Unsupervised Graph Alignment with Wasserstein Distance Discriminator

Ji Gao  
jg6yd@virginia.edu  
University of Virginia

Xiao Huang  
xhuang.polyu@gmail.com  
Hong Kong Polytechnic University

Jundong Li  
jundong@virginia.edu  
University of Virginia

ABSTRACT

Graph alignment aims to identify node correspondence across multiple graphs, with significant implications in various domains. As supervision information is often not available, unsupervised methods have attracted a surge of research interest recently. Most of existing unsupervised methods assume that corresponding nodes should have similar local structure, which, however, often does not hold. Meanwhile, rich node attributes are often available and have shown to be effective in alleviating the above local topology inconsistency issue. Motivated by the success of graph convolution networks (GCNs) in fusing network and node attributes for various learning tasks, we aim to tackle the graph alignment problem on the basis of GCNs. However, directly grafting GCNs to graph alignment is often infeasible due to multi-faceted challenges. To bridge the gap, we propose a novel unsupervised graph alignment framework WAlign. We first develop a lightweight GCN architecture to capture both local and global graph patterns and their inherent correlations with node attributes. Then we prove that in the embedding space, obtaining optimal alignment results is equivalent to minimizing the Wasserstein distance between embeddings of nodes from different graphs. Towards this, we propose a novel Wasserstein distance discriminator to identify candidate node correspondence pairs for updating node embeddings. The whole process acts like a two-player game, and in the end, we obtain discriminative embeddings that are suitable for the alignment task. Extensive experiments on both synthetic and real-world datasets validate the effectiveness and efficiency of the proposed framework WAlign.

KEYWORDS

Graph Alignment, Graph Neural Networks, Wasserstein Distance

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

ACM Reference Format:

1 INTRODUCTION

Graph alignment aims to identify the node correspondence across multiple graphs and is essential to reveal insightful graph patterns that are otherwise inaccessible with a single graph. With roots in graph theory, the graph alignment problem has significant implications in a myriad of high-impact domains such as social network analysis [44], bioinformatics [34], database [27], semantic web [10], chemistry [36], and computer vision [6]. For example, in bioinformatics, performing graph alignment to find the best mapping between different biological networks could help identify conserved functional components [33] and predicting high-quality orthology [30] across different species. Another example is that users often join multiple online social networks to enjoy more services, thus finding user identity correspondence across different social media networks with graph alignment will greatly facilitate user profiling [11] and cross-domain recommendation [23].

A vast majority of existing works formulate the graph alignment problem as a supervised learning task with the help of anchor links [5, 26, 44, 45], i.e., the observed mapping relations between two corresponding nodes. In particular, they train a classifier based on the node structural features and the observed anchor links, and the classifier is later used to predict unknown anchor links across graphs. Despite their empirical success, the annotation of anchor links is both time and labor expensive. Thus, many efforts have been devoted to investigating unsupervised graph alignment. Existing studies of unsupervised graph alignment [2, 16, 35, 43] are fundamentally based on the assumption that corresponding nodes should have similar local topology structure in different graphs, which is too strict to hold in practice [41, 43]. For example, the degrees of nodes may vary rapidly in different graphs, and a user may have a large number of friends on Facebook but could be much less active on Twitter.

Meanwhile, in many real-world graphs, nodes are often associated with wealthy side information (a.k.a. node attributes). For example, we often have rich profile information of users in social networks and gene ontology annotation information for proteins in biological networks. These attribute information are highly correlated with the graph structure and could be leveraged to address the above local structure inconsistency issue [16, 43]. In addition, recent advances in graph convolution networks (GCNs) [14, 18, 37] enable us to model complex interactions between the graph structure and node attributes for node representation learning, which have shown to be effective in various graph mining tasks, such as node classification [14], link prediction [19], and anomaly detection [9]. Hence, it motivates us to investigate if the success can be shifted to unsupervised graph alignment.

Here, one straightforward solution is to learn node embeddings with GCNs first and then find node correspondence with the learned
embedding representations. However, this straightforward solution often yields suboptimal performance, mainly because of the following multi-faceted challenges: (1) If we directly apply GCN to obtain node embeddings, the embeddings of nodes of different graphs may not be in the same feature space, which is not suitable for the alignment task. Additionally, as GCNs often suffer from the over-smoothing problem [24], thus they cannot capture the global graph patterns effectively, which may also be useful for the alignment task. (2) Suppose we have embeddings of nodes from two different graphs, then the alignment problem reduces to the bipartite matching problem based on the similarity matrix of node embeddings. However, the time complexity of most existing search algorithms (e.g., Kuhn–Munkres algorithm [28]) that find exact matches is $O(n^3)$, where $n$ is the number of nodes in graphs, prohibiting its usage in practice. (3) The node representation learning and the graph alignment are not two standalone tasks but should complement each other. Better node embeddings can help achieve better alignment results while better alignment also provides supervision signals for embedding learning. Thus, it is essential to model these two phases in a unified framework.

To tackle these challenges, in this paper, we propose a novel graph alignment framework with Wasserstein distance discriminator - WAlign. We aim to answer two research questions. 1) How to learn effective embeddings that are suitable for the alignment task? 2) How to model embedding learning and graph alignment jointly? The contributions of this study could be summarized as follows.

- We propose a novel lightweight GCN architecture that can capture both local and global graph patterns and its inherent correlations with node attributes.
- We prove that identifying the alignment of two graphs based on embedding representations is equivalent to minimizing the Wasserstein distance between embeddings of nodes in these two graphs.
- We propose a novel unsupervised graph alignment framework - WAlign, which learns discriminative node representations across different graphs for the alignment task. An efficient algorithm is designed to simultaneously optimize the embedding and graph alignment.
- We perform experiments on synthetic and real-world datasets and validate the effectiveness and efficiency of the proposed framework WAlign.

