. 1), the embedding difference between two true anchor nodes is only caused by consistency noises. Thanks to this property, we define stability as follows. Given a node v and their corresponding embeddings at all layers $H^{(1)}, \ldots, H^{(k)}$, we want to determine if node v is a stable node or not. A node $v \in V_s$ is defined as stable if its anchor nodes (highest alignment scores of v) in all layer-wise alignment matrices are the same and the corresponding alignment scores must be higher than a confident factor λ:

$$\begin{cases} \text{argmax}_{u' \in V_t} S^{(l)}(v, u') = v' & \forall 0 \leq l \leq k \\ \max_{u' \in V_t} S^{(l)}(v, u') > \lambda & \forall 0 \leq l \leq k \end{cases} \tag{13}$$

The motivation behind this formulation is that if a node is stable, its anchor node should be stable as well. In other words, a pair of anchor nodes is stable if they have similar embeddings at any layer of the model. On the other hand, nodes that do not satisfy Eq. 13 are called unstable nodes.

Noise-aware propagation of embeddings. In terms of GCN aggregation, we do not want unstable nodes to aggregate their information into neighbouring nodes. We propose a weighted propagation mechanism for GCN embeddings such that more stable nodes have greater influence and vice-versa:

$$AGG_w\big(\mathbf{X}^{(l)}\big) = \alpha(v) g(v) \sum_{t \in \hat{\mathbf{N}}(v)} \alpha(t) g(t) H^{(l-1)}(t) W^{(l)}$$

where $\mathbf{X}^{(l)} = \{H^{(l)}(t), \forall t \in \hat{\mathbf{N}}(v)\}$ and $g(t) = deg(t)^{-\frac{1}{2}}$. $\alpha(v)$ is the influence factor of node v. Intuitively, if v is a stable node, its influence should be intensified:

$$\alpha(v) = \beta \times \alpha(v) \tag{14}$$

where $\beta > 1$ is an accumulation constant. Put it altogether, the new aggregation rule of GCN is:

$$H^{(l+1)} = \sigma(\hat{D_q}^{-\frac{1}{2}} \hat{A} \hat{D_q}^{-\frac{1}{2}} H^{(l)} W^{(l)}) \tag{15}$$

where $\hat{D_q} = \hat{D}Q$ and Q is a diagonal matrix ($Q(v, v) = \alpha(v)$).

Alignment Algorithm. The whole process of alignment algorithm is put altogether as Alg. 2. At the beginning, we aggregate the embeddings into layer-wise alignment matrices as in §VI-A. Then, the alignment is refined by a searching strategy. First, we initialise the influence factor of each node to 1. Next, an iterative process is performed. On the one hand, we use the current alignment matrices as reference anchor links (highest alignment scores) to detect stable nodes. The influence factors of those nodes in turn are increased. On the other hand, those factors are used to refine the current embeddings, which is then aggregated into new alignment matrices. However, the search space of this iterative process is not uniform since there is no supervision information. We employ a greedy selection based on the criterion that true anchor nodes should have embeddings close to each other as much as possible. More precisely, we keep track the sum of top-1 alignment scores for each refined alignment matrix, i.e. $g(\mathbf{S}) = \sum_{v \in V_s} max(\mathbf{S}(v))$, and return the one with best score in the end.

Algorithm 2 Alignment computation with stability refinement

Input: G_s and G_t: source and target graphs
  Source embeddings $H_s^{(0)}, \ldots, H_s^{(k)}$
  Target embeddings $H_t^{(0)}, \ldots, H_t^{(k)}$
Output: Fine-tuned alignment matrix $\mathbf{S}$
Compute layer-wise alignment matrices $\{S^{(l)}\}_{l=0}^{k}$ from embeddings $\{H_s^{(l)}, H_t^{(l)}\}_{l=0}^{k}$ by Eq. 11
$Q_s^{init} = I_{[n_s \times n_s]}$, $Q_t^{init} = I_{[n_t \times n_t]}$ ▷ Initialize all influence factors to 1
Initialize $g_{best} = -1$, $\mathbf{S}_{best} = 0_{[n_s \times n_t]}$
for some iterations **do**
  Find stable nodes $\mathbf{U}_s$ and $\mathbf{U}_t$ using Eq. 13
  for $v \in \mathbf{U}_s$, $v' \in \mathbf{U}_t$ **do**
    Update $Q_s(v, v)$ and $Q_t(v', v')$ using Eq. 14
  Update embeddings $\{H_s^{(l)}, H_t^{(l)}\}_{l=1}^{k}$ using Eq. 15
  Update $\{S^{(l)}\}_{l=0}^{k}$ from $\{H_s^{(l)}, H_t^{(l)}\}_{l=0}^{k}$ using Eq. 11
  Compute aggregated alignment matrix $\mathbf{S}$ by Eq. 12
  if $g(\mathbf{S}) = \sum_{v \in V_s} max(\mathbf{S}(v)) > g_{best}$ **then**
    $g_{best} = g(\mathbf{S})$ $\mathbf{S}_{best} = \mathbf{S}$
return $\mathbf{S}_{best}$

C. Complexity Analysis

Without loss of generality, we rely on: n as the number of nodes, d as the number of feature dimensions, and e as the number of edges (non-zero entries of adjacency matrix).

