# iMap: Incremental Node Mapping between Large Graphs Using GNN

Yikuan Xia , Jun Gao *, Bin Cui *
Key Laboratory of High Confidence Software Technologies, Department of Computer Science, Peking University
*Corresponding Authors
{2101111522,gaojun,bin.cui}@pku.edu.cn

## ABSTRACT

Node mapping between large graphs (or network alignment) plays a key preprocessing role in joint-graph data mining applications like social link prediction, cross-platform recommendation, etc. Most existing approaches attempt to perform alignment at the granularity of entire graphs, while handling the whole graphs may lower the scalability and the noisy nodes/edges in the graphs may impact the effectiveness. From the observation that potential node mappings always appear near known corresponding nodes, we propose iMAP, a novel sub-graph expansion based alignment framework to incrementally construct meaningful sub-graphs and perform alignment on each sub-graph pair iteratively, which reduces the unnecessary computation cost in the original raw networks and improves effectiveness via excluding possible noises. Specifically, iMap builds a candidate sub-graph around known matched nodes initially. In each following iteration, iMap trains an alignment model to infer the node mapping relationship between sub-graphs, from which the sub-graphs are further extended and refined. In addition, we design a Graph Neural Network(GNN) based model named MAP on each sub-graph pair in the iMap framework. MAP utilizes trainable Multi-layer Perception (MLP) prediction heads for similarity computation and employs a mixed loss function consisting of the ranking loss for contrastive learning and the cross-entropy loss for classification. Extensive experiments conducted on real social networks demonstrate superior efficiency and effectiveness (above 12% improvement) of our proposed method compared to several state-of-the-art methods.

## CCS CONCEPTS

• **Computing methodologies** → *Supervised learning*; • **Mathematics of computing** → **Graph algorithms**.

## KEYWORDS

Network alignment, Sub-graph expansion, Graph Neural Network

**ACM Reference Format:**
Yikuan Xia , Jun Gao *, Bin Cui *. 2021. iMap: Incremental Node Mapping between Large Graphs Using GNN. In *Proceedings of the 30th ACM International Conference on Information and Knowledge Management (CIKM '21), November 1–5, 2021, Virtual Event, QLD, Australia.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3459637.3482353


*CIKM '21, November 1–5, 2021, Virtual Event, QLD, Australia*

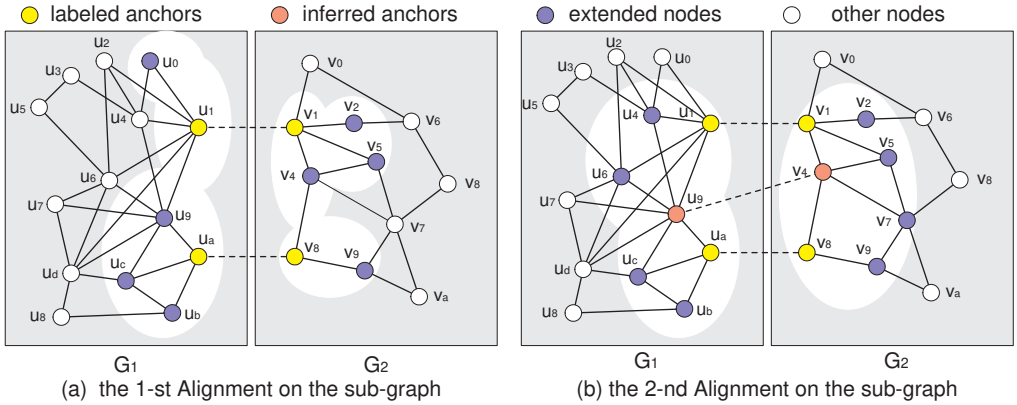
ACM ISBN 978-1-4503-8446-9/21/11…$15.00
https://doi.org/10.1145/3459637.3482353

## 1 INTRODUCTION

Graphs are widely used in modeling relations like user communication in social networks, partnership in co-author networks, or protein-protein interactions. Graphs emerged from different applications may have some common nodes (a.k.a anchors), which can be exploited to exchange information between these graphs and therefore enrich features of a single graph [43]. For example, simply mining potential friendships in separate university social network and enterprise social network may lead to incomplete recommendations only in one community, while joint mining in two networks may unlock more relationships between academia and industry. Besides recommending from multiple domains, information from a more sophisticated network can also help mining on a relatively sparse network which lacks information. Similar approaches can also be found in other fields like e-commerce, bioinformatics, or news analysis[10]. Clearly, mapping between nodes in these sources is an unavoidable step, which aims at mining potential corresponding nodes across multiple networks to benefit the downstream analysis [11]. Naturally, achieving a better alignment result can often lead to a better performance in the following joint mining task.

It is not trivial to establish an alignment between two graphs. One node's unique attributes, like the username or ID number of an account's owner in social network, or the name of people and publications in academia network, are usually not available or exist in different namespaces. For example, the node's IDs are most likely to be missing while analyzing in an anonymous social network, due to the increased public awareness of privacy protection [12, 20]. Considering labeling is an expensive operation [34], the few labeled nodes act as valuable guides in network alignment. Since aligning without any supervision in real-world noisy graphs is almost impossible, we expect the method proposed in this paper to handle graphs with minimal features (pure-structured graphs) and a limited number of anchors. At the same time, the method should naturally support the scenarios when rich features can be collected.

Earlier approaches treated the alignment problem as an optimization problem, aiming to preserve the consistency of the alignment result, which represents a basic assumption that the corresponding nodes may have similar connecting structures across different networks. While recent researchers mainly focus on solving this task via embedding-based approaches. A low-dimensional representation for each node is learned, and the alignment results can be later obtained by directly computing the similarities between the node embeddings. The network embedding approaches based on




(a) the 1-st Alignment on the sub-graph (b) the 2-nd Alignment on the sub-graph

**Figure 1: A Quick Illustration of Incremental Network Alignment.**

GNN models can more reliably and completely capture the topological information and other potential content features among networks, and meanwhile avoid the quadratic running time and space constraints of the traditional methods to some extent (i.e. maintaining a k-d tree [18]) [15, 37]. Taking the above reasons into consideration, we follow the network-embedding approaches to align two pure-structural graphs.

Despite all these efforts of the previous researches, there are still limitations of the existing approaches. First, the existing method attempts to exploit all information between two graphs, while handling the raw graph as a whole may hinder the performance. On the one hand, such methods often lead to high space and computation costs. Usually, the GNN models require $O(E)$ memory and computation time, where $E$ is the number of edges in the graph. Such requirement impacts the scalability of existing methods severely, as real-life graphs are usually extremely big. On the other hand, noises from irrelevant nodes may disturb the embeddings of anchors. For example, in Fig. 1, information passed from the irrelevant $u_7$, $u_d$ to $u_9$ may harm aligning $u_9$ with $v_4$. Directly applying a GNN based embedding algorithm on the raw graph may lead to poor performance.

Secondly, the existing network-embedding algorithms still face the shortcoming of unsuitable inductive bias. Most previous works selected one stationary function to calculate the similarities between two nodes' embeddings, such as Cosine similarity [5, 19, 26] or $L_2$-distance [18, 31, 39]. However, node embeddings from each graph may not fall into the same latent space, which are not directly suitable for cross-network similarity computation.

