Network Alignment with Holistic Embeddings

Thanh Trung Huynh[1], Thang Chi Duong[2], Thanh Tam Nguyen[3*], Van Vinh Tong[4], Abdul Sattar[1], Hongzhi Yin[5], Quoc Viet Hung Nguyen[1]

[1]*Griffith University,* [2]*Ecole Polytechnique Federale de Lausanne,* [3]*Ho Chi Minh City University of Technology (HUTECH),* [4]*Hanoi University of Science and Technology,* [5]*The University of Queensland*

Abstract—Network alignment is the task of identifying topologically and semantically similar nodes across (two) different networks. It plays an important role in various applications ranging from social network analysis to bioinformatic network interactions. However, existing alignment models either cannot handle large-scale graphs or fail to leverage different types of network information or modalities. In this paper, we propose a novel end-to-end alignment framework that can leverage different modalities to compare and align network nodes in an efficient way. In order to exploit the richness of the network context, our model constructs multiple embeddings for each node, each of which captures one modality or type of network information. We then design a late-fusion mechanism to combine the learned embeddings based on the importance of the underlying information. Our fusion mechanism allows our model to be adapted to various types of structure of the input network. Experimental results show that our technique outperforms state-of-the-art approaches in terms of accuracy on real and synthetic datasets, while being robust against various noise factors.

Index Terms—network alignment, network embedding, community detection, multi-embedding

1 Introduction

Networks, also known as graphs, are powerful natural structures that capture the interrelations between entities in various application domains, such as social, biological and economic networks [1], [2]. Many applications require an analysis of the correlations between multiple networks [3], such as cross-platform social network analysis [4], cross-lingual knowledge graph integration [5], and pattern matching between protein-protein interaction (PPI) networks [6]. Network alignment is the task of recognising corresponding node pairs between two networks that share a similar structure and feature information, and plays an important role in such applications [7]. For example, in social networks, individuals typically have accounts with different social networks (e.g. Facebook, LinkedIn), and discovering accounts held by the same person across these networks can help in understanding user behaviour. This can enhance the performance of downstream applications such as friend suggestion and personalised content recommendation [5]. In the field of bioinformatics, aligning PPI networks between two species can shed light on the evolutionary process as well as enhancing cross-species gene prioritisation [6].

Due to the importance of network alignment, there is a rich body of work that has attempted to tackle this problem. Traditional methods often attempt to align networks directly by applying matrix factorisation to adjacency matrices, e.g., NetAlign [8], IsoRank [9], UniAlign [10] and FINAL [11]. However, since real-world networks are usually large, their matrices are also huge, which makes it difficult to apply matrix factorisation due to the high computational cost.

Recently, the emergence of network embedding techniques [12], [13] has enabled network information to be represented in a more efficient way than by using adjacency matrices. More specifically, network nodes are embedded into a low-dimensional space such that the topological and attributional relations between any two nodes in the original network are reflected by their embedding distance. With this new representation, embedding-based network alignment models [14], [15] do not suffer from the scalability problem and have achieved good results.

However, embedding-based models still face several challenges. Firstly, network nodes that share similar topological characteristics are assigned to similar embeddings; this means that a given network node is likely to have an embedding that is analogous to those of its close neighbours, making it hard to discriminate between these nodes. Secondly, most network aligners employ certain notions of consistency as the backbones of their models. For instance, structural consistency requires the neighbourhoods of pairs of nodes to be similar across two networks. However, such strict constraints are hard to enforce, especially in real-world networks [16]. For example, a person may be more active and have more relationships in a particular social network (e.g. Facebook) than in another (e.g. MySpace, LinkedIn). Thirdly, existing network alignment models use only a single embedding; this limits the capacity of the model, as one embedding can only capture certain aspects of the network. Models that employ shallow embedding techniques (e.g. PALE [14]) can only consider a node's direct neighbours, and thus fail to embody richer information such as the higher-order topological proximity. Other aligners apply graph neural network techniques (e.g. GCN [17]) to integrate more types of information, such as node attributes. However, node attributes provide only a local view, and cannot give a global view of the whole network. This impairs the accuracy of alignment, as several nodes with similar local information exist in the target graphs.

To address the above challenges, we propose a multi-embedding-based network alignment model. Our main idea

*Corresponding author

is to go beyond existing methods by adopting multiple embeddings for a single node and then to align the networks based on the similarity of their node embeddings. We implement this idea by constructing three types of node embedding: shallow embedding (for first-order proximity), deep embedding (for higher-order proximity and attributional information) and community-based embedding (for a global view of the network structure). Our model then finds the alignment result that maximises the consensus view of the three types of embedding. We can summarise our contributions as follows:

- We propose **NAME** (**N**etwork **A**lignment with **M**ultiple **E**mbedding), an end-to-end network alignment framework that learns multiple representations to identify the corresponding nodes. This is the first attempt to leverage holistic embeddings for network alignment, and enables us to incorporate the rich information available from the network structure and the node features, regardless of its modality.
- We propose two new presentation techniques that capture different node individual characteristics, such as higher-order structural proximity, attributional and community-aware information. The aim of the proposed presentation is to complement the common first-order proximity-aware embedding approaches [14].
- We design a late-fusion mechanism that uses data augmentation to combine the embeddings based on the importance of the type of information to the network structure. This means that our model is adaptive to various structures of the input network.
- We evaluate our model using several experiments on real and synthetic datasets. Our framework not only achieves better results than baseline techniques but also exhibits robustness to noise.

The remainder of this paper is organised as follows. Section 2 introduces a motivating example for the network alignment problem and reviews related works. Section 3 formulates the network alignment problem and gives an overview of our approach. Section 4.2, Section 4.1 and Section 4.3 describe our techniques for learning the rich information embeddings for network nodes. Section 5 describes how we combine the learned embeddings and retrieve the alignment result. Our empirical results are presented in Section 6, and Section 7 concludes the paper.

2 Preliminaries

2.1 Motivating Example

In this section, we describe the network alignment problem based on the motivating example shown in Fig. 1. In this example, eight accounts $1, 2, 3, 4, 5, 6, 7, 8$ in the source network Facebook are aligned with eight accounts $1', 2', 3', 4', 5', 6', 7', 8'$ in the target network LinkedIn. These accounts belong to eight people working in two universities. Each node contains heterogeneous types of information: structural information, such as the relationships (e.g. friendship, followership) between nodes in the networks, and attribute information, such as the *username*, *degree* and *age*.

In general, corresponding nodes retain their individual characteristics across different networks. For example, node 1 maintains connections to its neighbour nodes 2, 3 and 4

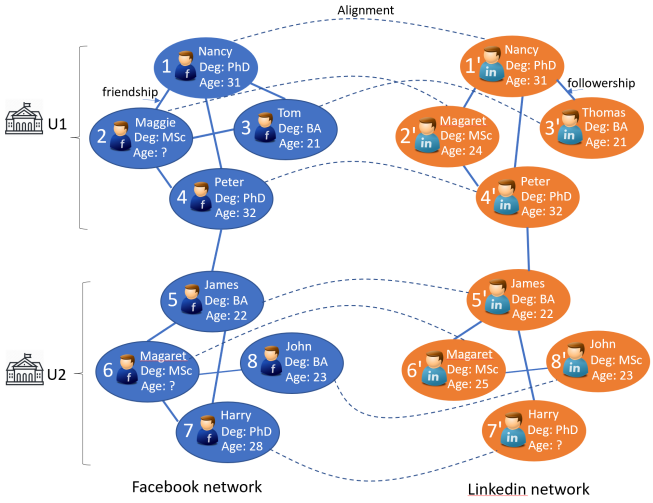


Fig. 1: Motivation example

in the target network. In addition, node 1 has the same attributes (username, degree, age) as its aligned node $1'$ in the target network. From a global viewpoint, node 1 and its neighbours form a densely connected group that represents a community of colleagues from university U1. This community structure is preserved in the target network. The consistency of the characteristic information of the nodes, which captures the individuality of the entity that the node represents, is the fundamental foundation of network alignment models.

However, this consistency does not always hold across networks. For example, a person may be more active or provide more profile information in one social network (e.g. Facebook) than another (e.g. Linkedin). In the above example, nodes 2 and 3 are connected, while their counterparts $2'$ and $3'$ are separate, and the same is true for nodes 6 and 7. Moreover, the attributes of corresponding nodes may also be different across networks. In the above example, node 8 and its corresponding node $8'$ have different academic degree information, since this user has not updated his information on his Facebook account. Another example is that the username of node 3 is Tom, while that of its counterpart $3'$ is Thomas. In some cases, information is even missing (e.g. the age for node 2). These consistency violations make the alignment process harder, as they can mislead the model into giving inaccurate results.

The richness of this node information can be leveraged to handle the issue of inconsistency, as the different types of information can be used to support each other. For example, while nodes 6 and $6'$ have different neighbourhoods, and node 6 is likely to be misaligned with node $2'$ as they share the same username, they can still be aligned, since they belong to the same community as nodes 5, 7 and 8 and share the same attributes. This example also demonstrates the usefulness of global information such as community in terms of alignment; this is valuable information that has often been ignored in recent works.

2.2 Related works

Network alignment (NA) is the task of identifying nodes belonging to the same identity across different networks [18].

It plays an important role in various domains of application such as social network analysis, cross-domain recommendation, PPI networks, and cross-lingual knowledge graphs. We present a brief overview of the literature and discuss the novelty of our approach compared to existing works.