Time Complexity. There are two steps to analyse:

- *Multi-order embedding:* the normalized Laplacian matrix C is computed once. Since the adjacency matrix $\hat{A}$ is sparse and $\hat{D}$ is diagonal matrix, the time complexity for calculating C is $\mathcal{O}(e)$. Therefore, the propagation of k GCN layers takes $\mathcal{O}(k(ed + nd^2)) = \mathcal{O}(ed + nd^2)$.
- *Alignment computation:* first the refinement process relies on the forward pass, which takes $\mathcal{O}(ed + nd^2)$ as above. Second, finding stables nodes take $\mathcal{O}(kn^2)$. Updating the influence factors of stable nodes in worst case is $\mathcal{O}(n)$ (all nodes are stable).

Therefore, the total time complexity is $O(ed + n(d^2 + 1))$.

Space Complexity. There are also two steps to analyse:

- *Multi-order embedding:* First, we need to store embeddings at every layer, which takes $\mathcal{O}(knd)$. Second, we need to store trainable parameters at every layer, which takes $\mathcal{O}(kd^2)$. Third, the normalized sparse Laplacian matrix C takes $\mathcal{O}(e)$ space complexity. Put it all together, the space complexity of embedding step is $\mathcal{O}(nd+d^2+e)$.
- *Alignment computation:* We do not need to store the whole alignment matrix $\mathbf{S}$ in memory. All operations we use on $\mathbf{S}$ is finding stable nodes, which can be done by separately iterating the rows of $\mathbf{S}$. In other words, we only need to compute and store one-row vector of $\mathbf{S}$ from the embeddings at each iteration, which takes $\mathcal{O}(n)$ space.

In sum, the total space complexity is $\mathcal{O}(n(d+1) + d^2 + e)$.

VII. Experimental evaluation

Our experiments answer the following research questions:
(RQ1) Does our model outperform the baseline methods?
(RQ2) How does each component of our model matter?
(RQ3) Is our model adaptive to adversarial conditions?

(RQ4) Is our model sensitive to hyper-parameters?
(RQ5) Can our technique be interpreted qualitatively?

In the remainder, we first describe our experimental setting (§VII-A). Then we present our empirical evaluations, including end-to-end comparison (§VII-B), adaptivity to adversarial conditions (§VII-D), and hyperparameter sensitivity (§VII-E).

A. Experimental setup

Real datasets. We use three state-of-the-art alignment datasets of 6 real-world networks with different domains.

- *Douban Online vs Douban offline:* Douban network is a Chinese social network with nodes as users and edges as friendships [38], containing 1118 anchor links [39].
- *Flickr vs Myspace:* The alignment ground-truth between subnetworks of Flickr and Myspace is extracted and validated by [39], containing 323 anchor links.
- *Allmovie vs Imdb.* Allmovie network is constructed from Rotten Tomatoes website[1]. Two films have an edge connecting them if they have at least one common actors. *Imdb* network is constructed in a similar way from Imdb website[2]. The alignment output is constructed by the identity of the film, containing 5176 anchor links.

TABLE II: Statistics of real-world networks

Networks	#Nodes	#Edges	#Attributes
Douban Online	3906	8164	538
Douban Offline	1118	1511	538
Flickr	5740	8977	3
Myspace	4504	5507	3
Allmovie	6011	124709	14
Tmdb	5713	119073	14
Bn	1781	9016	20
Econ	1258	7619	20
Email	1133	5451	20

Synthetic data. We further synthesise alignment data to comprehensively evaluate adversarial conditions such as noises or graph size imbalance. The adversary level is controlled by a synthesis procedure similar to §V-C. For an original network, we generate a noisy version and perform network alignment between the two versions (node identity is preserved, indicating the alignment ground-truth). We apply the following real-world networks as they do not have alignment data yet.

- *bn:* represents a part of brain structure [1]. Each node of the network depicts a brain voxel, and the edge represent one fibre tract that connects two voxels.
- *econ:* represents an economic model of Victoria state, Australia during the banking crisis in 1880 [30]. The nodes represent the organisations located in the state (e.g. firms, banks), and the edge represent the contractual relationships between them.
- *email:* is generated from email data of European universities [30]. The nodes represent the email addresses, and the edges between the any two addresses are formed if they both sent and received emails from each other.