We observe that the anchors are major hints while establishing node mapping in the context of aligning with pure-structural information. Thus, the potential node mapping is more likely to occur near the anchors. The observation inspires us to extract a sub-graph carefully around the anchors and perform alignment between two small-scale sub-graphs instead of aligning raw graphs. As the sub-graph constructed may not contain all necessary nodes, we expect that the sub-graphs can be refined adaptively in the next round alignment according to the results of the current round. Take Fig. 1 as example, we first extract two sub-graphs (with white color) from raw graph $G_1$ and $G_2$ and perform an alignment model to infer mapped nodes in the first round. Suppose $u_9$ and $v_4$ are detected as corresponding nodes with high confidence, we refine the sub-graph in the second round. We can see that $u_9$ and $v_4$ are kept remaining in the sub-graphs, while $u_0$ in the 1-st round sub-graph does not exist in the sub-graph in the 2-nd round sub-graph, because the model may be inadequately confident of $u_0$. The time cost of performing network alignment on the sub-graphs can be significantly reduced on graphs with limited anchors, and the noisy nodes/edges in the raw graphs can be partially removed because not all nodes/edges are considered in the alignment.

The contributions of this paper are summarized as follows:

(1) We propose a framework, named iMAP, to incrementally extract the sub-graphs in multiple rounds. In each round, a random-walk-based method is adopted to select meaningful nodes according to the known anchors and inferred anchors from the previous network alignment models. Sub-graph is proposed to improve the scalability and lower the impact of noisy nodes/edges.
(2) We design a GNN-based network alignment model, called MAP, to perform network alignment on each selected sub-graphs in iMap. Different from other embedding-based methods, MAP utilizes a trainable MLP prediction head as our similarity metric and introduces a mixed loss function consisting of the ranking loss for contrastive learning and the cross-entropy loss for classification. These two strategies can be more suitable to the requirement of network alignment, leading to performance improvement.
(3) Extensive experiments demonstrate that the proposed method outperforms the current state-of-the-art approaches in efficiency and effectiveness to a large extent. Further analysis reveals that how training on raw graphs affects the alignment results, and how the proposed incremental training process helps to settle this predicament. Concrete ablation studies are also conducted to show that each component of our algorithm is indispensable for the overall advance.

We organize the rest of our paper as follows. Section 2 further explores our method's relations with related works, and Section 3 contains a detailed description of our proposed model. We provide the results of experiments and several related case studies in Section 4 to show the effectiveness of our approach and concludes our work in the last section.

## 2 RELATED WORK

We review the progress of related works, including the network alignment with rich features, optimization-solver-based alignment, network-embedding-based alignment, and the scalability of network alignment.

**Network Alignment with Rich Features.** Much work of network alignment focused on attribute-intensive scenarios, like social networks [14, 25, 42] and e-commercial platforms with relatively complete personal information [47], or co-authorship networks with rich author names and article titles [41].

**Alignment as an Optimization Problem.** When it comes to general networks with less or noisier attributes, earlier approaches like IsoRank [33] or NetAlign [2] proposed alignment consistency restraints and managed to solve the proposed optimization problem. Later variants, like FINAL [45] or REGAL [18], introduced approximate computation methods or matrix factorization to solve similar optimization problems. This class of algorithms aims to directly compute the similarity matrix between two networks, making them hard to scale to large graphs. The weak expressivity of them also hinders their performance on larger and sparser networks.

**Network Embedding based Alignment.** Network embedding means encoding the features in the original graph into low-dimen -sional node representations, from which multiple downstream tasks, including network alignment, are supported. Influential methods include matrix-factorization based approaches, like Graph Factorization [1], and random-walk based approaches, like Deepwalk [30], node2vec [16], struct2vec [32] and LINE [36]. The later proposed Graph Convolutional Network(GCN) [22] and its GNN variants, like GraphSage [17], GAT [38], served greatly as a feature encoder which aggregates each node's neighbors' information to their own. In our proposed method, we aim to employ a GNN backbone network to obtain an alignment representation.

Impacted by the development of GNN, a large number of works started to employ network-embedding into the alignment task. MAH [35] constructed hypergraphs to model higher-order relationships between nodes. IONE [26] modeled users' bi-directional relationship in social networks. PALE [27] aimed to align two networks via projecting the representation of anchors to be as close as possible. G-CREWE [31] extended REGAL and adopted graph-compression to this problem, and DALAUP [8] further adopted active learning strategies to mine more valuable anchors. Apart from all these, a series of works, like SNNA [24], DANA [19] emphasized the importance of aligning the latent spaces and employed adversarial training techniques to solve the problem. Enlightened by the ideas from contrastive learning, we introduced a trainable prediction head and a mixed loss function to achieve better results based on the network-embedding approaches.

**Scalability of Network Alignment.** Recently, people also put much effort into designing scalable alignment frameworks. Multi level-NA [46] employed a coarsen-solve-interpolate procedure to solve network alignment in a multi-level way. MGCN [5] split the networks into small partitions and obtained the joint alignment via a two-phase reconciliation process. We address the scalability problem on large graphs with few anchors by sampling a refined sub-graph around the limited anchors to avoid most unnecessary computation.

## 3 PROPOSED METHOD

In this section, we present the preliminaries of network alignment and introduce our proposed framework iMap and model MAP.

### 3.1 Preliminary

**Network Alignment.** The goal of network alignment in the supervised setting is to discover new corresponding nodes given two network structures and a bunch of pre-aligned anchors.

Formally, we represent the two graphs to be aligned as $G_1 = \{V_1, E_1\}$ and $G_2 = \{V_2, E_2\}$. $V_1, V_2, E_1, E_2$ represent the nodes and edges in $G_1, G_2$ respectively. The anchor set is noted as $S_A = \{(a_1, a_2)|a_1 \in V_1, a_2 \in V_2\}$. Our task is to find a new corresponding node set $S_T$ based on the existing one. We define $A_1 = \{a_1|a_1 \in V_1, \exists a_2 \in V_2 : (a_1, a_2) \in S_A\}$ , $A_2 = \{a_2|a_2 \in V_2, \exists a_1 \in V_1 : (a_1, a_2) \in S_A\}$ as the node sets with a known corresponding node in these two graphs.

Embedding-based alignment methods aim to learn a $d$-dimension vector representation for each graph, noted as $E_1 \in \mathbb{R}^{|V_1|\times d}$ and $E_2 \in \mathbb{R}^{|V_2|\times d}$. The embedding can be directly used to measure whether two nodes are counterparts. That is to say given a similarity calculating metric $Sim(.,.)$ and the representations $E_1, E_2$ derived from it, the alignment results can be easily calculated as $S_T = \{(a_1, a_2)|a_1 \in V_1, a_2 = \arg\max_a Sim(a1, a), a \in V_2, a \notin A_2\}$.

**Graph Neural Network.** GNN introduced neural networks to graphs, providing an end-to-end training pattern for graph-structure data. Though first enlightened by spectral methods on graphs, the computation of GNNs can be interpreted as a message-passing process [40]. In each graph neural network layer, node features are first passed through a parameterized network, and then each node aggregates and combines its features and its neighbors' features to form a hidden representation of this layer. GNNs are mostly used in tasks like node classification or link prediction. It's also a decent choice in network alignment because the learning objective force known anchors share similar embeddings, and the unknown equivalent nodes tend to aggregate the equivalent nodes' embeddings in their neighborhoods to achieve similar representations. Besides, for rich-attributed graphs, the attributes can naturally fit as the input of a GNN. Thus a well-designed GNN can leverage all structure and attribute information to embed all nodes into a low-dimensional latent space, in which similar nodes share similar embeddings.

The notations frequently used in this paper are summarized in Table 1 for a clearer presentation.

### 3.2 iMap Framework

In this section, we present our incremental sampling and training framework iMap, which is presented in Fig. 2.