Traditional approach. Traditional methods often align the nodes by comparing their names or descriptions using string distance metrics (e.g. Euclidean distance, Jaccard distance, Levanstein, TF-IDF) [19]. Although these techniques are straightforward to apply, they are prone to textual noise as they fail to exploit the rich information of the input networks, such as topology information. Some techniques utilise external knowledge or specific constraints in an attempt to mitigate this issue [20], [21]. However, they require intensive engineering and the alignment results are inferior to those of more advanced techniques.

Matrix-factorisation-based approach. As the network topology is an essential source of information, it has been exploited many recent techniques to achieve better node alignment. They apply matrix factorisation to the adjacency matrices of the input networks to compare the topological similarity of the nodes, with results in the form of an alignment matrix [22]. IsoRank [9], inspired by PageRank [23], propagates pairwise node similarities with respect to the topological consistency over the network. IsoRankN [24] extends the original IsoRank approach to a multiple network alignment setting. NetAlign [8] formulates the task as an integer programming problem. UniAlign [10] simplifies the original problem in order to use bipartite graphs, while FINAL [11] unifies the network topology, node features, and edge features under a loss function, and then finds a near-optimal alignment matrix by optimising the loss. REGAL [25] employs low-rank matrix approximation to reduce the computational effort required. In general, approaches based on matrix factorisation suffer from scalability issues when handling real-world, large-scale networks.

Vector representation approach. Following the emergence of network embedding techniques that learn low-dimensional embeddings for network nodes [26], recent network alignment techniques have utilised these techniques as alternative representations of traditional adjacency matrices to overcome the scalability issue [27]. Such techniques often follow a two-step pipeline: the nodes of the input networks are first embedded independently into the embedding spaces to capture the topological information (e.g. first-order proximity), and a mapping function is then learned in order to reconcile the separated learned embedding spaces into a shared space in which the corresponding nodes have similar embeddings. Following this strategy, PALE [14] learns the node embeddings by maximising the co-occurrence of direct neighbours for the input networks (thus focusing on first-order proximity). It then trains a linear transformation to reconcile the embedding spaces, using the partially pre-aligned nodes (called anchor links) as supervision information. IONE [15], [28] instead considers the second-order node similarity when constructing the node representations. DeepLink [29] employs an auto-encoder to better construct the mapping function. GAlign [30] exploits the multi-order nature of a graph convolutional network [17] to simultaneously integrate the attributional and topological consistency constraints to tackle the alignment task. However, these methods rely solely on network topology and fail to exploit the richness of network node information, and thus remain susceptible to structural noise in real-world networks.

Unlike existing works, our approach is the first to unify holistic embeddings to fully exploit the richness of the individual information on the nodes to create an end-to-end, consistent alignment. We were inspired by the PALE [14] technique, which intensively explores the first-order proximity of the network topology. We propose two new representation learning approaches that are tailored to the network alignment task and include information that PALE does not consider (e.g. attributional information, high-order proximity, community structure). The first approach, GCN-based embedding, is the improvement from our previous work [30], where we redesign the loss module to be appropriate with the supervised setting and thus facilitate the unifying with other embeddings. The second approach, global community-aware embedding, goes beyond the existing works in graph community detection [31], [32], [33] by encoding the community information in nodes' embedding. The approach is also designed to overcome the local optimal issue, where all nodes are assigned to one partition. Similar to our work is [28], but our communities preserve better the topology and leverages the community membership of the node itself rather than the neighbor anchors.

3 Problem Statement And Framework Overview

In this section, we first formulate the network alignment problem and then give an overview of our framework.

3.1 Problem Statement

Network alignment is the problem of recognising corresponding nodes across two isomorphic or near-isomorphic networks. Before presenting a formal statement of the problem, we define some necessary concepts as follows.

Attributed Network. A network or graph is a popular data structure that captures the interrelations between entities. It can be expressed as $\mathbb{G} = (V, \mathbf{A}, \mathbf{F})$, where V is a set of n network nodes and $\mathbf{A} \in \{0,1\}^{n \times n}$ is the adjacency matrix in which $\mathbf{A}(u,v) = 1$ if there exists an edge between u and v and $\mathbf{A}(u,v) = 0$ otherwise. $\mathbf{F} \in \mathbb{R}^{n \times m}$ is a node attribute matrix where each row of the matrix represents the features of the respective node (e.g. a user profile in a social network, a protein category in a bioinformatic network).

Anchor links. Given two networks $\mathbb{G}_s = (V_s, \mathbf{A}_s, \mathbf{F}_s)$ and $\mathbb{G}_t = (V_t, \mathbf{A}_t, \mathbf{F}_t)$, a pair of nodes (v, v'), with $v \in V_s, v' \in V_t$ and v, v', is defined as an anchor link if they belong to the same identity. Intuitively, the ultimate aim of network alignment is to discover all potential anchor links across two input networks.

Alignment matrix. Existing works often use an alignment matrix $\mathbf{S} \in \mathbb{R}^{n_s \times n_t}$ to represent the node correlations between $\mathbb{G}_s$ and $\mathbb{G}_t$, where n_s and n_t are the sizes of the respective networks. $\mathbf{S}(v, v')$ is the matching level between $v \in V_s$ and $v' \in V_t$. The alignment matrix therefore assigns a higher score to (v, v') if it is an anchor link, and vice-versa.

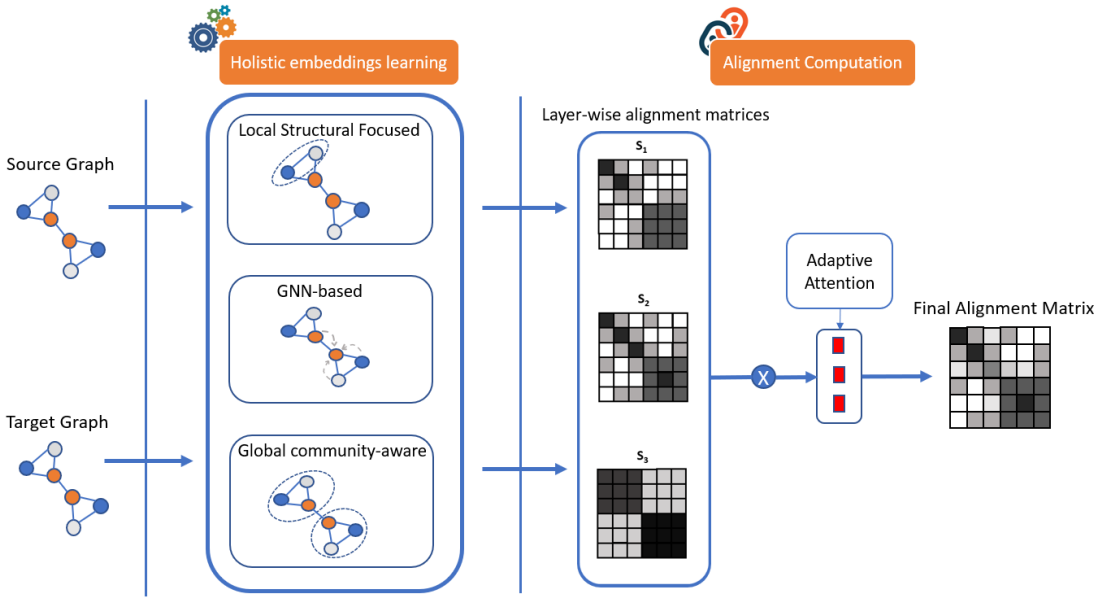


Fig. 2: Overview of NAME framework

Problem 1 (Attributed network alignment) *Given two attributed networks* $\mathbb{G}_s = (V_s, \boldsymbol{A}_s, \boldsymbol{F}_s)$ *and* $\mathbb{G}_t = (V_t, \boldsymbol{A}_t, \boldsymbol{F}_t)$*, the network alignment problem involves determining an alignment matrix* $\mathbf{S}$ *that accurately captures the node matching degree between* $\mathbb{G}_s$ *and* $\mathbb{G}_t$*.*

This formulation allows for a scalable calculation of the network alignment, since the anchor links can be inferred directly from the learned alignment matrix by selecting the highest matching score [25], [34]. The alignment matrix also enables not only a one-to-one but also a one-to-many alignment setting, which is required for network applications of different sizes [6].

3.2 Alignment constraints

As illustrated in the motivating example in Fig. 1, anchor links should have similar individual information across networks, since they reflect the same real-world entity. Based on this observation, existing works have stated the constraints on anchor links that need to be met to ensure good alignment performance [11], [25].

Structural consistency. This constraint states that the neighbourhoods between two nodes in a given network should be maintained by their anchor nodes in the other network. For example, in the motivating example in Fig. 1, Nancy (node 1) and Maggie (node 2) are friends on Facebook and their corresponding accounts on LinkedIn (nodes $1'$ and node $2'$) are also connected. Formally, if v, u are close neighbours in G_s and (u, u') and (v, v') are anchor links, v', u' should be close in G_t.

Attribute consistency. This constraint states that anchor nodes should share similar attribute values [11], [25]. For instance, in the motivating example, nodes 1 and $1'$ are Facebook and LinkedIn accounts that belong to Nancy, and the profile information is therefore the same (PhD degree, U1 affiliation). Mathematically, if (v, v') is an anchor link then it should hold that $\mathbf{F}_s(v) = \mathbf{F}_t(v')$ (implying $|\mathbf{F}_s(v)| = |\mathbf{F}_t(v')|$).

Many existing embedding-based network aligners fail to satisfy these two consistency constraints, due to differences in the modality of the information type and the limits on the richness of the information incorporated in the embedding [14]. Although a few techniques have attempted to cover both types of constraints, they consider them in separate steps [25] or overlook valuable information [30].