Table II summarises the networks used for alignment.

[1]https://www.kaggle.com/ayushkalla1/rotten-tomatoes-movie-database
[2]https://www.kaggle.com/jyoti1706/imdbmoviesdataset

Baseline methods. We study five representative methods:

1) *REGAL:* is a spectral method which models alignment matrix by topology and nodes' feature similarity then employs low-rank matrix approximation speed-up [16].
2) *IsoRank:* is a spectral approach which propagates the pair-wise node similarity over the network with the homophily principle assumption, which states that two corresponding nodes in two networks connect to similar characteristic neighbours [32].
3) *FINAL:* is a spectral technique which defines a model with three criteria, namely structure consistency, node feature consistency and edge feature consistency to tackle alignment problem on attributed networks [39].
4) *PALE:* is an embedding-based technique which learns nodes embedding by maximising the co-occurrence likelihood of edge's vertices then applies linear or multi-layer perceptron (MLP) as mapping function [24].
5) *CENALP:* is an embedding-based model that unifies network alignment and link prediction tasks in a unified model, which first leverages a tailored biased random-walk strategy across the networks, then learns nodes embedding by maximising the co-occurrence likelihood of nodes within the walks [7].

It is worth noting some baselines require supervision data in the form of *prior alignment matrix* (*FINAL* and *IsoRank*) and partial ground truth to reconcile embedding spaces (*PALE* and *CENALP*). To respect their original settings (even though our model is handicapping), we use 10% of ground truth for training (*PALE* and *CENALP*) and generating the *prior alignment matrix* if it is not available (*FINAL* and *IsoRank*).

Metrics. We evaluate network alignment with state-of-the-art metrics in both *prediction* perspective and *ranking* perspective [31]. For *prediction* perspective, we employ $Success@q$ (aka $Accuracy@q$ [39]), which indicates whether the true positive match occurs in top-q candidates. More precisely, for each anchor pair (v_s^*, v_t^*) in the ground-truth, if the alignment score $\mathbf{S}(v_s^*, v_t^*)$ is within the q-highest values in the row $\mathbf{S}(v_s^*)$ of the alignment matrix $\mathbf{S}$, the alignment output for node v_s^* is recorded as a successful case:

$$Success@q = \frac{\sum_{v_s^* \in V_s} \mathbb{1}_{\mathbf{S}(v_s^*, v_t^*) \in \text{top-}q\ \mathbf{S}(v_s^*)}}{\#\{\text{True anchor links}\}} \tag{16}$$

For *ranking* perspective, we use Mean Average Precision [24] (aka Mean Reciprocal Rank under pair-wise setting):

$$MAP = mean\left(\frac{1}{\text{ra}}\right) \tag{17}$$

where ra is the rank of true anchor target in the sorted list of anchor candidates. Another popular ranking metric is AUC, which reflects the trade-off between precision and recall. In network alignment setting where the output must yield an anchor link for every node, AUC can be simplified to [31]:

$$AUC = \frac{\#\{\text{Negative match}\} + 1 - ra}{\#\{\text{Negative match}\}} \tag{18}$$

Hyperparameter tuning. If not stated otherwise, the hyper-parameters are set as follows: $\gamma = 0.8$ (balancing factor between consistency loss and adaptivity loss), $\beta = 1.1$ (accumulation factor of stable nodes), $\lambda = 0.94$ (embedding stability threshold), $k = 2$ (number of GCN layers), the importance weights $\theta^{(l)}$ of all GCN layers are equal ($= \frac{1}{k+1}$). Embedding size of all GCN layers $d^{(l)} = 200$. On the other hand, we also study hyper-parameter sensitivity in §VII-E. For baseline methods, we report the best performance following the parameter tuning of their original papers.

Reproducibility environment. The results are averaged over 50 runs to mitigate randomness. All experiments are conducted on an AMD Ryzen ThreadRipper 3.8 GHz system with 64 GB of main memory and four GTX Titan X graphic cards. We use Pytorch for implementation and Adam as gradient optimiser.

B. End-to-end comparison

Effectiveness. Table III answers (RQ1) by showing the end-to-end comparison of our alignment model against baseline methods. In general, our model outperforms all the baselines across all datasets in terms of *MAP*, *AUC*, and *Success@1* metrics. FINAL is the best method among the baselines and emerges to compete our model in Allmovie-IMDB dataset in terms of *Success@10*. This is because FINAL is a state-of-the-art method that, similar to us, takes into account structural information and node attribute information. Moreover, FINAL also considers the supervision data in the form of prior alignment matrix, while ours do not examine any.