As we have mentioned above, extracting a small and denser-connected sub-graph is extremely beneficial for the succeeding application of alignment algorithms. In this section, we will explain our incremental training and extracting procedure.

**Motivation.** Our starting point is that practically speaking, nodes that could be potentially mined are those closely connected with

| Notations | Meanings |
|---|---|
| $G_1, G_2$ | Graphs to be aligned |
| $V_1, V_2$ | Set of nodes in $G_1, G_2$ |
| $E_1, E_2$ | Set of edges in $G_1, G_2$ |
| $S_A$ | Set of anchor pairs |
| $S_T$ | Unknown anchor pairs to be mined |
| $A_1, A_2$ | Set of nodes in $G_1, G_2$ already aligned |
| $\|V\|$ | # of the nodes in set $V$ |
| $N(v)$ | Neighbor set of node $v$ |
| $Len_p$ | Length of a metapath $p$ |
| $d$ | Dimension of output embeddings |
| $M_1, M_2$ | Input features |
| $H_1, H_2$ | Output features |
| $G[S]$ | The sub-graph of $G$ extracted from node set $S$ |

**Table 1: Notations frequently used in the paper and their meanings.**

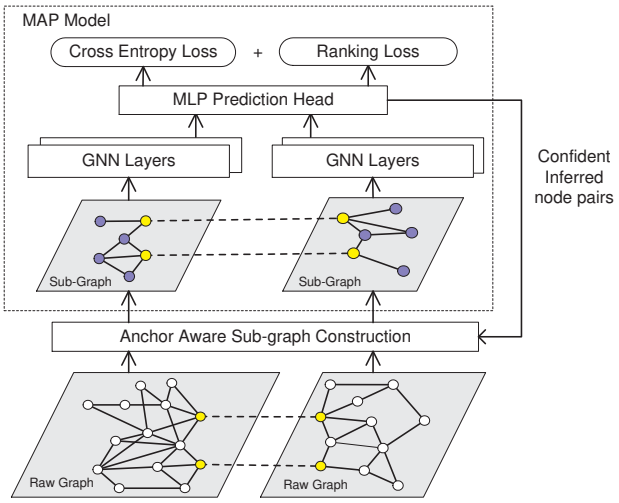



**Figure 2: The overview of our proposed iMAP framework. We construct anchor aware sub-graphs and adopt our MAP model to them. We then apply the inferred results to the next-step construction.**

the anchors. For nodes close to the anchors, information from the anchors can be easily aggregated through a GNN, leading to a more accurate alignment result. Also as we have mentioned in the introduction, extracting the nodes close to the anchors also signifies less noise introduced to the training data. In addition, anchors are often a little portion of the whole network, so extracting a sub-graph around them directly leads to a smaller and more approachable amount of computation. Thus the principle of our sampling procedure is to sample a sub-graph around the known anchors. Therefore, we call nodes that cannot be sampled in random paths near anchors noisy nodes. However, a dilemma would appear if we only sample part of the raw graph. Sampling less noisy nodes also means that we may miss the potential anchors. The dilemma demands our sampling algorithm to better distinguish the valuable anchors from the useless ones. To address this problem, we propose to expand the sub-graphs at hand according to the results of a trained model and further improve the model on the expanded graphs.

**Expanding and Selecting.** In specific, we start from the sub-graphs extracted from the anchors. In each iteration, we first follow an expansion strategy to expand a sub-graph around the existing sub-graph. For example, the strategy can be random-walk sampling, which means performing random walks from each node of the existing sub-graph and selecting the most frequently arising nodes to add to the expanded sub-graph. The expanding budget $B$ of this step should be relatively higher to include more potential anchors.

Next, a model outputting the similarities between node pairs will be trained on the expanded sub-graphs. We then select the most confident nodes from each sub-graph according to our heuristic rule. Confident here can be defined as:

$$Conf(v) = \max_i Sim(v, i) \tag{1}$$

which means selecting the nodes whose predicted counterpart has the highest similarity with it. We set a relatively small selecting budget $b$ here to ensure that the selected nodes in this step are probably anchors, to make the expansion of the next iteration more accurate. The two steps are conducted alternately until an expected whole budget is reached.

**Incremental Training.** Sampling in the above iterative manner means multiple training times of the alignment model, which may lead to corresponding increase of running time. Noticing that the sub-graphs are expanded step by step, the parameters of each step's model may be reused to some extent. Based on this, we propose to incrementally train our model in the above iterations, sharing the parameters of our model throughout the whole training process. The training epochs should be relatively higher in the first few iterations to enforce the model to reach a meaningful state, while the later iterations require fewer epochs for fine-tuning. The iMap framework is summarized in Algorithm 1.

**Algorithm 1** iMap Framework.

**Input:** Graph $G_1, G_2$, Anchor Set $S_A$, Known Node Set $A_1, A_2$, Expand Budget $B$, Select Budget $b$, Expected Budget $B\_E$, Network Alignment Model *Model*;
**Output:** Predicted Anchor Set $S_T$;

```
Set the initial sub-graphs Sub_1 = G_1[A_1], Sub_2 = G_2[A_2];
Set S_T = ∅;
while |S_T| < B_E do
    Expand Sub_1, Sub_2 to Sub_1', Sub_2' using random walk under budget B;
    Train Model with Sub_1', Sub_2' , get similarity matrix Sim;
    Update Sub_1, Sub_2 according to equation 1, Sim and budget b;
    Update S_T with Sim;
return S_T;
```

## 3.3 MAP Model

In this section, we propose our GNN-based MAP model for network alignment on fixed graphs.

**Input features.** The quality of input features for a GNN is quite essential for the model's performance. When we train from pure structure, some previous researches [39] set the input features

as trainable parameters. However, it would make the model hard to train, since the pure structural signal is quite weak, let alone the extra training cost led by the additional parameters. So we propose pre-training a Metapath2Vec [9] embedding to feed into the following GNN. We add virtual edges between the anchors and define the metapath guiding the random walk sampling in Metapath2Vec as:

$$p = u \xleftrightarrow{E_1} \dots \xleftrightarrow{E_1} u \xleftrightarrow{E_1} u \xleftrightarrow{Virtual} v \xleftrightarrow{E_2} v \xleftrightarrow{E_2} \dots \xleftrightarrow{E_2} v$$

and its reverse pattern. The number of nodes on one side of the virtual edge is half of the metapath's length $Len_p$ to make the pattern symmetric. The paths sampled according to this metapath and the Word2vec [28] model applied to it demand the initial representations for the anchors to be close, and the nodes in the neighborhood of the anchors to be close either. This provides a decent initial input for the model, the attention mechanism introduced afterward cooperating with this feature can also help the model focus more on the relations with the anchors and their neighbors, leading to more accurate message passing.

**Graph Attention Network Backbone.** Graph Attention Network (GAT) [38] introduced self-attention mechanism to learn the importance of node $i \in v \cup N(v)$ to every node $v$. Each graph attention aggregation module updates the hidden representations and aggregates information from every node's neighborhood proportional to the learnable attention coefficients. Multiple attention aggregation modules are concatenated to form one graph attention layer, and several layers are stacked to form the whole network.