In this paper, we propose an end-to-end model that leverages multiple representations of nodes to exploit the node information to the fullest from many angles, thus allowing us to handle the different consistency constraints at the same time. We also show that in practice, the mechanism of consensus between many types of information makes our model more robust against the minor consistency violations (noise) that occur frequently in real-world networks.

3.3 Framework Overview

Addressing the problem formulated above is indeed a challenging task. The alignment model should consider the structure and attribute consistencies in an efficient manner, as these are essential guidelines for a high-quality alignment solution [35]. The alignment model should also be aware of structure and feature consistency violations (noise). While their presence can lead to performance degradation of the alignment result [22], straightforward consideration of this noise may conflict with integration of the consistency constraints. In addition, confusion between true corresponding and similar nodes in terms of their topology and attributes is one of the main factors that can reduce the accuracy of network aligners. Finally, the lack of information incorporated into a single embedding leads to short-sighted decisions when choosing true corresponding nodes.

In this work, we address the above challenges by developing an end-to-end network alignment framework in which the source and target networks are embedded into multiple representation spaces while simultaneously satisfying the consistency constraints. Fig. 2 shows an overview of our model. We first forward the source and target networks

through three embedding models. In addition to the first-order proximity-aware embedding, inspired by the work in [14], we propose two new embeddings to exhaustively exploit the network information: a GCN-based embedding that unifies the multi-order topological and attribute information of the network nodes, and a global-view embedding that captures the membership of network nodes in the structural communities of the whole graph. The learned representations are then fused to compute the final alignment result. Due to the use of this structure, our model requires the functionalities described below.

Embedding focused on local structure. We employ the skip-gram-based embedding model proposed in [14] to learn the embeddings for the source and target network nodes, with an emphasis on topological first-order proximity information. Since the representation spaces for the source and target networks are learnt independently, anchor links are employed to reconcile the two spaces. We briefly describe this embedding in Section 4.1.

Embedding based on a graph neural network. We design a GCN-based model to learn the representation of network nodes regarding both the high-order proximity and the attribute distribution information. To achieve this, the model contains multiple layers, each of which captures hidden features at a different order of topological neighbourhood structure. To efficiently train the model, we adopt a dual criteria loss, which aims to satisfy both the consistency constraint and the available anchor links. The details of this embedding can be found in Section 4.2.

Global community-aware embedding. We learn this global-aware representation for network nodes by considering their membership of communities in the whole network. We consider a community to be a set of nodes with many edges between them but few edges to nodes outside the group. For each node, we incorporate the membership information as a probability distribution over a set of communities. Nodes are similar if their membership distributions are close to each other, e.g., if they are both likely to be part of the same community. The details of the process of constructing these representations are described in Section 4.3.

Alignment computation. Rather than using a single embedding, as in existing works, we employ all of the above embeddings simultaneously to determine the alignment result. More precisely, we first calculate an embedding-based alignment matrix for each embedding, and then fuse these alignment matrices using a weighted sum that characterises the importance of the information incorporated in each embedding into the final alignment output. Since the importance of the type of information varies with the properties of the graph, we adopt an augmented learning process that helps to optimise the important coefficients and thus allows our model to adapt to different networks. The details are described in Section 5.

4 Holistic Representation Learning

In this section, we describe the embedding models that capture rich information available in the network nodes.

4.1 Embedding for local structure

Representation Learning. The embedding model was inspired by the skip-gram model [36], and is used to maximise the co-occurrence probability of two direct neighbour nodes v_i, v_j with the log-likelihood function:

$$L = \sum_{(v_i,v_j)\in E} log(p(v_i, v_j)) = \sum_{(v_i,v_j)\in E} log(\sigma(z_i, z_j)) \quad (1)$$

where z_i, z_j are the corresponding embeddings of v_i, v_j and σ is the sigmoid function.

However, the above loss function may lead to overfitting, as it considers only observed network edges. On the other hand, considering all network node pairs is costly. To alleviate this issue, a *negative sampling* [36] strategy is often employed:

$$L = \sum_{(v_i,v_j)\in E} (log(\sigma(z_i, z_j)) + \sum_{k=1}^{K}(1 - log(\sigma(z_i, z_k^{neg}))) \quad (2)$$

where in the second term, we want to maximise the distance between a node and K negative samples z_k^{neg}.

Representation Space Reconciliation. Since the source and target networks are embedded independently, a mapping function Θ is required to reconcile the two embedding spaces. The mapping function Θ may be either a linear function [37] or a multilayer perceptron [14], and can be learned by optimising the loss function:

$$L = \sum_{(v_s,v_t)\in T} ||\Theta(z_{v_s}) - z_{v_t}||_F \quad (3)$$

where z_v^s, z_v^t is the embedding of two anchor nodes (v_s, v_t) in the prealigned anchor link set T and $||.||_F$ is the Frobenius norm. The intuitive aim of the loss function is to guarantee that after the mapping, the embeddings of the anchor nodes in T are similar. After reconciliation, the correlation between the source and target nodes can be computed directly based on their embeddings.

4.2 Embedding based on a graph neural network

To reinforce the local structural embedding, we design a specific GCN model that simultaneously encodes the structural and attribute information. The similarity of two anchor nodes then depends on both their topological and semantic properties.

GCN-based embedding model. A GCN is a deep neural network model consisting of k layers. Each hidden layer captures the structural and attribute information at the same time via a message-passing strategy. Starting from the initial node features, the hidden features at the l-th layer are aggregated from the hidden features of the neighbour nodes at the $(l-1)$-th layer [38]. Formally, let the hidden feature at layer l be $H^{(l)}$ $(1 \leq l \leq k)$. We then have:

$$\mathbf{H}^{(l)} = \sigma(\hat{\mathbf{D}}^{-\frac{1}{2}}\hat{\mathbf{A}}\hat{\mathbf{D}}^{-\frac{1}{2}}\mathbf{H}^{(l-1)}\mathbf{W}^{(l)}) \quad (4)$$

where $\mathbf{H}^{(l-1)}$ is the embedding in the previous layer, and $\hat{\mathbf{A}} = \mathbf{A} + \mathbf{I}_n$ is the self-loop adjacency matrix, where $\mathbf{I}_n$ is the identity matrix, $\hat{\mathbf{D}}$ is a diagonal matrix constructed from $\hat{\mathbf{A}}$ as $\hat{\mathbf{D}}_{i,i} = \sum_j \hat{\mathbf{A}}_{i,j}$, $\mathbf{W}^{(l)}$ is a $d^{(l-1)} \times d^{(l)}$ weight

matrix and $\sigma(.)$ is an activation function. Rather than using a ReLU, as in traditional approaches [17], we use $tanh$ as an activation function, since a ReLU loses information when the signal is negative and is therefore unsuitable for the network alignment task.

Most existing works employ only the hidden features in the final layer $\mathbf{H}^{(k)}$ in the node representation [17], [39], [40], since the deepest layer collects information from all previous layers. However, while deeper layers contain richer topological patterns, they are prone to structural noise that is aggregated from the previous layers. The topological information also tends to be diluted in the deeper layers [41], especially for expander-type networks, since the collective information from a large neighbour set would overshadow the effect of each individual node.

In light of these problems, we use the learned embeddings at all layers $\{\mathbf{H}^{(l)}(v)\}_{l=0}^{k}$ for node embedding, in an approach called *multi-order embeddings*. This strategy allows our framework to exhaustively exploit the topological information of the network at both the local and global levels. To find a trade-off between the deep and shallow GCN models, we carefully tune the number of GCN layers, and the importance of this step is demonstrated through experiments (see Section 6.7)

Loss function. To leverage the multi-layer architecture of the GCN-based model, we propose a *layer-wise* loss function that guarantees two criteria, i.e. is both *consistency-aware* and *anchor-aware*. More precisely, for each embedding at layer $\mathbf{H}^{(l)}$, the loss function contains two components, as follows:

- *Consistency-aware loss*: This loss component aims to integrate the consistency constraints by minimising the distance between layer-wise embeddings of neighbouring nodes while maximising that of unrelated nodes:

$$J_s^{(l)} = ||\hat{\mathbf{D}}^{-\frac{1}{2}}\hat{\mathbf{A}}\hat{\mathbf{D}}^{-\frac{1}{2}} - \mathbf{H}^{(l)}{\mathbf{H}^{(l)}}^T||_F \tag{5}$$

  where $||.||_F$ is the Frobenius norm that allows us to measure the distance between matrices. We use normalised Laplacian matrix $\hat{\mathbf{D}}^{-\frac{1}{2}}\hat{\mathbf{A}}\hat{\mathbf{D}}^{-\frac{1}{2}}$ rather than the adjacency matrix $\mathbf{A}$ in order to incorporate richer topological information into the embeddings, which helps to maintain the individual information of the entity nodes.
- *Anchor-aware loss*: The core idea of this loss component is to ensure that the source and target node embeddings belong to a shared representation space. This loss enforces the embeddings of the anchor pairs at each layer to be similar:

$$J_a^{(l)} = \sum_{(v,v')\in T} ||\mathbf{H}_s^{(l)}(v) - \mathbf{H}_t^{(l)}(v')||_F \tag{6}$$

  where T is the set of known anchor pairs.

The final loss function at each layer is a weighted-sum combination of the two parts:

$$J^{(l)} = \alpha^{(l)}J_s^{(l)} + (1-\alpha^{(l)})J_a^{(l)} \tag{7}$$

where $\alpha^{(l)}$ is a learnable parameter that is used to balance the importance of two components. The combined loss function at all layers is as follows:

$$J(\mathbb{G}) = \sum_{l=1}^{k} J^{(l)} \tag{8}$$

Our loss function goes beyond existing approaches to incorporate all low- and high-order embeddings, to find a trade-off between these local and global forms of information and to exploit the rich context of the network nodes.