A key finding is that alignment methods ill-perform significantly in Flickr - Myspace dataset. This could be explained by the fact that the networks are sparse with average degree less than five, making the structural consistency often being violated. Methods that strictly follow this consistency constraint would fail in such cases. For example, it is normal for two peoples to be friends on Facebook but not connected to each other on Twitter; or one person could use different profile attributes for different social network accounts.

Efficiency. The running time of alignment techniques is shown in Table III. *REGAL* is the fastest because of low-rank matrix approximation. Our model stays in top-3 with less than 6 min.

C. Ablation test

We answer (RQ2) by comparing our model and its variants:

- *GAlign-1:* No data augmentation for adaptivity. The loss function contains only the consistency term (Eq. 7).
- *GAlign-2:* The refinement step (§VI-B) is removed. The learned multi-order embeddings are used directly to compute the alignment matrix (§VI-A).
- *GAlign-3:* Only the embeddings at the final layer of GCN is used (as in traditional works) instead of using the multi-order embeddings (§VI-A).

Table IV presents the result with only important metrics and datasets due to space limitation. It can be seen that our original model *GAlign* outperforms other variants, which proves the importance of our data augmentation and refinement mechanisms. In particular, *GAlign* has *Success@1* ≈20% better than *GAlign-3*, which confirms the correctness of using multi-order embeddings over traditional single-order embeddings.

D. Adaptivity to adversarial conditions

We answer (RQ3) by evaluating several adversarial factors.

Structural noises. In this experiment, we study the effect of structural noises. The noises are added by removing edges randomly. Fig. 3 illustrates the *Success@1* results, where the ratio of edge removal is varied from 10% to 50%. In general, all methods suffer performance drop when the noise level increases. Our model outperforms the baseline methods, with the success@1 goes from nearly 100% to around 80% when the edges removal ratio goes from 10% to 50%. Our model keeps the margin of 20% with the runner-up (FINAL). The performance of the two models PALE and REGAL drop more dramatically than the others. IsoRank does not perform well even when the noise level is low, as IsoRank may need more prior supervised information.

Attribute noises. This experiment studies the effect of attribute noises. The noises are added by changing the nodes' attribute randomly. Fig. 4 depicts the results with the attribute noise level varies from 10% to 50%. We only consider REGAL, FINAL and CENALP as baselines because other methods do not utilise attribute information. In general, the alignment output is worse when the noise ratio increases. Interestingly, our model maintains superior performance at all levels of noise and across datasets, with the Success@1 drops from nearly 100% to 60% when the noises go up to 50%. It is worth noting that the attribute noise has more effect on our model than the structural noise. For the baselines, REGAL are more robust to attribute noise than FINAL and CENALP.

Isomorphic level. This experiment investigates whether alignment methods can be used for an orthogonal problem of reconciling different types of networks (e.g. social network vs citation network). Fig. 8 depicts the result, where we vary the isomorphic level by synthesising the source and target networks from an original network such that two of them share a ratio of original nodes as overlap. In general, the alignment performance drops when the isomorphic level is small. However, our model outperforms in all cases, with a *Success@1* margin of 30% better than the runner-up (REGAL).

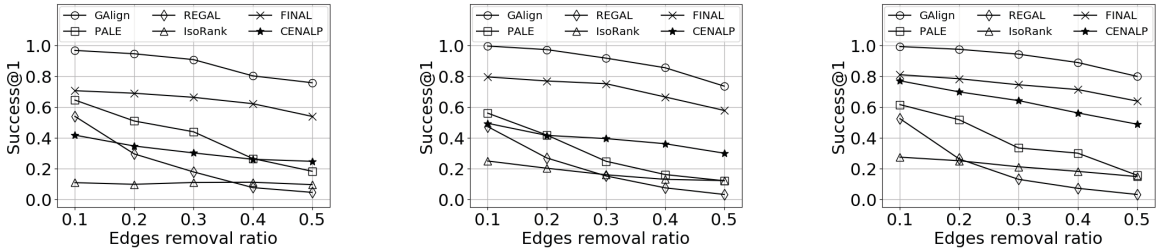
E. Hyperparameter sensitivity

This experiment answers (RQ4). Only important hyperparameters are shown due to space limitation.

Number of GCN layers. Fig. 6 shows the effect of the number of GCN layers k, on our model performance, where $\{H^{(l)}\}_{l=1}^{k}$ column represents our multi-layer approach and other columns represent the cases of using the embeddings at that layer only. While $k = 2$ produces the best result, it is interesting to see that using more number of layers does not increase alignment performance. This result confirms existing empirical evaluations [37], which point out the paradox that too deep GCNs are often worse than 2-layer models.