In detail, the $l$-th graph attention layer for a graph $G = \{V, E\}$ takes the hidden representation for the last layer $H^{l-1} \in \mathbb{R}^{|V| \times d_{in}}$ as input, and outputs a representation $H^l \in \mathbb{R}^{|V| \times d_{out}}$. The aggregation operation for an arbitrary node $i$ in one GAT module can be represented as:

$$\vec{h_i^l} = \sigma\Big(\sum_{j \in \{i\} \cap N(i)} \alpha_{ij} \vec{h_j^{l-1}} W^l\Big) \tag{2}$$

Here, $\sigma$ is a nonlinearity function (i.e. ReLU), $\vec{h_i^l} \in \mathbb{R}^{1 \times d_{in}}$ represents the embedding for node $i$. $W^l$ represents the trainable parameter matrix of this module in the $l$-th layer, and $\alpha_{ij}$ represents the attention coefficient between node $i$ and $j$, which is computed as follows:

$$\alpha_{ij} = softmax_j(att(\vec{h_i} W^l, \vec{h_j} W^l)) \tag{3}$$

Here, $att$ can be any attention mechanism, and the $softmax$ function regularizes the attention coefficients among node $i$'s neighborhood.

More specifically, we propose using a 2-layer GAT model to encode the input structural features. Previous works reveal that the closeness of the latent space of the final embeddings greatly affects the alignment quality [19, 24, 27]. Therefore we let the GAT models for the two graphs share parameters among the first layer to satisfy the above assertion. Sharing the parameters also reduces the size of our model, making our model more favorable to training.

We define the first shared GAT layer as $GAT^1$, and the second GAT layers for $G_1, G_2$ as $GAT_1^2, GAT_2^2$ respectively. $GAT_1^2, GAT_2^2$, as the last layer, runs without nonlinearities. Suppose the input features we obtain from $G_1, G_2$ are $M_1, M_2$, the output embeddings $H_1, H_2$ derived from our model are:

$$\begin{aligned} H_1 &= GAT_1^2(GAT^1(M_1)) \\ H_2 &= GAT_2^2(GAT^1(M_2)) \end{aligned} \tag{4}$$

**Similarity Metric.** Previous works mostly chose a fixed similarity metric. The most commonly used similarity functions are L2-norm [18, 31, 39], $Sim(\vec{x}, \vec{y}) = -|\vec{x} - \vec{y}|_2$, inner product similarity, $Sim(\vec{x}, \vec{y}) = \vec{x} \cdot \vec{y}$, and Cosine similarity [5, 19, 26], $Sim(\vec{x}, \vec{y}) = \frac{\vec{x} \cdot \vec{y}}{|\vec{x}||\vec{y}|}$. To solve the drawbacks of an immobile similarity function, we propose using a parameterized similarity function. One chief advantage of using parameterized similarity metric is that the two latent spaces embedded from the two graphs are most likely in disorder. Projecting one latent space to the other is the simplest solution. Take the inner product similarity as example, the introduced projection means:

$$Sim(\vec{x}, \vec{y}) = Sim(\vec{x} W_P, \vec{y}) = \vec{x} W_P \vec{y} \tag{5}$$

in which $W_P \in \mathbb{R}^{d_x \times d_y}$ is the projection matrix, and $d_x, d_y$ denotes dimension of $\vec{x}, \vec{y}$ respectively.

Inspired by the recent progress in contrastive learning [6, 7], we introduce a bidirectional MLP prediction head to generalize the equation 5:

$$\begin{aligned} Sim(\vec{x}, \vec{y}) &= MLP_{pred}(\vec{x}) \cdot \vec{y} + MLP_{pred}(\vec{y}) \cdot \vec{x} \\ Prob(\vec{x}, \vec{y}) &= (\sigma(MLP_{pred}(\vec{x}) \cdot \vec{y}) + \sigma(MLP_{pred}(\vec{y}) \cdot \vec{x}))/2 \end{aligned} \tag{6}$$

Here we provided an additional probability definition between nodes, which can also be regarded as a kind of similarity. In the above equation, $MLP_{pred}$ represents a simple feed-forward MLP network, and $\sigma$ is a nonlinearity which maps the inner product similarity into the probability scale $[0, 1]$, such as $Sigmoid$, so as to combine the bidirectional similarities into one with a simple average operation.

**Loss Design.** In order to guide the GAT model to embed equivalent entities into close representations in the latent space, the optimization goal, a.k.a loss, has to be carefully designed. Viewing network alignment as a binary classification task or link prediction task for anchor pairs brings out the cross-entropy loss, which can be formulated as follows:

$$\begin{aligned} L_1 = \sum_{i \in A_1} \Big( & \sum_{(i,j) \in S_A} -log(Prob(\vec{h_{1,i}}, \vec{h_{2,j}})) \\ & + \sum_{(i,k) \notin S_A} -log(1 - Prob(\vec{h_{1,i}}, \vec{h_{2,k}})) \Big) \end{aligned} \tag{7}$$

Here, $\sum_{(i,k) \notin S_A}$ is actually implemented by negative sampling from the node pairs excluded from the anchor pairs. Both corrupting one node from every element from $S_A$ and directly sampling from the complementary set of $S_A$ are acceptable negative sampling methods. From the other perspective, the task of network alignment actually only requires the similarity between equivalent nodes to be higher than the similarities between the irrelevant nodes. The widely-used margin-ranking loss for network alignment can be formulated as:

$$\begin{aligned} L_2 = \sum_{i \in A_1} \sum_{(i,j) \in S_A} \sum_{(i,k) \notin S_A} max(0, & -Sim(\vec{h_{1,i}}, \vec{h_{2,j}}) + \eta \\ & + Sim(\vec{h_{1,i}}, \vec{h_{2,k}})) \end{aligned} \tag{8}$$

Here, $\eta$ denotes the margin hyperparameter, which represents how far the positive and negative samples should be separated.

Conclusively, the two losses are combined to form the optimization goal $L$:

$$L = L_1 + \gamma_1 L_2 \tag{9}$$

Here, $\gamma_1$ is the hyper-parameter to balance the two losses. An Adam optimizer [21] is adopted to minimize the above loss function.

The MAP algorithm is summarized in Algorithm 2.

**Algorithm 2** MAP Algorithm.

**Input:** Graph $G_1, G_2$, Anchor Set $S_A$, Hyperparameters;
**Parameter:** Encoders $\theta_1 = GAT^1, GAT_1^2, \theta_2 = GAT^1, GAT_2^2$, Prediction head $\theta_{MLP}$;
**Output:** Similarity Matrix $Sim$, Network Embedding $H_1, H_2$;

```
Initialize the parameters θ1, θ2, θ_MLP;
for epoch = 1 to training epoch do
    Negative Sample (i, k) ∉ S_A;
    Sample Sam1, Sam2;
    Calculate H1, H2 using equation 2-4;
    Calculate L1, L2, L using equation 5-9;
    Update θ1, θ2, θ_MLP with Adam Optimizer to minimize L;
Calculate the final H1, H2 using equation 2-4;
Calculate Sim using equation 5-6;
return Sim, H1, H2;
```

## 4 EXPERIMENTS

In this section, we evaluate MAP and iMap on various real-world and synthetic datasets and provide case studies and ablation studies to penetrate into the effects of each component in our framework.