To ensure that the source and target networks are embedded into the same vector space, we use a weight sharing scheme similar to Siamese models [42]. In particular, the GCN models of both the source and target networks use the same weight matrix for each layer, i.e. same set of matrices $\{W^{(l)}\}_{l=1}^{k}$. This mechanism guarantees that the embeddings of the source and target nodes have a common embedding space. This removes the requirement for a reconciliation step for the two embedding spaces. In addition, this mechanism ensures that the consistency constraints are satisfied, since corresponding nodes with the same structural and attributional information propagated with shared weights will have identical embeddings [30].

4.3 Global community-aware embedding

In this section, we design another embedding model that provides a global view of network nodes based on the structure of the community, which provides another valuable source of information for the alignment process.

Community detection is a fundamental problem in network science [31], [32]. Intuitively, a node community (cluster) is often defined as a group of densely connected nodes with more edges joining these nodes than edges connecting to nodes in other groups. Formally, given a graph $\mathbb{G} = (V, \mathbf{A})$, the aim of community detection is to find a partition $\mathbb{C} = \{C_1, \ldots, C_m\}$ of m disjoint communities where $\bigcup_{i=1}^{m} C_i = V$ and $C_i \cap C_j = \emptyset$.

Membership matrix. We represent the partition by a membership matrix $\mathbf{M} \in \{0,1\}^{n\times m}$, with n rows representing the network nodes and m columns representing the m communities in $\mathbb{C}$. Note that since the communities are disjoint, each network node belongs to only one community, and thus each row of the membership matrix $\mathbf{M}$ is a one-hot vector. The problem of detecting communities can then be represented as the problem of finding an optimised membership matrix.

The membership matrix $\mathbf{M}$ not only clearly represents the community partition of network nodes but can also be used to generate a new coarsened adjacency matrix in which the "nodes" are communities:

$$\mathbf{A}^{com} = \mathbf{M}\mathbf{A}\mathbf{M}^T \tag{9}$$

In $\mathbf{A}^{com}$, the elements on the main diagonal $\mathbf{A}_{ii}^{com}$ represent the number of inner connections (edges) among the nodes in community C_i, while the non-diagonal elements $\mathbf{A}_{ij}^{com}$ represent the number of cross connections between communities C_i and C_j.

Embedding model. In order to learn the optimised membership matrix $\mathbf{M}$, we propose to use a *membership probability* matrix $\mathbf{P} \in \mathbb{R}^{n\times m}$, where each element p_{ij} represents the likelihood of node i belonging to community j. This probability matrix can be considered a relaxed version of the membership matrix (from a binary matrix to a real matrix). Using this representation bring many benefits. Firstly, the membership matrix $\mathbf{M}$ can be inferred from the membership

probability matrix **P** (by assigning to each node the community with the highest probability). Secondly, the use of a real value matrix helps to facilitate the learning process. Thirdly, the probability vectors can be used as representations of the network nodes, similar to the embedding matrix in shallow embedding techniques [12], [43].

Since the membership matrix **M** is discrete while its relaxed version **P** is continuous, we apply a Gumbel-Softmax [44] technique to sample **M** from **P**, as this is a differentiable operator (while the *argmax* operator is non-differentiable) and is therefore suitable for our end-to-end neural network model. Each row of **M** is then calculated from **P** as follows:

$$m(i) = \frac{\exp(\frac{p(i)+g\beta}{t})}{\sum_j^m \exp(\frac{p(j)+g\beta}{t})} \tag{10}$$

where t is a constant representing the temperature, g is a sample from the Gumbel distribution and β is a noise hyperparameter. When the temperature t approaches zero, a sample from the Gumbel-Softmax distribution becomes a one-hot vector, which helps in transferring the probability matrix **P** to **M**.

Community-aware loss function. As mentioned above, the ultimate goal of community detection is to find a partition in which the network nodes are divided into well-separated communities, where nodes within each community are densely connected and interconnections between communities are less common. Hence, the desired coarsened adjacency matrix $\mathbf{A}^{com}$ from Eq. 9 should have large values on the main diagonal elements, which reflect the number of connections inside the communities, and small values on the non-diagonal elements, which reflect the cross-connections between the communities. This desired characteristic can be captured by the following loss function:

$$\mathbf{L} = -\sum_i^m \mathbf{a}_{ii}^{com} = -\text{tr}(\mathbf{A}^{com}) = -\text{tr}(\mathbf{MAM}^T) \tag{11}$$

where tr(**A**) is the trace of matrix **A**. Note that when integrated into the model, the loss function receives continuous values rather than integer values, as the membership matrix **M** is the estimated version based on the probability matrix **P**. We learn the optimised membership matrix probability **P*** by optimising the community-aware loss function, and the rows of this matrix are used as representations of the network nodes.

5 Alignment Computation

In this section, we explain how we construct the alignment result from the separately learned representations. Note that each embedding captures different aspects of the node information, ranging from local features (first-order proximity, attribute distribution) to global ones (community membership). Combining these representations is helpful, as this information can be used together to overcome the consistency violations that arise in real-world networks.

5.1 Construction of the alignment matrix

In order to combine the different embeddings, we need to overcome several challenges. Firstly, the embeddings are not equally important, and depend heavily on the properties of the networks. For example, although the neighbourhood aggregation procedures used in GCN models are able to integrate structural and attribute information at the same time, they perform poorly in expander-type networks, as a node may have a large number of neighbours, leading to saturation of the topology information in GCN models. In these networks, local structural embedding plays a more important role, since the high-order embedding loses the individual information of the node due to too much propagation of node information. In addition, global information such as the community structure may be unclear for some networks, meaning that aligning individual nodes is less important. In other words, the way in which multiple embeddings are combined must be adapted to each network. Secondly, anchor links are scarce in practice, meaning that it is impractical to base the reconciliation of source and target embedding spaces on anchor nodes. As a result, self-supervised methods that can learn anchor links are more useful, although these must be applied carefully to avoid bias and degraded alignment results.

Embedding-wise alignment matrix. To mitigate the incompatibility between different types of representations, we first define an embedding-wise alignment matrix constructed from each type of embedding. In more detail, given the embeddings of a source network $\mathbf{Z_s^{(l)}}$ and a target network $\mathbf{Z_t^{(l)}}$ generated by a representation learning technique l, the embedding-wise alignment matrix is calculated as follows:

$$\mathbf{S}^{(l)} = \mathbf{Z}_s^{(l)} {\mathbf{Z}_t^{(l)}}^T \tag{12}$$

where $\mathbf{S}^{(l)}(i,j)$ represents the similarity of node v_i in the source network to node v_j in the target network according to the embedding technique l. Note that the size of $\mathbf{S}^{(l)}$ does not depend on the embedding technique, as it always has the same size $n_s \times n_t$ where n_s and n_t are the numbers of nodes in the source and the target networks, respectively. This facilitates the combination of different embedding-wise alignment matrices. The intuition underlying the above equation is that the source and target node embeddings are in the same embedding space, and the level of matching between the two nodes can be measured directly based on their dot product.

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

$$\mathbf{S} = \sum_{l=0}^{k} \theta^{(l)} \mathbf{S}^{(l)} \tag{13}$$

where $\theta^{(l)}$ is the importance factor of embedding technique l. The anchor links can then be retrieved from **S** by taking the target node with the highest matching score for each source node.

5.2 Data augmentation for matrix aggregation

As mentioned above, one of the challenges of this approach involves determining the optimised importance factor of each embedding technique, as these vary with the properties of the input network. In our work, we use network augmentation to automatically optimise the importance factor.

Perturbation-based Network Augmentation. Formally, starting from an original network $\mathbb{G} = (V, \mathbf{A}, \mathbf{F})$ (a source or target network), we produce an augmented network $\mathbb{G}_p = (V_p, \mathbf{A}_p, \mathbf{F}_p)$ with adjacency matrix $\mathbf{A}_p$ as follows:

$$\mathbf{A}_p = \mathbf{P}\mathbf{A}\mathbf{P}^T \tag{14}$$

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

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

Adaptive attention consensus mechanism. We leverage the augmentation mechanism to enable the model to reflect important embeddings. Given the original graph and its perturbed version, a good model should maintain similar embeddings between the two versions and perfectly align the network nodes. With this in mind, we learn the importance factor by optimising the following function:

$$\underset{\theta}{\operatorname{argmax}} \sum_{(v_i^{\mathbb{G}}, v_j^{\mathbb{G}^*}) \in A^{\mathbb{G},\mathbb{G}^*}} \mathbf{s}(i,j) \tag{15}$$

where $\mathbb{G}^*$ is the perturbed version of the original network $\mathbb{G}$. Intuitively, the aim of Eq. 15 is to maximise the matching scores of all the anchor links $(v_i^{\mathbb{G}}, v_j^{\mathbb{G}^*})$ between $\mathbb{G}$ and $\mathbb{G}^*$. As the source and target networks are isomorphic, we choose only the source network as the representative original graph.

6 Experimental Evaluation

In this section, we evaluate the following research questions:
(RQ1) Does our model outperform the baseline methods?
(RQ2) How much labeling effort does our model save?
(RQ3) How does our model scale up with network size?
(RQ4) What is the effect of each component of our model?
(RQ5) Is our model robust to adversarial conditions?
(RQ6) Is our model sensitive to hyper-parameters?
(RQ7) Can our technique be interpreted qualitatively?