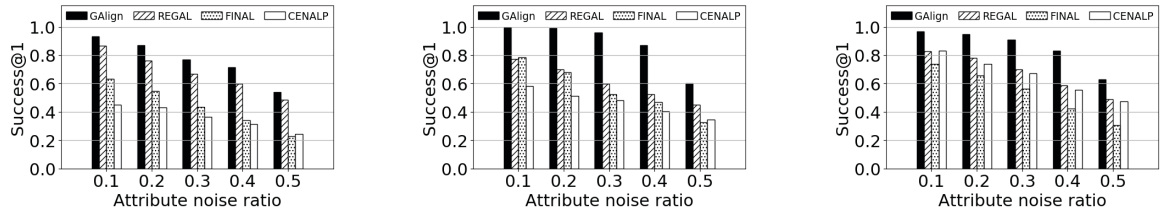
TABLE III: Network alignment comparison on real-world datasets

Dataset	Metric	GAlign	CENALP	PALE	REGAL	IsoRank	FINAL
Douban Online-Offline	MAP	**0.5632**	0.3537	0.1901	0.1005	0.1299	0.5539
	AUC	**0.9917**	0.7429	0.8899	0.9107	0.9005	0.9872
	Success@1	**0.4526**	0.2572	0.0775	0.0456	0.0903	0.4383
	Success@10	**0.7800**	0.4618	0.4479	0.2030	0.2048	0.7710
	Time(s)	26.7	10157.2	68.1	**14.3**	25.5	198.7
Flickr-Myspace	MAP	**0.1608**	0.1322	0.0059	0.0990	0.0085	0.0429
	AUC	**0.9738**	0.8670	0.5444	0.9692	0.6470	0.6130
	Success@1	**0.0774**	0.0687	0.0000	0.0464	0.0000	0.0206
	Success@10	**0.3127**	0.2302	0.0206	0.1950	0.0275	0.0722
	Time(s)	93.1	25520.0	123.0	**33.9**	222.6	249.1
Allmovie-Imdb	MAP	**0.8496**	0.5693	0.7601	0.1888	0.5271	0.8459
	AUC	**0.9971**	0.9581	0.9868	0.9862	0.9596	0.9885
	Success@1	**0.8214**	0.4866	0.6947	0.0953	0.4653	0.7647
	Success@10	0.9003	0.8327	0.7159	0.3869	0.6427	**0.9609**
	Time(s)	336.5	57401.0	1679.7	**76.1**	323.7	353.3



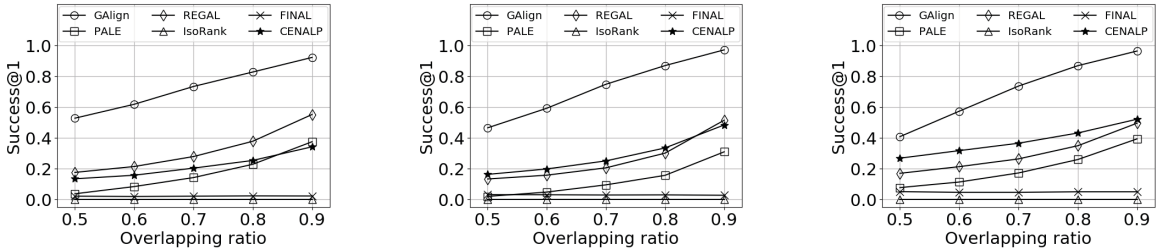
(a) BN (b) Econ (c) Email

Fig. 3: Robustness against structural noises



(a) BN (b) Econ (c) Email

Fig. 4: Robustness against attribute noises



(a) BN (b) Econ (c) Email

Fig. 5: Robustness against isomorphic level

TABLE IV: Ablation test

Dataset	Metric	GAlign	GAlign-1	GAlign-2	GAlign-3
Douban	MAP	**0.5632**	0.5577	0.5622	0.3467
	Success@1	**0.4526**	0.4472	0.4453	0.2290
Allmovie-Imdb	MAP	**0.8496**	0.8466	0.6894	0.6934
	Success@1	**0.8214**	0.8170	0.6100	0.6376

Importance of GCN layers. Table V shows the alignment performance with different setup of importance weights $\theta^{(l)}$ of GCN layers. Here, the weights represent the relative importance of one layer over another (their sum is 1). It can be seen that using only one layer (corresponding weight = 1) does not lead to the best performance and sometimes degrades the alignment output significantly (especially when using only node attributes). Interestingly, the best case happens when we put most of importance in the middle layer ($\theta_1 = 0.5$),

less importance in the deeper layer ($\theta_2 = 0.166$), and more importance in the shallower layer ($\theta_0 = 0.33$). This is because each network has a particular structural degree (e.g. diameter, communities, node degrees); and thus, finding the right amount of neighbouring information is important.