### 4.1 Experiment Setup

**Datasets.** We evaluate MAP and iMap on several public real-world datasets published in previous researches in network alignment or joint-network data mining. The statistics of the datasets we used are summarized in Table 2. The Douban&Weibo dataset (Db↔Wb) [3], the Foursquare&Twitter dataset (Fs↔Tw) [44], and the Douban&Weibo_Large dataset ($\text{DB}_L \leftrightarrow \text{Wb}_L$) [4] were constructed from several widely used social networks, while the ground truth anchor links were obtained from a third party social network where a person records both accounts on, or from a web crawler. The Soc_Pokec dataset [23] is a large-scale dataset from the most popular online social network in Slovakia. A sub-graph is later extracted from this dataset to simulate the aligning process between a newly-established social network and this giant one.

| $G_1 \leftrightarrow G_2$ | $\lvert V_1\rvert \leftrightarrow \lvert V_2\rvert$ | $\lvert E_1\rvert \leftrightarrow \lvert E_2\rvert$ | #Anchors |
|---|---|---|---|
| Db ↔ Wb | 10k↔ 9k | 527k↔270k | 4,752 |
| Fs↔ Tw | 5k↔5k | 76k↔164k | 3,148 |
| $\text{Db}_L \leftrightarrow \text{Wb}_L$ | 96k↔133k | 27,00k↔6,280k | 93k |
| Soc_Pokec | 101k↔1,632k | 1,179k↔22,301k | 100k |

**Table 2: The statistics of the datasets.**

**Baselines.** We compare our proposed MAP and iMap with the following methods:

(1) **IONE [26].** This method (IJCAI 16') proposed to learn a network's embedding through exploiting each node's bidirectional relationships with its neighbors.
(2) **PALE [27].** This method (IJCAI 16') obtains anchor predictions via learning a projection function to minimize the distance between the embeddings of the anchors.
(3) **GCN-Align [39].** This method (EMNLP 18') uses a GCN to encode two networks.
(4) **DANA [24].** This method (TKDE 20') considers aligning the latent space via an adversarial training approach.
(5) **MGCN [5].** This method (KDD 20') proposed a multi-level GCN and a partition and reconcile tactic to align on given graphs.

The first four baselines are pure network alignment algorithms, while the latter one involves aligning on sub-graphs. So we evaluate the effectiveness of our MAP algorithm comparing with the first four methods, and we later conducted experiments to show the superiority of our proposed incremental iMap framework.

**Parameter Settings.** The dimension of the final embeddings on the dataset is set to 128. It's worth noting that for IONE, DANA, and our proposed MAP model, which take bidirectional embedding into account, the dimension of embeddings from each direction is set to half of the above-mentioned dimensions for the sake of fair comparison. 80% percent of the observed anchors are set as training seeds, and the rest 20% remain as testing data. The MLP prediction head in our proposed Map model is a vanilla 2-Layer MLP with equal input and output length and twice the length as the dimension of the hidden layer. The coefficient $\gamma_1$ is set as 4, and the margin $\eta$ in our margin-ranking loss is set to 0.1. We set the expanding budget $B$ to 10 times the selecting budget $b$, and the total iterations in iMap to 10. We train our model using an Adam optimizer with a learning rate of 1e-3 and $\lambda$=1e-3 for regularization, and the total training epoch is set to 2000. For the baseline models involving GNN, the number of GNN layers are all set as 2, and the GNNs are trained in a full batch manner. All the competing baselines are trained under the hyperparameters described in their papers, or via the open-source codes.

**Evaluation Metrics.** We can either treat the problem as a binary classification task predicting the anchor links or as a ranking problem. From the first perspective, we adopt the widely used classification metric, prediction accuracy [5, 26, 27]. The ratio of positive and negative pairs for testing is set to 1:1. If we view it as a ranking problem, we evaluate the performance of our method and the baselines using the Hits@k metric [19, 24, 26, 29, 31]. Hits@k as defined as:

$$Hits@k = \frac{\#hits@k}{|S_T|} \tag{10}$$

in which $\#hits@k$ represents the number of nodes whose top-$k$ candidates of anchor predictions from the evaluating model contain its exact counterpart. This metric is usually computed bidirectional from the two graphs and the reported metric is averaged. It's obvious that the higher the two metrics are, the better the model performs.

**Machines.** All the methods are implemented on a Linux server deployed with a Ubuntu 20.04.2 LTS operating system. The server is equipped with a RTX-3060 GPU, an Intel Xeon Silver 4210R @ 2.40GHz CPU, and a 256G RAM.

## 4.2 Experiments of MAP

We first compare our proposed MAP model with the baselines on two relatively small real-world datasets to evaluate its effectiveness.

*4.2.1 Performance Analysis.* We evaluate the performance of our proposed MAP model on the Douban
&Weibo and Foursquare&Twitter datasets in this section. The observed results are presented in Table 3.

Several conclusions can be made from the above results. Firstly, our proposed MAP model significantly outperforms the baselines on these two datasets. A 12.8% and 14.3% improvement can be observed on the Douban&Weibo and Foursquare&Twitter datasets respectively, compared to the second-best baselines in terms of the *Hits*@30 metric.

Secondly, the earlier network embedding approach PALE has a relatively unsatisfying result, since purely learning a projection function between latent spaces is not capable of learning an accurate alignment result only using structural information. The other models all adopted an encoding and similarity computation framework. The promising results of the other models indicate the effectiveness of adopting GNNs to this problem as an encoding sub-unit.

Thirdly, our model's advantage against GCN-Align and MGCN suggests that employing GAT as the encoding layer has huge superiority in performance compared to using GCN, and adopting a pre-trained input feature is far better than setting them as trainable parameters.

Last, IONE, as a relatively earlier approach has comparable performance with the later GNN approaches. One possible reason is that IONE defines the input and output vector of a node to represent a node's role as follower and followee. It's quite similar to our proposed bidirectional similarity calculating function, which may contribute much to its fine results.

*4.2.2 Effect of anchor link percentage.* A model's sensitivity to the ratio of training anchor links is very crucial to evaluating a model. We select two most competitive baselines to compare with MAP. The results of models trained under different ratios of anchors is presented in Fig. 3. We can imply that our MAP model consistently outperforms the other two competitive methods, which shows the robustness of our MAP model.

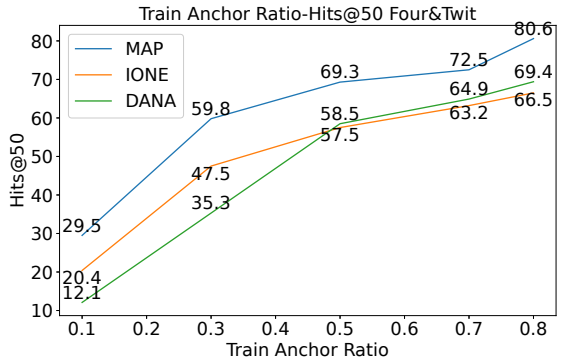



**Figure 3: Results of different ratio of train anchor set in Foursquare&Twitter**

*4.2.3 Ablation study of MAP.* We present the results of the ablation study of MAP in Table 4, Figure 4.

We remove three important components of our proposed MAP model. In Table 4, MAP(-p) means removing the MLP **p**rediction head in our similarity calculation function, MAP(-r) means removing the **r**anking loss, and MAP(-m) means replacing the pre-trained **m**etapath2vec features with random features. We can easily imply that each of our components has irreplaceable benefits to our whole MAP model.

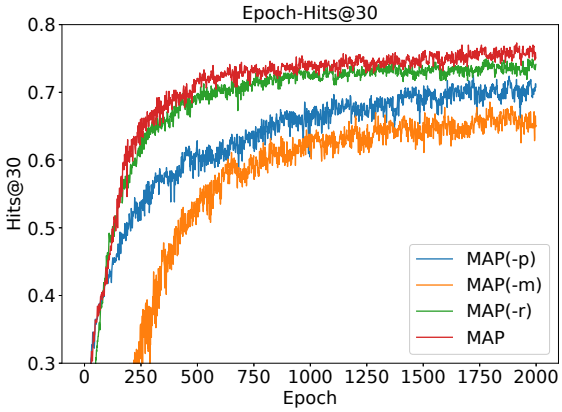



**Figure 4: The convergence comparison of MAP models before and after removing components on Foursquare&Twitter.**

The results on the training set are plotted in Fig. 4. We observe that the pre-trained features are very crucial to the final results because it provides a fine initial representation for each node making the training process much easier. The introduction of the MLP prediction head greatly accelerates the convergence process and the final performance peak. The application of ranking loss also slightly increases the convergence speed and the alignment results.