In the remainder of this section, we first describe the experimental setup (Section 6.1), and then present our empirical evaluations, including an end-to-end comparison (Section 6.2), an analysis of supervision data effect (Section 6.3), an scalability analysis (Section 6.4), an ablation test (Section 6.5), an assessment of the adaptivity to adversarial conditions (Section 6.6), and a test of hyperparameter sensitivity (Section 6.7).

6.1 Experimental setup

Real datasets. We used three real-world alignment datasets from different domains, as follows:

- *Douban online vs Douban offline:* Douban is a Chinese social network in which nodes represent users and edges represent friendships [45]. The dataset is constructed from users of an online network and their offline event counterparts, and contains 1,118 anchor links [11].
- *Allmovie vs Imdb.* The Allmovie and IMDB netwoork was collected from the Rotten Tomatoes and the IMDB website [30]. The network nodes are movies, and edges connect any two movies that have at least one actor in common. The alignment output is constructed based on the identity of the film, and it contains 5,176 anchor links.
- *PPI:* This is a biological PPI dataset collected from the Molecular Signatures Database [40]. The nodes in the network represent for protein molecules, while the edges represent for their interactions in human tissues. There are 4,064 anchor links in the dataset.

TABLE 1: Statistics of real-world networks

Networks	#Nodes	#Edges	#Attributes
Douban Online	3906	8164	538
Douban Offline	1118	1511	538
Allmovie	6011	124709	14
Tmdb	5713	119073	14
PPI	1819	23735	50
Bn	1781	9016	20
Econ	1258	7619	20
Email	1133	5451	20

Synthetic data. We also synthesised an alignment dataset to investigate how adversarial factors such as structural noise affect the network alignment models. Starting with an original network, we generated a noisy version as described in Section 5.2 and performed alignment between the two versions.

- *bn:* This is a bioinformatic network which represents a partial brain structure [46]. Each node of the network depicts a brain voxel, and each edge represents a fibre tract that connects two voxels.
- *email:* This was collected from email data from European universities [30]. The nodes represent email addresses, and the edges connect any two addresses that exchange emails.
- *small-ppi:* This was generated by extracting large connected components (with more than 3,000 nodes) from the PPI dataset.

Table 1 summarizes the properties of alignment datasets using in our evaluation.

Baseline methods. We compared the performance of our proposed techniques with seven representative alignment methods:

1) *GAlign:* This is an embedding-based method that leverages multi-order representations for network nodes using a GCN model [30].
2) *PALE:* This is an embedding-based technique that learns node embeddings by maximising the co-occurrence of direct neighbours [14].
3) *REGAL:* This is a matrix-factorisation-based technique that employs low-rank matrix approximation to speed up the calculation [25].
4) *IsoRank:* This is a matrix-factorisation based alignment model that propagates pairwise node similarities over the network based on the assumption of structural consistency [9].

5) *FINAL:* This is a matrix-factorisation method that applies the structural, the node feature and the edge feature consistencies to optimise the alignment matrix [11].
6) *IONE:* This is a embedding-based method which uses two extra embeddings for each node to capture first and second-order proximity [15].
7) *UniAlign:* This is a matrix-factorization based technique which converts the input networks to a bipartite graph to facilitates the alignment matrix calculation [10].

Besides, we also examine the unsupervised variant of our proposed framework, namely *NAME-u*, which removes the anchor-aware loss from the GCN-based embedding and uses the nodes with highly correlated embeddings as anchor links for local structural embedding and global community aware embedding.

TABLE 2: Characteristics of baseline techniques

Technique	Prior mappings	Using Topology	Using Node Features
GAlign	Yes	Yes	No
PALE	Yes	Yes	No
REGAL	No	Yes	Yes
IsoRank	Yes	Yes	No
FINAL	Yes	Yes	Yes
IONE	No	Yes	No
UniAlign	No	Yes	Yes
NAME	Yes	Yes	Yes

For supervision data, we used 20% of the ground-truth values to train the baseline models. It should be noted that some techniques such as *FINAL* and *IsoRank* require a *prior alignment matrix* for training rather than the labelled alignments. In these cases, we generated the matrix using the labelled alignments if it was not available.

Metrics. We evaluated the alignment results with metrics that reflected the performance in terms of both *prediction* and *ranking* [4]. For *prediction*, we used $Success@q$ (also referred to as $Accuracy@q$ [11]), which indicates whether the true anchor link occurs in the top-q candidates. More precisely, for each anchor pair (v_s^*, v_t^*), 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 successful:

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

For *ranking*, we used the Mean Average Precision [14], a metric that indicates the rank ra of the true anchor target in the candidate list:

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

We also used AUC, a ranking metric that reflects the trade-off between precision and recall [4]. In a network alignment setting, where the output must find an anchor link for every node, AUC can be simplified to [4]:

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

We also report the total running time for each model, including training and alignment computing times, in seconds.

Hyperparameter tuning. Unless stated otherwise, the hyper-parameters were set as follows: $K = 20$ negative samples for *local structure embedding*; number of GCN layers $k = 2$; embedding size for all layers $d^{(l)} = 200$ for *GNN-based embedding*; noise hyperparameter $\beta = 1$ and Gumbell-softmax temperature $t = 10$ for *global community-aware embedding*. We also study the hyper-parameter sensitivity in Section 6.7. For each baseline methods, we report the best performance after following the parameter tuning methods reported in the original papers.

Reproducibility environment. The results were averaged over 20 runs to reduce the effects of randomness. All experiments were conducted on an AMD Ryzen ThreadRipper 3.8 GHz system with 64 GB of main memory and four GTX 2080 Ti graphic cards. We used Pytorch for implementation and Adam as a gradient optimiser.

6.2 End-to-end comparison

Table 3 shows an end-to-end comparison of our proposed model with the baseline techniques, which we use to answer (RQ1). In general, our model outperforms all the baselines in terms of the quality of alignment. We achieve better results for the important metrics of MAP, AUC, and Success@1 on all datasets, and especially for PPI, with values of Success@1 of 20-30% higher than other techniques. This is because our method is able to exploit the global community structure of protein interactomes. The unsupervised variant *NAME-u* of our model is able to achieves viable results and slightly better than the SOTA unsupervised technique *GAlign*, which proves the potential of our work in the adaptation to unsupervised setting. However, the performance of NAME-u is still considerably behind NAME. This might be because the GCN-based high-correlated embeddings can provide false anchor links.

GAlign and FINAL are the best methods of the baselines, and FINAL is even comparable to our model in terms of Success@10 for the Allmovie-IMDB dataset. This is because these two state-of-the-art approaches take into account both the structural and attribute information, in the same way as our model. Although REGAL also considers these two types of information, its performance suffers heavily from a strict model of consistency. The performance of PALE and IsoRank lags significantly, since these models consider only topology information. UniAlign performs the worst, as this model almost ignore the topology of the input networks by converting them into bipartite graphs.

The results of the running times for these techniques are shown in Fig. 3. Our technique is on a par with FINAL and IONE, although it is slower than some baselines, which is due to the trade-off between accuracy and computation time. REGAL and UniAlign are the fastest, as REGAL uses low-rank matrix approximation while UniAlign discard the topology information. However, their accuracy is quite low, with less than 20% of Success@1 for all the three datasets.

6.3 Saving of Labelling Effort

We answer (RQ2) by examining the performance of the techniques under different level of supervision data, as shown in Fig. 4. We vary the training ratio from 0.1 to 0.5 and compared the values for *Success@1* on the AllMovie dataset. It can be seen that our technique *NAME* outperform

TABLE 3: Comparison of network alignment on real-world datasets

Dataset	Metric	NAME	NAME-u	GAlign	PALE	REGAL	IsoRank	FINAL	IONE	UniAlign
Douban Online-Offline	MAP	**0.6366**	0.5811	0.5632	0.1901	0.1005	0.1299	0.5539	0.2467	0.0415
	AUC	**0.9949**	0.9943	0.9917	0.8899	0.9107	0.9005	0.9872	0.9289	0.8725
	Success@1	**0.5285**	0.4718	0.4526	0.0775	0.0456	0.0903	0.4383	0.1061	0.0422
	Success@10	**0.8648**	0.7931	0.7800	0.4479	0.2030	0.2048	0.7710	0.4715	0.1347
Allmovie-Imdb	MAP	**0.8993**	0.8544	0.8496	0.7601	0.1888	0.5271	0.8459	0.6892	0.0133
	AUC	**0.9980**	0.9972	0.9971	0.9868	0.9862	0.9596	0.9885	0.9828	0.9262
	Success@1	**0.8708**	0.8321	0.8214	0.6947	0.0953	0.4653	0.7647	0.3354	0.0788
	Success@10	0.9490	0.9111	0.9003	0.7159	0.3869	0.6427	**0.9609**	0.8114	0.1184
PPI-PPI	MAP	**0.9877**	0.7819	0.7499	0.8648	0.3029	0.3585	0.5151	0.1734	0.0054
	AUC	**1.000**	0.9999	0.9999	0.9995	0.9756	0.9907	0.7652	0.9770	0.8349
	Success@1	**0.9812**	0.6724	0.6233	0.8030	0.1787	0.2619	0.4958	0.1460	0.0080
	Success@10	**0.9941**	0.9868	0.9862	0.9669	0.5666	0.5516	0.5422	0.2791	0.0093

all other baselines with all level of supervision, especially at lower level. In more details, with the training ratio of only 10%, our technique can achieve the Success@1 of 86%, which is 20% higher than the figure for other state-of-the-art baselines such as PALE and GAlign. This proves the potential of our technique in handling the real-world dataset, where the prior anchor links are hard to obtain. For the baselines, FINAL and PALE enjoy the most when the training ratio increases, but their performance still is 5% behind us when the ratio reaches 0.5. GAlign and IsoRank show no difference for higher training level as they operate in unsupervised setting.