Embedding dimension. Fig. 7 studies the sensitivity of the embedding dimension of GCN layers. In general, users should not choose a high number of dimensions as it does not increase the performance (*Success@1*) significantly while the time and space complexity definitely become larger.

F. Qualitative study

We answer (RQ5) by demonstrating our whole pipeline on a toy dataset, which is extracted from 10 movie pairs of the AllMovie - IMDB dataset. The node attribute is the movie categories. The multi-order embedding step (§V) is visualised by t-SNE in: (i) Fig. 8a for embeddings at final layer (traditional), (ii) Fig. 8b for the multi-order embeddings (the embeddings at all layers are concatenated). It can be seen that the multi-order approach produces closer embeddings for anchor nodes. The refinement step (§VI-B) is visualised in Fig. 8c, which improves the result by making the embeddings of anchor nodes more distinctive to others (e.g. embeddings of "School Ties" and "Duets" are separated after refinement).

VIII. Related Work

The comparative network analysis has been studied for decades [10]. One classical problem is network matching, where the similarity level of two networks are determined, with various techniques such as network distances [3], network properties [14] and network kernels [17]. Another problem is network reconciliation, where two different networks are connected for data integration [22, 26]. Our work solves the recent network alignment problem, which finds anchor links between an original network and its variants [2, 29, 32, 34].

A. Network alignment

Classical approach. Basic methods use string distance metrics to compare the name or the description of the nodes (e.g. TF-IDF, Jaccard, Levanstein, Euclidean) [6, 27]. However, they are prone to textual noise and fail to utilize the topology information. Some other techniques consider external knowledge [15] or internal structure-based constraints [12]. However, they are domain-specific and requires intensive feature engineering. Instead of such trial-and-error, our model is end-to-end, saving human efforts and adapting to various adversarial conditions.

Spectral (Matrix factorisation) approach. Many approaches use matrix factorisation to compute the alignment matrix directly [34]. The classic IsoRank [32] propagates the pairwise node similarity along with structural consistency over the network. NetAlign [2] models the alignment problem as an integer quadratic programing problem and solves by a belief propagation. FINAL [39] leverages network structure, node feature and edge feature. REGAL [16] employs low-rank matrix approximation to speed up calculation. However, these approaches struggle to deal with large-scale networks due to the sparsity and massive size of the adjacency matrices.

Embedding-based approach. With the appearance of network embedding techniques [13]; alignment techniques leverage their scalability to deal with large networks. PALE [24] learns nodes embedding by maximising the co-occurrence likelihood of edge's vertices then applies linear or multi-layer perceptron (MLP) as mapping function. IONE [23] uses the same mapping function as PALE, but its embedding process takes into account second-order node similarity. DeepLink [41] employs unbiased random walk to generate embeddings using skip-gram then using auto-encoder and MLP to construct mapping function. However, these methods rely only topology information and therefore remain vulnerable to structure noise, which is very common in real-world networks.

Our work goes beyond the state-of-the-art theoretically and empirically by leveraging multi-order properties of GCN for consistent and unsupervised network alignment. Similar to our approach is [4]; however, they do not consider the violation cases of consistency constraints, which is common in real-world networks. We are the first to make the alignment model adapt to different types of noises via network augmentation.

B. Network embedding

Network embedding maps network nodes to a low-dimensional embedding space [5, 13], whose vectors can be used as features for various machine learning tasks such as network alignment [23, 24]. *Matrix-factorization* methods attempt to represent the topological relationship of network nodes in a matrix (e.g. node adjacency matrix, Laplacian matrix), then adopt factorization techniques directly on this matrix to obtain the embeddings [13]. *Random-walk* methods generate the random walks rooted from the network nodes, then learn the embeddings for the nodes so that the embeddings can capture the nodes' co-occurrences in the walks [28]. *Deep learning* methods leverage neural architectures, such as graph neural network and autoencoder, to incoporate node features and inductive capability in the same model [5, 9].

In this paper, we develop a specific network embedding model for network alignment by multi-order GCN. Unlike existing embedding-based techniques, our framework is fully unsupervised, which minimises human efforts.

IX. Conclusion

Our paper proposes a novel framework of fully unsupervised network alignment for attributed networks It is built on top of a multi-order embedding model that leverages the properties of GCN to guarantee consistency constraints. Moreover, the model is augmented with a data perturbation method to make the alignment adaptive to noises and consistency violations, which are all neglected by existing baselines. Especially, we propose an alignment refinement to detect potential noises, adjust the embedding accordingly, and make the alignment output robust to structural differences and attribute mismatches. The experiments show the superiority of our model, especially in *Success@1*, which are crucial for high-quality applications.