*4.2.4 Effect of pre-trained features.* We present the performance of different combinations of input features and models in Table 5. MAP(n) represents replacing the input features of MAP with pre-trained node2vec features, and GCN(m) represents replacing the backbone network of MAP with a simple 2-layer GCN. We can infer that the carefully designed metapath2vec features perform better than node2vec features trained separately, and metapath2vec features alone cannot fully explain the success of MAP. The GAT backbone network also matters.

## 4.3 Experiments of iMap

In this sub-section, we first provide two case studies to show the validity of our motivation. Further experiment results are then presented to show the advantage of our iMap framework in effectiveness and efficiency.

*4.3.1 A case study of the importance of eliminating the noisy nodes.* As we have mentioned above, the noises from irrelevant nodes may harm the performance of GNN-based models. On the Foursquare& Twitter dataset, we randomly remove different ratios of irrelevant nodes (nodes outside of the anchor set), and the performance of our MAP model is presented in Fig. 5. We can clearly see that the more noisy nodes we remove, the higher our model's performance would be. Test results in the training process are shown in Fig. 6.

| Dataset | Method | Hits@1 | Hits@5 | Hits@10 | Hits@30 | Hits@50 |
|---|---|---|---|---|---|---|
| Douban & Weibo | MAP(Ours) | 10.1 | **27.4** | **36.6** | **50.8** | **59.8** |
| | IONE(16) | 11.5 | 22.2 | 27.7 | 38.0 | 42.4 |
| | GCN-Align(18) | 4.5 | 12.3 | 18.5 | 31.2 | 38.5 |
| | PALE(16) | 7.8 | 15.5 | 20.3 | 29.6 | 33.4 |
| | DANA(20) | **12.1** | 22.8 | 27.9 | 37.5 | 41.3 |
| | MGCN(20) | 8.6 | 18.7 | 24.2 | 30.9 | 36.6 |
| Foursquare & Twitter | MAP(Ours) | **29.6** | **56.9** | **66.3** | **77.1** | **80.6** |
| | IONE(16) | 26.9 | 45.1 | 51.9 | 62.8 | 66.5 |
| | GCN-Align(18) | 13.8 | 39.4 | 50.3 | 62.7 | 70.4 |
| | PALE(16) | 11.7 | 31.2 | 43.9 | 52.3 | 55.4 |
| | DANA(20) | 24.4 | 43.7 | 51.6 | 64.4 | 69.4 |
| | MGCN(20) | 18.1 | 40.5 | 48.1 | 56.5 | 63.1 |

**Table 3: Performance Comparison on fixed graphs.**

| Method | Foursquare&Twitter | Douban&Weibo |
|---|---|---|
| MAP | **77.1** | **41.9** |
| MAP(-p) | 72.5 | 35.3 |
| MAP(-r) | 74.9 | 31.2 |
| MAP(-m) | 64.5 | 26.4 |

**Table 4: Ablation study of MAP(Metric:Hits@30)**

| Method | Foursquare&Twitter | Douban&Weibo |
|---|---|---|
| MAP | **77.1** | **41.9** |
| MAP(n) | 72.5 | 33.8 |
| GCN(m) | 56.1 | 27.4 |

**Table 5: Effect of pre-trained features(Metric:Hits@30)**

We can see that training on the filtered dataset can lead to higher performance as well as faster convergence speed. The importance of eliminating the noisy nodes demonstrates the correctness of the design principle of our iMap framework.

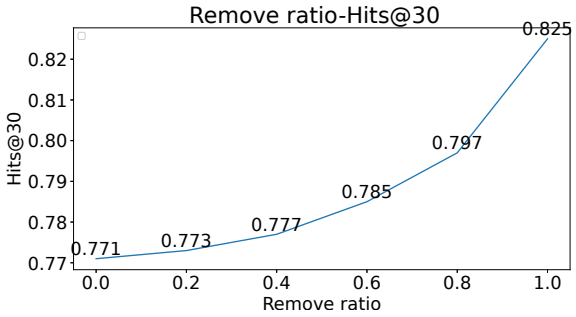



**Figure 5: The Hits@30 comparison of MAP models after removing different ratio of noisy nodes.**

*4.3.2 A case study of the distribution of correctly aligned nodes.* One of our motivations is that the potential corresponding nodes that can be correctly aligned via a GNN model are closely connected with the anchors. We simply define a node $v$'s connectivity as:

$$Conn(v) = \frac{|(Nei_1(v) \cap A_1) \cap (Nei_2(v) \cap A_2)|}{(|Nei_1(v) \cap A_1| + |Nei_2(v) \cap A_2|)/2} \quad (11)$$

We count the number of nodes of different connectivity and observe different distributions of them in correctly and falsely aligned node sets on the Douban&Weibo dataset, and the results are presented in Fig. 7. Statistics show that the average connectivity of falsely aligned nodes is only 0.13, while the average connectivity

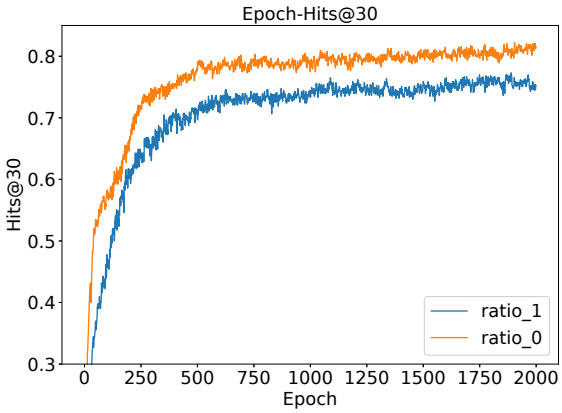



**Figure 6: The convergence comparison of MAP model before and after elimination of irrelevant nodes.**

| Metric | Method | Douban&Weibo_large | Soc_Pokec |
|---|---|---|---|
| Hits@30 | iMap(Ours) | **16.7** | **66.6** |
| | IONE(16) | 15.2 | 39.4 |
| | MGCN(20) | 13.1 | OOM |
| Accuracy | iMap(Ours) | **77.9** | **98.8** |
| | IONE(16) | - | - |
| | MGCN(20) | 66.4 | OOM |

**Table 6: Comparison of iMap**

of correctly aligned nodes can reach 0.33. Blindly expanding the sub-graph not only introduces noisy nodes but also brings about node pairs unpractical for the model to recognize.

*4.3.3 Performance Analysis.* We evaluate the performance of our proposed iMap framework with the baselines on two large-scale datasets. The results are presentedb in Table 6. The fifth line of Table 6 are empty because IONE treat network alignment only as a mapping probelm, which caused the missing of the other metric. We can imply that iMap can not only outperform the existing methods in effectiveness but also handle million-scale graphs which vanilla GNN-based models, like MAP can't handle. As you can see, training MGCN with the open-source code will cause GPU out of memory in the large Soc-Pokec dataset.