6.4 Scalability Analysis

In this experiment, we investigate the scalability of the techniques (RQ3) by studying their running time with large networks. To this end, we employ the generative model Erdös-Rényi [47] to generate the network node from 1,000 to 100,000 nodes with average degree of 10 for source network, and randomly delete 10% of nodes to obtain target network. The result is shown in Figure 5. The result is quite similar to the end-to-end experiment, as our technique running time is approximately similar to FINAL, PALE and is faster than IONE. When the network nodes reaches 100,000, our technique is still able to generate the result in less than 10^5 seconds. REGAL and UniAlign are still the fastest for all settings, as they sacrifice accuracy for speed.

6.5 Ablation test

We answer (RQ4) by designing some variants of our model:

- *NAME-1:* uses only *local structural embedding*.
- *NAME-2:* uses only *GNN-based embedding*.
- *NAME-3:* uses only *global community-aware embedding*.
- *NAME-4:* uses all three embeddings, but treats them equally rather than using data augmentation as described in Section 5.2.
- *NAME-5:* replaces the *GNN-based embedding* by the sampling-based GNN technique GraphSAGE [40].

Table 4 presents the results for only the most important metrics and datasets, due to space limitations. It can be seen that our original NAME model outperformed the other variants, thus proving the importance of unifying network information by our augmentation-based augmentation process. In more details, NAME performs significantly better than the single-embedding variants: NAME-1, NAME-2 and NAME-3, which justifies the need of information fusion. The accuracy of the global-aware embedding NAME-3 is poor, as only the community membership of the node is captured in this embedding and thus there is no way to distinguish nodes in a same community. On the other hand, NAME has 10-20% of Success@1 better than NAME-4 for the two datasets, which confirms the need of weighting embedding techniques by our augmentation-based mechanism. In NAME-5, the replacement of our GNN-based embedding by GraphSAGE, another sampling-based GNN network, causes the degradation of 5-10 % of Success@1. This highlights the superior of original GCN mechanism in the alignment task, which requires the information is maintained to the fullest.

6.6 Adaptivity to adversarial conditions

We answer (RQ5) by evaluating the robustness of each technique to three adversarial factors: structural noise, attribute noise, and graph size imbalance.

Structural noise. In this experiment, we studied the effect of structural noise on the performance of the alignment models. Noise was simulated in the target graph by randomly removing edges from the source graph. Fig. 6a shows the Success@1 results when the edge removal ratio ranged from 0.1 to 0.5. In general, our proposed NAME model showed the best robustness to this noise factor. Its Success@1 slightly decreased by around 5% when the ratio reached 0.5 and was at least 20% better than the runner-up model (GAlign). This is because our model is able to leverage the rich information of the node to compensate for the structural incompleteness. In contrast, the baseline models suffered significantly when the noise level was high. The Success@1 of PALE, FINAL and GAlign dropped by around 20% when the removal ratio reached 0.5. The performance of REGAL declined more dramatically, while IsoRank did not perform well even with a low level of noise.

Attribute noise. This experiment explored the effect of the attribute noise factor on the performance of the network aligners. Noise was added by randomly changing the attributes of the source network nodes. Fig. 8 shows the results of increasing the noise ratio from 10% to 50%. We only compare our NAME model with GAlign, REGAL and FINAL, since the other methods do not utilise attribute information. In general, all models suffered significantly from a degradation in accuracy as the attribute noise level increased. Our technique achieved stable results for all three datasets. For example, in small-ppi, the Success@1 only dropped to 60% when the noise level reached 0.5. REGAL

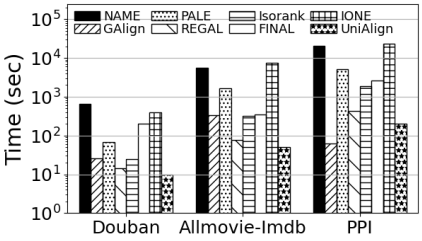


Fig. 3: End-to-end running times

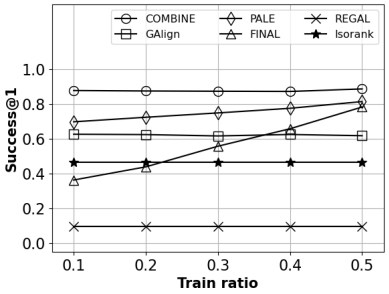


Fig. 4: Savings in labeling effort

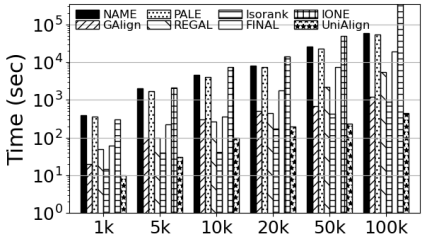
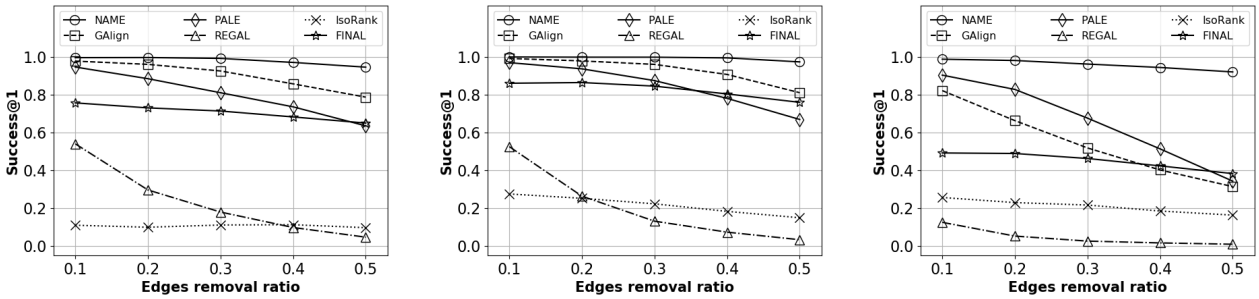


Fig. 5: End-to-end scalability study

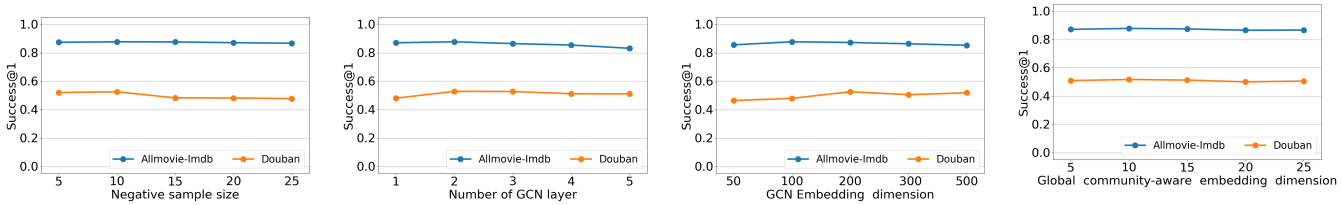
TABLE 4: Ablation Test

Dataset	Metric	NAME	NAME-1	NAME-2	NAME-3	NAME-4	NAME-5
PPI	MAP	**0.9877**	0.8648	0.9802	0.0016	0.7823	0.8423
	Success@1	**0.9812**	0.8030	0.9790	0.0004	0.7623	0.8012
Allmovie-Imdb	MAP	**0.8993**	0.7840	0.8197	0.0017	0.7599	0.7999
	Success@1	**0.8708**	0.7280	0.7754	0.0002	0.7227	0.7452



(a) bn (b) email (c) small-ppi

Fig. 6: Robustness against structural noises



(a) Negative sample size (b) Number of GCN layers (c) GCN embedding dim. (d) Global community aware embedding dimension

Fig. 7: Hyperparameter sensitivity tests

performed surprisingly well, and was even better than our model on the BN dataset, thanks to the absence of structural noise. FINAL performed the worst.

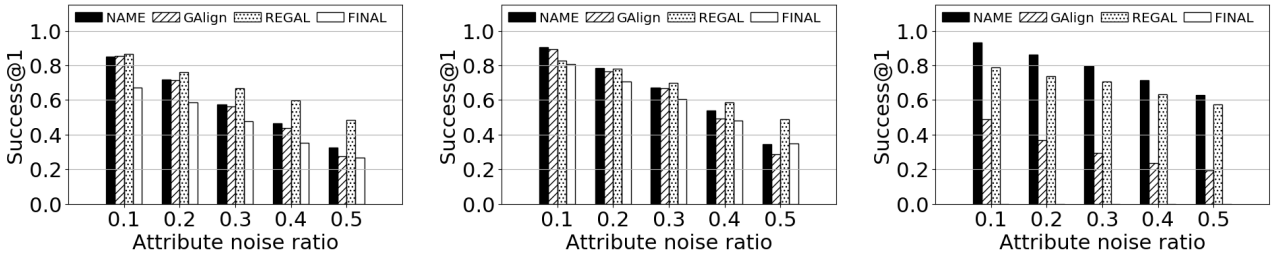
Graph size imbalance. In this experiment, we studied how an imbalance between the source and target networks affected the performance of the alignment models. Noise was added to the target graph by randomly removing nodes from the source graph. Fig. 9 shows the Success@1 of the alignment output determined by the models when the noise ratio was varied from 0.1 to 0.5. In general, the models showed similar trends to those found in the experiment with structural noise. More precisely, our model performed the best for this factor, and the Success@1 was nearly unchanged as the noise level was increased. Of the baselines, PALE and GAlign performed the best, though their robustness to this noise factor was significant lower than our model. REGAL and IsoRank were susceptible to the imbalance, and the Success@1 dropped below 20% when the noise level reached 0.5 for all three datasets.