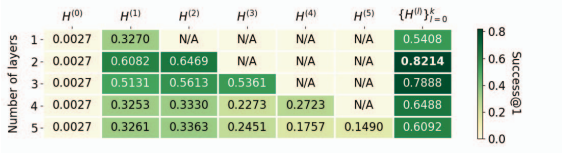


Fig. 6: Effects of #GCN-layers against Success@1 (w.r.t. embeddings used to compute alignment matrix)

TABLE V: Layer Weights

θ_0	θ_1	θ_2	Success@1
0.33	0.33	0.33	**0.8214**
0.33	0.50	0.17	0.8002
0.33	0.17	0.50	0.8179
0.00	0.67	0.33	0.7120
0.67	0.00	0.33	0.7820
0.33	0.67	0.00	0.7298
0.00	1.00	0.00	0.6082
0.00	0.00	1.00	0.6469
1.00	0.00	0.00	0.0027

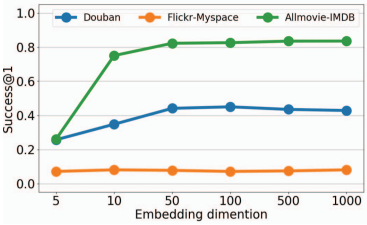
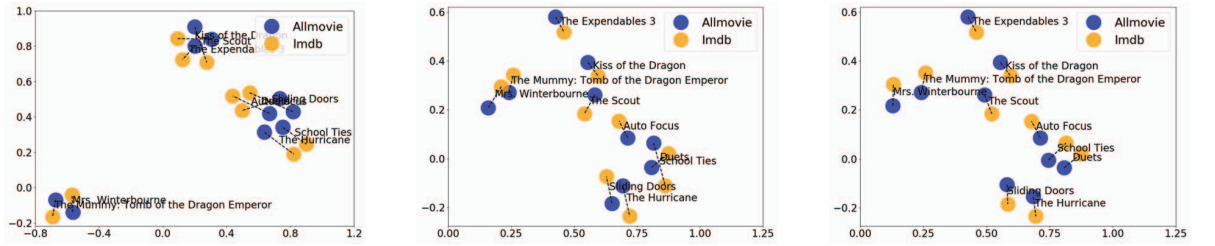


Fig. 7: Embedding dimension



(a) Traditional embeddings (b) Multi-order embeddings (c) Multi-order embeddings after refinement

Fig. 8: Qualitative study

REFERENCES

[1] K. Amunts, C. Lepage, et al. "BigBrain: an ultrahigh-resolution 3D human brain model". In: *Science* (2013), pp. 1472–1475.

[2] M. Bayati, M. Gerritsen, et al. "Algorithms for large, sparse network alignment problems". In: *ICDM*. 2009, pp. 705–710.

[3] H. Bunke. "Recent developments in graph matching". In: *ICPR*. 2000, pp. 117–124.

[4] C. Chen, W. Xie, et al. "Unsupervised Adversarial Graph Alignment with Graph Embedding". In: *arXiv preprint arXiv:1907.00544* (2019).

[5] H. Chen, H. Yin, et al. "Exploiting Centrality Information with Graph Convolutions for Network Representation Learning". In: *ICDE*. 2019, pp. 590–601.

[6] W. W. Cohen, P. Ravikumar, et al. "A Comparison of String Distance Metrics for Name-Matching Tasks." In: *IIWeb*. 2003, pp. 73–78.

[7] X. Du, J. Yan, et al. "Joint Link Prediction and Network Alignment via Cross-graph Embedding". In: *IJCAI*. 2019, pp. 2251–2257.

[8] C. T. Duong, T. D. Hoang, et al. "On Node Features for Graph Neural Networks". In: *arXiv preprint arXiv:1911.08795* (2019).

[9] C. T. Duong, H. Yin, et al. "Parallel Computation of Graph Embeddings". In: *arXiv preprint arXiv:1909.02977* (2019).

[10] F. Emmert-Streib, M. Dehmer, et al. "Fifty years of graph matching, network alignment and network comparison". In: *Inf. Sci* (2016), pp. 180–197.

[11] R. C. Fernandez, E. Mansour, et al. "Seeping semantics: Linking datasets using word embeddings for data discovery". In: *ICDE*. 2018, pp. 989–1000.

[12] F. K. Glückstad. "Terminological ontology and cognitive processes in translation". In: *PACLIC*. 2010, pp. 629–636.