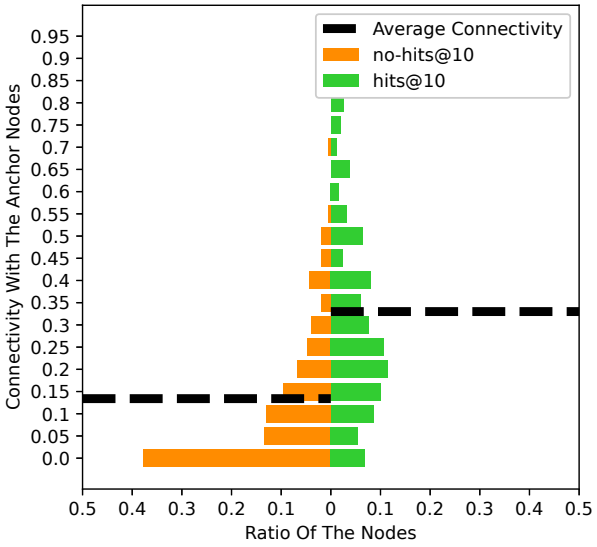



**Figure 7: The ratio of nodes of different connectivity with anchors in correctly and falsely aligned nodes.**

*4.3.4 Scalability Analysis.* The scalability of our iMap is discussed in this subsection. When it comes to time consumption, we construct a synthetic dataset to illustrate each method's scalability. The graphs in this dataset are sparse, and the average node's degree is set to 20. The anchor set is set to 1/10 of the graph size to simulate real scenarios. We set the average training time of IONE to 1/10 of its actual training time because it usually takes 1/10 of the GNN models' training epochs to reach convergence. The results are plotted in Fig. 8.

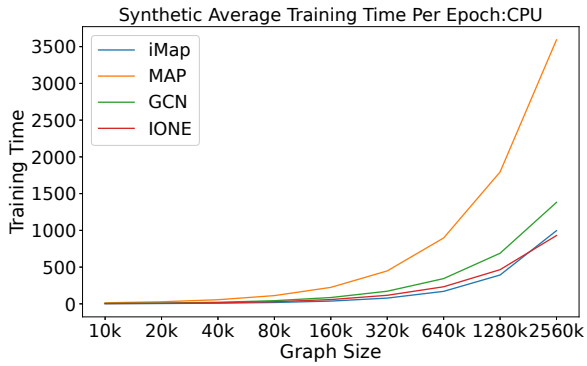



**Figure 8: Scalability analysis of time**

We can imply from Fig. 8 that adopting the iMap framework can save approximately 4/5 training time compared to the GAT-based MAP model. Training on the relatively small sub-graphs contributes the most to iMap's scalability. The reason why iMap exceeds IONE at the end of the chart is that the quadratic similarity computation process in the selecting and refining process starts to rise when the graph becomes extra-large. This can be relieved by introducing strategies like LSH [13], k-d tree in the future.

When it comes to memory constraints, GPU memory is usually more badly needed. Training a 128-dim GAT-based alignment model on graph pairs of size 10,000 in a full-batch manner may take up to 10G GPU memory. In substitution, training on its small sub-graph of size 1,000 according to iMap takes much less GPU memory, which may scale the input graph to 10 times bigger.

*4.3.5 A case study of the expanding process.* We present the expanding process of one training process in the Soc_Pokec dataset in Table 7. The second column records the covered anchors after each iterations' expansion. The third and fourth column record the Hits@30 metric before and after each iterations' training process evaluated on the sub-graph. The fifth column records the anchors covered after the selecting process, and the last column records the Hits@30 metric on the whole graph. We can imply from the statistics that, firstly, the anchors covered in the expansion process is quite crucial for the overall performance. The Hits@30 metric on the sub-graph can easily reach up to 90 via the MAP model, so the proportion of the successfully expanded anchors would dominate the final performance. Secondly, our iMap frameworks can consistently select the anchors according to the fifth column. Another fact is that the pure random-walk expansion can only cover 14,503 anchors, which is less than the iMap framework. These all indicate the effectiveness of our iMap framework. Last, we can imply that after the first long training epochs, the evaluation metric before each iteration's training doesn't decline much, which totally supports our incremental training assumption.

| Iter | Ex Anc. | Hits Bef. | Hits Af. | Sel Anc. | Hits All |
|---|---|---|---|---|---|
| 1 | 13,748 | 5e-3 | 91.6 | 711 | 61.0 |
| 2 | 13,715 | 89.5 | 91.9 | 1,349 | 62.3 |
| 3 | 13,999 | 90.6 | 92.6 | 2,036 | 64.0 |
| 4 | 14,202 | 90.4 | 91.6 | 2,623 | 64.3 |
| 5 | 14,335 | 89.3 | 91.7 | 3,374 | 65.0 |
| 6 | 14,503 | 89.1 | 90.5 | 3,954 | 64.8 |
| 7 | 14,648 | 88.8 | 91.0 | 4,576 | 65.9 |
| 8 | 14,805 | 88.2 | 90.8 | 5,244 | 66.4 |
| 9 | 14,909 | 88.8 | 90.6 | 5,864 | 66.7 |
| 10 | 15,067 | 88.2 | 89.4 | 6,411 | 66.6 |

**Table 7: Statistics of one iMap training process**

## 5 CONCLUSION

In this paper, we propose our iMap framework for network alignment. By incrementally constructing meaningful sub-graphs and training on them, our method solidly outperforms the state-of-the-art approaches in effectiveness and efficiency. We also introduce a mix loss function and a MLP prediction head to boost the performance of GNN models for fixed network alignment. Additionally, several case studies provide some new insights in the difficulties of network alignment in large graphs with limited anchors. The future work might be further discovering heuristic rules or self-adaptive strategies for mining the closely related sub-graphs, or generalizing this method to heterogeneous graphs, which has wider applications.

## ACKNOWLEDGMENTS

This work is supported by NSFC under Grant No. 61832001.

## REFERENCES


[1] Amr Ahmed, Nino Shervashidze, Shravan Narayanamurthy, Vanja Josifovski, and Alexander J Smola. 2013. Distributed large-scale natural graph factorization. In *Proceedings of the 22nd international conference on World Wide Web*. 37–48.

[2] Mohsen Bayati, Margot Gerritsen, David F Gleich, Amin Saberi, and Ying Wang. 2009. Algorithms for large, sparse network alignment problems. In *2009 Ninth IEEE International Conference on Data Mining*. IEEE, 705–710.

[3] Xuezhi Cao and Yong Yu. 2016. ASNets: A benchmark dataset of aligned social networks for cross-platform user modeling. In *Proceedings of the 25th ACM International on Conference on Information and Knowledge Management*. 1881–1884.

[4] Xuezhi Cao and Yong Yu. 2016. BASS: A bootstrapping approach for aligning heterogenous social networks. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*. Springer, 459–475.

[5] Hongxu Chen, Hongzhi Yin, Xiangguo Sun, Tong Chen, Bogdan Gabrys, and Katarzyna Musial. 2020. Multi-level graph convolutional networks for cross-platform anchor link prediction. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 1503–1511.

[6] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. 2020. A simple framework for contrastive learning of visual representations. In *International conference on machine learning*. PMLR, 1597–1607.

[7] Xinlei Chen and Kaiming He. 2020. Exploring Simple Siamese Representation Learning. *arXiv preprint arXiv:2011.10566* (2020).

[8] Anfeng Cheng, Chuan Zhou, Hong Yang, Jia Wu, Lei Li, Jianlong Tan, and Li Guo. 2019. Deep active learning for anchor user prediction. *arXiv preprint arXiv:1906.07318* (2019).