6.7 Hyperparameter sensitivity

This experiment answers (RQ6). Only important hyperparameters are reported here, due to space limitations.

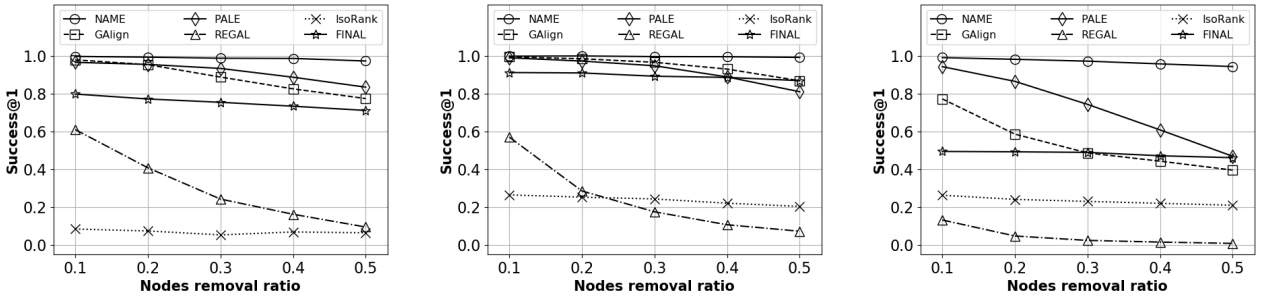
Negative sample size. Fig. 7a shows the Success@1 of our model when the negative sample size K for the local structure embedding is varied from 5 to 25. It can be seen that the performance of the model is stable against this hyperparameter, with $K = 10$ giving a slightly better result.

Number of GCN layers. Fig. 7b shows the effect of the number of GCN layers k on the performance of our model.



(a) bn (b) email (c) small-ppi

Fig. 8: Robustness against attribute noises



(a) bn (b) email (c) small-ppi

Fig. 9: Robustness against imbalance in the graph size

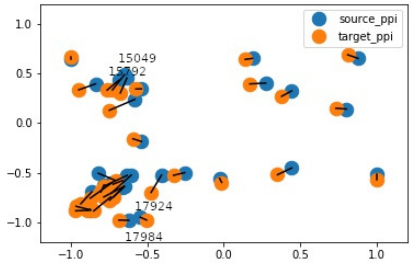


Fig. 10: Local structure

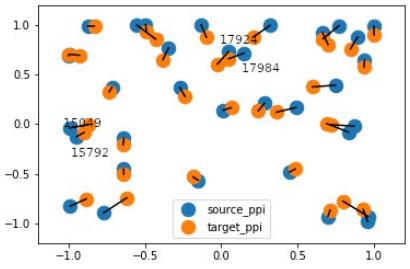


Fig. 11: Local structure & GCN-based

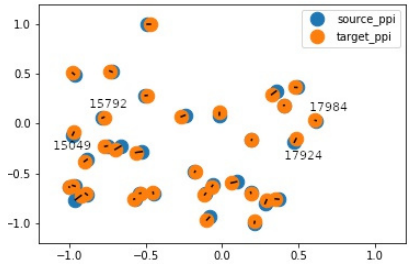
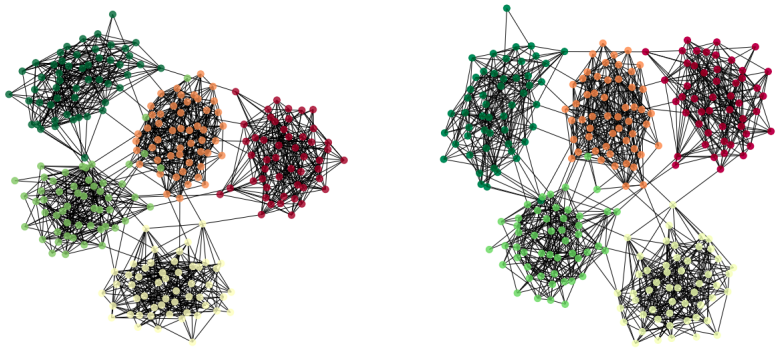


Fig. 12: Final model



(a) Source (b) Target

Fig. 13: Visualization of global community-aware embedding

While $k = 2$ produces the best result, it is interesting to see that using a higher number of layers does not improve the alignment performance. This result confirms existing empirical evaluations [48] which have found that GCNs that are too deep are often worse than two-layer models.

GCN Embedding dimension. Fig. 7c illustrates the sensitivity of the dimension of GCN-based embedding. In general, users should not choose a high number of dimensions, as this does not significantly increase the performance (Success@1) while the time and space complexities become considerably larger.

Global community-aware embedding dimension. Fig. 7d shows the sensitivity of the dimension of global community-aware embedding. While the dimension also presents the number of communities, which is unknown for the input networks, the empirical results show that our technique is not vulnerable to this hyperparameter.

6.8 Qualitative study

We answer (RQ7) by visualising the results using t-SNE on a toy dataset extracted from 20 protein pairs from the small-ppi dataset. Fig. 10, Fig. 11 and Fig. 12 represent the results generated by the variants NAME-1 and NAME-2 specified in Section 6.5 and the final model, respectively. It can be seen that when only the local structural embedding (NAME-1) is used, the embeddings of the nodes become entangled, since the true anchors and the close neighbours are similar. When a combination of GCN-based embedding (NAME-2) is used, the embeddings are more distinctive. The combination of all three embeddings in the final NAME model gives the best result. This is because the add of a global view (via global community-aware embedding) enrichs the node context and thus helps to disentangle the similar nodes (e.g. see the pairs (15049, 15792) and (17984, 17924)). To further demonstrate the effect of this embedding, we visualize the embedding of subgraphs extracted from PPI dataset in Fig. 13. The embedding successfully captures densely connected clusters and maintains stable result across source and target networks.

7 Conclusion

Our paper proposes a novel end-to-end network alignment framework that utilises the rich individual information of network nodes to align the nodes efficiently. In more detail, the model learns multiple representations for each network nodes, each of which captures one aspect of the node information. We reuse a representation learning technique from existing work and design two brand new techniques to complement this with different modalities (global community and attribute information). We then apply a late-fusion mechanism to unify the learned embeddings based on the importance of the underlying information. The mechanism is augmented with a data perturbation method to ensure that our model is adaptive to various structures of the input network and robust to potential noise. Our experiments demonstrate the superiority of our model, especially in terms of Success@1, which is crucial for high-quality applications. In future work, we plan to combine our model with approximate and caching mechanisms [49], e.g. a low-rank matrix representation, to reduce the memory complexity.

Acknowledgments

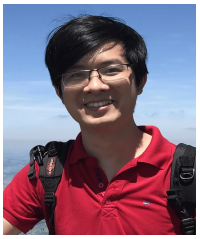
This work was supported by ARC Discovery Early Career Researcher Award (Grant No. DE200101465). Tong Van Vinh was funded by Vingroup Joint Stock Company and supported by the Domestic Master/ PhD Scholarship Programme of Vingroup Innovation Foundation (VINIF), Vingroup Big Data Institute (VINBIGDATA), code VINIF.2020.ThS.BK.07.