[13] P. Goyal and E. Ferrara. "Graph embedding techniques, applications, and performance: A survey". In: *KBS* (2018), pp. 78–94.

[14] W. Hayes, K. Sun, et al. "Graphlet-based measures are suitable for biological network comparison". In: *Bioinformatics* (2013), pp. 483–491.

[15] W. He, X. Yang, et al. "A hybrid approach for measuring semantic similarity between ontologies based on wordnet". In: *KSEM*. 2011, pp. 68–78.

[16] M. Heimann, H. Shen, et al. "REGAL: Representation Learning-based Graph Alignment". In: *CIKM*. 2018, pp. 117–126.

[17] T. Horváth, T. Gärtner, et al. "Cyclic pattern kernels for predictive graph mining". In: *KDD*. 2004, pp. 158–167.

[18] T. T. Huynh, C. T. Duong, et al. "Network Alignment by Representation Learning on Structure and Attribute". In: *PRICAI*. 2019, pp. 698–711.

[19] K. Kim and J. Altmann. "Effect of homophily on network formation". In: *Comm. Nonlinear Sci. Numer.* (2017), pp. 482–494.

[20] G. Kollias, S. Mohammadi, et al. "Network Similarity Decomposition (NSD): A Fast and Scalable Approach to Network Alignment". In: *TKDE* (2012), pp. 2232–2243.

[21] D. Koutra, H. Tong, et al. "Big-align: Fast bipartite graph alignment". In: *ICDM*. 2013, pp. 389–398.

[22] M. Lenzerini. "Data integration: A theoretical perspective". In: *PODS*. 2002, pp. 233–246.

[23] L. Liu, W. K. Cheung, et al. "Aligning Users across Social Networks Using Network Embedding." In: *IJCAI*. 2016, pp. 1774–1780.

[24] T. Man, H. Shen, et al. "Predict Anchor Links across Social Networks via an Embedding Approach". In: *IJCAI*. 2016, pp. 1823–1829.

[25] H. Nassar, N. Veldt, et al. "Low rank spectral network alignment". In: *WWW*. 2018, pp. 619–628.

[26] Q. V. H. Nguyen, T. T. Nguyen, et al. "Pay-as-you-go reconciliation in schema matching networks". In: *ICDE*. 2014, pp. 220–231.

[27] L. Otero-Cerdeira, F. J. Rodríguez-Martínez, et al. "Ontology matching: A literature review". In: *ESWA* (2015), pp. 949–971.

[28] B. Perozzi, R. Al-Rfou, et al. "Deepwalk: Online learning of social representations". In: *KDD*. 2014, pp. 701–710.

[29] Y. Ren, C. C. Aggarwal, et al. "Meta diagram based active social networks alignment". In: *ICDE*. 2019, pp. 1690–1693.

[30] R. Rossi and N. Ahmed. "The network data repository with interactive graph analytics and visualization". In: *AAAI*. 2015.

[31] K. Shu, S. Wang, et al. "User Identity Linkage Across Online Social Networks: A Review". In: *KDD EN* (2017), pp. 5–17.

[32] R. Singh, J. Xu, et al. "Global alignment of multiple protein interaction networks with application to functional orthology detection". In: *PNAS* (2008), pp. 12763–12768.

[33] N. T. Tam, M. Weidlich, et al. "From anomaly detection to rumour detection using data streams of social platforms". In: *PVLDB* (2019), pp. 1016–1029.

[34] H. T. Trung, N. T. Toan, et al. "A comparative study on network alignment techniques". In: *ESWA* (2020), p. 112883.

[35] Z. Wang, Q. Lv, et al. "Cross-lingual knowledge graph alignment via graph convolutional networks". In: *EMNLP*. 2018, pp. 349–357.

[36] Q. Xie, Z. Dai, et al. "Unsupervised data augmentation". In: *arXiv preprint arXiv:1904.12848* (2019).

[37] K. Xu, C. Li, et al. "Representation learning on graphs with jumping knowledge networks". In: *arXiv preprint arXiv:1806.03536* (2018).

[38] H. Yin, L. Zou, et al. "Joint event-partner recommendation in event-based social networks". In: *ICDE*. 2018, pp. 929–940.

[39] S. Zhang and H. Tong. "Final: Fast attributed network alignment". In: *KDD*. 2016, pp. 1345–1354.

[40] Y. Zhang, Q. Yao, et al. "NSCaching: simple and efficient negative sampling for knowledge graph embedding". In: *ICDE*. 2019, pp. 614–625.

[41] F. Zhou, L. Liu, et al. "DeepLink: A Deep Learning Approach for User Identity Linkage". In: *INFOCOM*. 2018, pp. 1313–1321.