[9] Yuxiao Dong, Nitesh V Chawla, and Ananthram Swami. 2017. metapath2vec: Scalable representation learning for heterogeneous networks. In *Proceedings of the 23rd ACM SIGKDD international conference on knowledge discovery and data mining*. 135–144.

[10] Y. Duan, A. Jatowt, S. S. Bhowmick, and M. Yoshikawa. 2019. Mapping Entity Sets in News Archives Across Time. *Data Science and Engineering* 4, 3 (2019), 208–222.

[11] Frank Emmert-Streib, Matthias Dehmer, and Yongtang Shi. 2016. Fifty years of graph matching, network alignment and network comparison. *Information sciences* 346 (2016), 180–197.

[12] Joshua Fogel and Elham Nehmad. 2009. Internet social network communities: Risk taking, trust, and privacy concerns. *Computers in human behavior* 25, 1 (2009), 153–160.

[13] Aristides Gionis, Piotr Indyk, and Rajeev Motwani. 1999. Similarity Search in High Dimensions via Hashing. In *Proceedings of the 25th International Conference on Very Large Data Bases (VLDB '99)*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 518–529.

[14] Oana Goga, Patrick Loiseau, Robin Sommer, Renata Teixeira, and Krishna P Gummadi. 2015. On the reliability of profile matching across large online social networks. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 1799–1808.

[15] Palash Goyal and Emilio Ferrara. 2018. Graph embedding techniques, applications, and performance: A survey. *Knowledge-Based Systems* 151 (2018), 78–94.

[16] Aditya Grover and Jure Leskovec. 2016. node2vec: Scalable feature learning for networks. In *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*. 855–864.

[17] William L Hamilton, Rex Ying, and Jure Leskovec. 2017. Inductive representation learning on large graphs. *arXiv preprint arXiv:1706.02216* (2017).

[18] Mark Heimann, Haoming Shen, Tara Safavi, and Danai Koutra. 2018. Regal: Representation learning-based graph alignment. In *Proceedings of the 27th ACM international conference on information and knowledge management*. 117–126.

[19] Huiting Hong, Xin Li, Yuangang Pan, and Ivor Tsang. 2020. Domain-adversarial Network Alignment. *IEEE Transactions on Knowledge and Data Engineering* (2020).

[20] David J Houghton and Adam N Joinson. 2010. Privacy, social network sites, and social relations. *Journal of Technology in Human Services* 28, 1-2 (2010), 74–94.

[21] Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* (2014).

[22] Thomas N Kipf and Max Welling. 2016. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907* (2016).

[23] Jure Leskovec and Andrej Krevl. 2014. SNAP Datasets: Stanford large network dataset collection.

[24] Chaozhuo Li, Senzhang Wang, Yukun Wang, Philip Yu, Yanbo Liang, Yun Liu, and Zhoujun Li. 2019. Adversarial learning for weakly-supervised social network alignment. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 33. 996–1003.

[25] Jing Liu, Fan Zhang, Xinying Song, Young-In Song, Chin-Yew Lin, and Hsiao-Wuen Hon. 2013. What's in a name? An unsupervised approach to link users across communities. In *Proceedings of the sixth ACM international conference on Web search and data mining*. 495–504.

[26] Li Liu, William K Cheung, Xin Li, and Lejian Liao. 2016. Aligning Users across Social Networks Using Network Embedding.. In *Ijcai*. 1774–1780.

[27] Tong Man, Huawei Shen, Shenghua Liu, Xiaolong Jin, and Xueqi Cheng. 2016. Predict anchor links across social networks via an embedding approach.. In *Ijcai*, Vol. 16. 1823–1829.

[28] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781* (2013).

[29] Shichao Pei, Lu Yu, Robert Hoehndorf, and Xiangliang Zhang. 2019. Semi-Supervised Entity Alignment via Knowledge Graph Embedding with Awareness of Degree Difference. In *The World Wide Web Conference* (San Francisco, CA, USA) *(WWW '19)*. Association for Computing Machinery, New York, NY, USA, 3130–3136. https://doi.org/10.1145/3308558.3313646

[30] Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. 2014. Deepwalk: Online learning of social representations. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*. 701–710.

[31] Kyle K Qin, Flora D Salim, Yongli Ren, Wei Shao, Mark Heimann, and Danai Koutra. 2020. G-CREWE: Graph CompREssion With Embedding for Network Alignment. In *Proceedings of the 29th ACM International Conference on Information & Knowledge Management*. 1255–1264.

[32] Leonardo FR Ribeiro, Pedro HP Saverese, and Daniel R Figueiredo. 2017. struc2vec: Learning node representations from structural identity. In *Proceedings of the 23rd ACM SIGKDD international conference on knowledge discovery and data mining*. 385–394.

[33] Rohit Singh, Jinbo Xu, and Bonnie Berger. 2007. Pairwise global alignment of protein interaction networks by matching neighborhood topology. In *Annual International Conference on Research in Computational Molecular Biology*. Springer, 16–31.

[34] Yi Song, Panagiotis Karras, Qian Xiao, and Stéphane Bressan. 2012. Sensitive label privacy protection on social network data. In *International Conference on Scientific and Statistical Database Management*. Springer, 562–571.

[35] Shulong Tan, Ziyu Guan, Deng Cai, Xuzhen Qin, Jiajun Bu, and Chun Chen. 2014. Mapping users across networks by manifold alignment on hypergraph. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 28.

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

[37] Huynh Thanh Trung, Nguyen Thanh Toan, Tong Van Vinh, Hoang Thanh Dat, Duong Chi Thang, Nguyen Quoc Viet Hung, and Abdul Sattar. 2020. A comparative study on network alignment techniques. *Expert Systems with Applications* 140 (2020), 112883.

[38] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. 2017. Graph attention networks. *arXiv preprint arXiv:1710.10903* (2017).

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

[40] Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and S Yu Philip. 2020. A comprehensive survey on graph neural networks. *IEEE transactions on neural networks and learning systems* (2020).

[41] Fanjin Zhang, Xiao Liu, Jie Tang, Yuxiao Dong, Peiran Yao, Jie Zhang, Xiaotao Gu, Yan Wang, Bin Shao, Rui Li, et al. 2019. Oag: Toward linking large-scale heterogeneous entity graphs. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 2585–2595.

[42] Jing Zhang, Bo Chen, Xianming Wang, Hong Chen, Cuiping Li, Fengmei Jin, Guojie Song, and Yutao Zhang. 2018. Mego2vec: Embedding matched ego networks for user alignment across social networks. In *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*. 327–336.

[43] Jiawei Zhang, Xiangnan Kong, and S Yu Philip. 2013. Predicting social links for new users across aligned heterogeneous social networks. In *2013 IEEE 13th International Conference on Data Mining*. IEEE, 1289–1294.

[44] Jiawei Zhang and S Yu Philip. 2015. Integrated anchor and social link predictions across social networks. In *Twenty-fourth international joint conference on artificial intelligence*.

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

[46] Si Zhang, Hanghang Tong, Ross Maciejewski, and Tina Eliassi-Rad. 2019. Multi-level network alignment. In *The World Wide Web Conference*. 2344–2354.

[47] Vincent W Zheng, Mo Sha, Yuchen Li, Hongxia Yang, Yuan Fang, Zhenjie Zhang, Kian-Lee Tan, and Kevin Chen-Chuan Chang. 2018. Heterogeneous embedding propagation for large-scale e-commerce user alignment. In *2018 IEEE International Conference on Data Mining (ICDM)*. IEEE, 1434–1439.