References

[1] A. Ghanbarpour and H. Naderi, "An attribute-specific ranking method based on language models for keyword search over graphs," *TKDE*, vol. 32, no. 1, pp. 12–25, 2018.
[2] J. Byun, S. Woo, and D. Kim, "Chronograph: Enabling temporal graph traversals for efficient information diffusion analysis over time," *TKDE*, vol. 32, no. 3, pp. 424–437, 2019.
[3] J. Wu, X. Zhu, C. Zhang, and S. Y. Philip, "Bag constrained structure pattern mining for multi-graph classification," *TKDE*, vol. 26, no. 10, pp. 2382–2396, 2014.
[4] K. Shu, S. Wang, J. Tang, R. Zafarani, and H. Liu, "User identity linkage across online social networks: A review," *KDD EN*, pp. 5–17, 2017.
[5] Z. Wang, Q. Lv, X. Lan, and Y. Zhang, "Cross-lingual knowledge graph alignment via graph convolutional networks," in *EMNLP*, 2018, pp. 349–357.
[6] H. Nassar, N. Veldt, S. Mohammadi, A. Grama, and D. F. Gleich, "Low rank spectral network alignment," in *WWW*, 2018, pp. 619–628.
[7] N. T. Tam, H. T. Trung, H. Yin, T. Van Vinh, D. Sakong, B. Zheng, and N. Q. V. Hung, "Entity alignment for knowledge graphs with multi-order convolutional networks," in *ICDE*, 2021, pp. 2323–2324.
[8] M. Bayati, M. Gerritsen, D. F. Gleich, A. Saberi, and Y. Wang, "Algorithms for large, sparse network alignment problems," in *ICDM*, 2009, pp. 705–710.
[9] R. Singh, J. Xu, and B. Berger, "Global alignment of multiple protein interaction networks with application to functional orthology detection," *PNAS*, pp. 12 763–12 768, 2008.
[10] D. Koutra, H. Tong, and D. Lubensky, "Big-align: Fast bipartite graph alignment," in *ICDM*, 2013, pp. 389–398.
[11] S. Zhang and H. Tong, "Final: Fast attributed network alignment," in *KDD*, 2016, pp. 1345–1354.
[12] A. Grover and J. Leskovec, "node2vec: Scalable feature learning for networks," in *KDD*, 2016, pp. 855–864.
[13] C. Shi, B. Hu, W. X. Zhao, and S. Y. Philip, "Heterogeneous information network embedding for recommendation," *TKDE*, vol. 31, no. 2, pp. 357–370, 2018.
[14] T. Man, H. Shen, S. Liu, X. Jin, and X. Cheng, "Predict anchor links across social networks via an embedding approach," in *IJCAI*, 2016, pp. 1823–1829.
[15] L. Liu, W. K. Cheung, X. Li, and L. Liao, "Aligning users across social networks using network embedding." in *IJCAI*, 2016, pp. 1774–1780.
[16] X. Yi, E. Bertino, F.-Y. Rao, K.-Y. Lam, S. Nepal, and A. Bouguettaya, "Privacy-preserving user profile matching in social networks," *TKDE*, 2019.
[17] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," in *ICLR*, 2017, pp. 1–14.
[18] T. T. Nguyen, M. T. Pham, T. T. Nguyen, T. T. Huynh, Q. V. H. Nguyen, T. T. Quan *et al.*, "Structural representation learning for network alignment with self-supervised anchor links," *ESWA*, vol. 165, p. 113857, 2021.
[19] L. Otero-Cerdeira, F. J. Rodríguez-Martínez, and A. Gómez-Rodríguez, "Ontology matching: A literature review," *ESWA*, vol. 42, no. 2, pp. 949–971, 2015.
[20] W. He, X. Yang, and D. Huang, "A hybrid approach for measuring semantic similarity between ontologies based on wordnet," in *KSEM*, 2011, pp. 68–78.
[21] F. K. Glückstad, "Terminological ontology and cognitive processes in translation," in *PACLIC*, 2010, pp. 629–636.
[22] H. T. Trung, N. T. Toan, T. Van Vinh, H. T. Dat, D. C. Thang, N. Q. V. Hung, and A. Sattar, "A comparative study on network alignment techniques," *ESWA*, vol. 140, p. 112883, 2020.
[23] L. Page, S. Brin, R. Motwani, and T. Winograd, "The pagerank citation ranking: Bringing order to the web." Stanford InfoLab, Tech. Rep., 1999.
[24] C.-S. Liao, K. Lu, M. Baym, R. Singh, and B. Berger, "Isorankn: spectral methods for global alignment of multiple protein networks," *Bioinformatics*, vol. 25, no. 12, pp. i253–i258, 2009.
[25] M. Heimann, H. Shen, T. Safavi, and D. Koutra, "Regal: Representation learning-based graph alignment," in *CIKM*, 2018, pp. 117–126.
[26] P. Goyal and E. Ferrara, "Graph embedding techniques, applications, and performance: A survey," *KBS*, vol. 151, pp. 78–94, 2018.
[27] T. T. Nguyen, T. T. Huynh, H. Yin, V. Van Tong, D. Sakong, B. Zheng, and Q. V. H. Nguyen, "Entity alignment for knowledge graphs with multi-order convolutional networks," *TKDE*, 2020.
[28] L. Liu, X. Li, W. Cheung, and L. Liao, "Structural representation learning for user alignment across social networks," *TKDE*, 2019.
[29] F. Zhou, L. Liu, K. Zhang, G. Trajcevski, J. Wu, and T. Zhong, "Deeplink: A deep learning approach for user identity linkage," in *INFOCOM*, 2018, pp. 1313–1321.
[30] 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 *ICDE*. IEEE, 2020, pp. 85–96.

[31] X. Liu, H. M. Cheng, and Z. Y. Zhang, "Evaluation of community detection methods," *TKDE*, vol. 32, no. 9, pp. 1736–1746, 2020.
[32] A. Ghasemian, H. Hosseinmardi, and A. Clauset, "Evaluating overfit and underfit in models of network community structure," *TKDE*, vol. 32, no. 9, pp. 1722–1735, 2020.
[33] M. E. Newman, "Modularity and community structure in networks," *PNAS*, vol. 103, no. 23, pp. 8577–8582, 2006.
[34] G. Kollias, S. Mohammadi, and A. Grama, "Network similarity decomposition (nsd): A fast and scalable approach to network alignment," *TKDE*, pp. 2232–2243, 2012.
[35] K. Kim and J. Altmann, "Effect of homophily on network formation," *Comm. Nonlinear Sci. Numer.*, vol. 44, pp. 482–494, 2017.
[36] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," in *NIPS*, 2013, pp. 3111–3119.
[37] M. Artetxe, G. Labaka, and E. Agirre, "Learning bilingual word embeddings with (almost) no bilingual data," in *ACL*, 2017, pp. 451–462.
[38] Z. Luo, L. Liu, J. Yin, Y. Li, and Z. Wu, "Deep learning of graphs with ngram convolutional neural networks," *TKDE*, vol. 29, no. 10, pp. 2125–2139, 2017.
[39] 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.
[40] W. Hamilton, Z. Ying, and J. Leskovec, "Inductive representation learning on large graphs," in *NIPS*, 2017, pp. 1024–1034.
[41] K. Xu, C. Li, Y. Tian, T. Sonobe, K.-i. Kawarabayashi, and S. Jegelka, "Representation learning on graphs with jumping knowledge networks," *arXiv preprint arXiv:1806.03536*, 2018.
[42] J. Bromley, I. Guyon, Y. LeCun, E. Säckinger, and R. Shah, "Signature verification using a" siamese" time delay neural network," in *NIPS*, 1994, pp. 737–744.
[43] B. Perozzi, R. Al-Rfou, and S. Skiena, "Deepwalk: Online learning of social representations," in *KDD*, 2014, pp. 701–710.
[44] E. Jang, S. Gu, and B. Poole, "Categorical reparameterization with gumbel-softmax," *arXiv preprint arXiv:1611.01144*, 2016.
[45] H. Yin, L. Zou, Q. V. H. Nguyen, Z. Huang, and X. Zhou, "Joint event-partner recommendation in event-based social networks," in *ICDE*, 2018, pp. 929–940.
[46] K. Amunts, C. Lepage, L. Borgeat, H. Mohlberg, T. Dickscheid, M.-É. Rousseau, S. Bludau, P.-L. Bazin, L. B. Lewis, A.-M. Oros-Peusquens *et al.*, "Bigbrain: an ultrahigh-resolution 3d human brain model," *Science*, pp. 1472–1475, 2013.
[47] P. Erdős and A. Rényi, "On the evolution of random graphs," *Publ. Math. Inst. Hung. Acad. Sci*, vol. 5, no. 1, pp. 17–60, 1960.
[48] K. Xu, C. Li, Y. Tian, T. Sonobe, K.-i. Kawarabayashi, and S. Jegelka, "Representation learning on graphs with jumping knowledge networks," *arXiv preprint arXiv:1806.03536*, 2018.
[49] Y. Zhang, Q. Yao, Y. Shao, and L. Chen, "Nscaching: simple and efficient negative sampling for knowledge graph embedding," in *ICDE*, 2019, pp. 614–625.

Huynh Thanh Trung is a Phd student at Griffith University, Australia. He received the Bachelor and Master degree in computer science from Hanoi University of Science and Technology, Vietnam. His research interests include graph analysis, network embedding, network alignment, image processing. He has publications in high-quality journals and conferences such as ICDE, ESWA and PRICAI.



Duong Chi Thang is a Phd student at Ecole Polytechnique Federale de Lausanne (EPFL), Switzerland. He received the Master and Bachelor degrees in computer science from EPFL and from Ho Chi Minh University of Technology, Vietnam. His research interests include graph analysis, network embedding, network alignment. He has publications in significant journals and conferences such as VLDBJ, SIGMOD, and SIGIR.

Nguyen Thanh Tam received the PhD degree in data science from Ecole Polytechnique Federale de Lausanne (EPFL), Switzerland. He received his Master degree in computer science at EPFL. His research interests include Data Filtering, Crowdsourcing, and Deep Learning for data lakes, data networks, and data streams. His research has been published in world-leading conferences and top-tier journals such as VLDB, ICDE, IJCAI, SIGIR, TKDE, JVLDB, and TIST.

Tong Van Vinh received the Bachelor's degree in computer science from Hanoi University of Science and Technology, Vietnam in 2020. His research focuses on deep learning and benchmarking technology. He has publications in top-tier conferences and journals such as ICDE, ESWA, TKDE and PRICAI.

Abdul Sattar is a Professor of Computer Science and Artificial Intelligence, and founding director of the Institute for Integrated and Intelligent Systems (IIIS) (2003-2015), at Griffith University, and was a research leader at NICTA (2005-2015), a national centre of excellence in ICT. He has published over 300 papers, several of these papers appeared in premier conferences and journals such as IJCAI, AAAI, AIJ, JAIR, CP, AAMAS. He is fellow of ACS, and life member of AAAI and ACM.

Hongzhi Yin received the PhD degree in computer science from Peking University, in 2014. He is a senior lecturer with the University of Queensland. He received the Australia Research Council Discovery Early-Career Researcher Award, in 2015. His research interests include recommendation system, user profiling,topic models, deeplearning, social media mining, and location-based services.

Nguyen Quoc Viet Hung is a Lecturer in Griffith University. He earned his Master and PhD degrees from EPFL (Switzerland). His research focuses on Data Integration, Data Quality, Information Retrieval, Trust Management, Recommender Systems, Machine Learning and Big Data Visualization, with special emphasis on web data, social data and sensor data. He published several papers in top-tier venues such as SIGMOD, VLDB, SIGIR, KDD, AAAI, ICDE, IJCAI, JVLDB, TKDE, TOIS, and TIST.