2 PRELIMINARY

In this paper, we denote an input graph as $G = (\mathcal{V}, \mathcal{E}, X)$, where $\mathcal{V} = \{v_1, v_2, ..., v_n\}$ denotes the set of nodes, $\mathcal{E}$ denotes the set of edges, and $X \in \mathbb{R}^{n \times d}$ denotes the node attribute matrix. We denote the adjacency matrix of the graph as $A \in \mathbb{R}^{n \times n}$. $A_{i,j}$ equals to 1 if the edge $(v_i, v_j) \in \mathcal{E}$, otherwise 0.

The purpose of graph alignment is to find the node correspondence of two graphs $G_i$ and $G_j$, where the number of nodes and node attribute categories in these two graphs are $(n_i, d_i)$ and $(n_j, d_j)$, respectively. Now, we formally define the unsupervised graph alignment problem as follows.

**Definition 1 (Unsupervised Graph Alignment).** Given two input graphs $G_i = (\mathcal{V}_i, \mathcal{E}_i, X_i)$ and $G_j = (\mathcal{U}_j, \mathcal{E}_j, X_j)$, an unsupervised graph alignment algorithm returns a set of corresponding node pairs $\mathcal{M} = \{(v_i, u_j) \mid v_i \in \mathcal{V}_i \land u_j \in \mathcal{U}_j\}$, where $v_i$ and $u_j$ denote two nodes of the same identity from two different graphs.

Furthermore, each node should appear at most once in an legit alignment. Formally, for a legit $\mathcal{M}$, there exists a partial mapping $\pi' : \mathcal{U}_j \rightarrow \mathcal{U}_i$ and a partial mapping $\pi : \mathcal{V}_i \rightarrow \mathcal{V}_j$, such that for all $(v_i, u_j) \in \mathcal{M}$, we have $\pi(v_i) = u_j$ and $\pi'(u_j) = v_i$. We then denote $\mathcal{M}$ as the set of all legit alignments.

Note that any two nodes in the graphs share some level of similarities. Without loss of generality, we assume the size of the aligned node pair set is often given to the algorithm, i.e. $|\mathcal{M}| = m$.

2.1 Graph Convolutional Neural Network

Graph convolutional networks (GCNs) have demonstrated their superior learning performance in various graph learning tasks [18]. The core idea is to learn the representation of each node by aggregating the information from its neighborhood. Normally, multiple graph convolutional layers are used in GCNs. Specifically, the $l$-th graph convolution layer takes an embedding matrix $H^{(l-1)} \in \mathbb{R}^{n \times d_l}$ as the input and returns a new embedding matrix $H^{(l)} \in \mathbb{R}^{n \times d_l}$, where the embedding dimension of the input and the output are $d_{l-1}$ and $d_l$, respectively. The forward propagation rule of the graph convolutional layer is defined as follows:

$$H^{(l)} = \sigma(\hat{A}H^{(l-1)}W^{(l)}),$$

where $W^{(l)} \in \mathbb{R}^{d_l \times d_{l-1}}$ denotes a weight matrix, $\sigma$ is a non-linear activation function, and $\hat{A} = \hat{D}^{-\frac{1}{2}}\hat{A}\hat{D}^{-\frac{1}{2}}$ is a degree normalized adjacency matrix. $\hat{A} = A + 1$ is the adjacency matrix after adding a self-loop for each node. $\hat{D}$ is a diagonal degree matrix from $\hat{A}$. In practice, we often set $H^{(0)} = X$, and a two-layer GCN framework is formulated as follows:

$$Z = \hat{A}\sigma(\hat{A}XW^{(1)})W^{(2)},$$

where $Z$ is an embedding matrix for all nodes, which often serves for various graph mining tasks, such as node classification, link prediction, and anomaly detection.

2.2 Problem Statement

Given node embedding representations, the unsupervised graph alignment problem can be converted to a classic task in graph theory, i.e., bipartite graph matching. In this paper, we use the sum of $l_2$ norms between embeddings to measure the alignment.

**Definition 2 (Network Embedding based Graph Alignment).** Given a node $v_i \in \mathcal{G}_i$ and a node $u_j \in \mathcal{G}_j$, let $z_{v_i}$ and $z_{u_j}$ denote the embedding representations of these two nodes respectively. The graph alignment problem is to find a set of node pairs $\mathcal{M}$ with size $m$ that minimize

$$\mathcal{M} = \arg\min_{|\mathcal{M}|=m, \mathcal{M} \in \mathbb{M}} \sum_{(v_i, u_j) \in \mathcal{M}} \|z_{v_i} - z_{u_j}\|.$$

If we let $c_{i,j} = \|z_{v_i} - z_{u_j}\|$, then the above problem is equivalent to the Minimum Cost Bipartite Matching Problem with the node set $\mathcal{V}_i \cup \mathcal{U}_i$ and the cost set $\{c_{i,j} \mid v_i \in \mathcal{V}_i, u_j \in \mathcal{U}_i\}$. 

To model graph topology and node attributes synergistically for unsupervised graph alignment, a straightforward solution is to obtain node embeddings with prevalent GCNs and then perform bipartite graph matching based on the node similarity matrix in the embedding space. However, this solution has several limitations as follows: (1) the embeddings of nodes from different graphs are in different feature spaces and the global graph patterns cannot be well captured; (2) the computational cost of traditional bipartite graph matching is high; (3) the node embeddings may not be suitable for the alignment task.

To tackle these issues, we propose a novel supervised graph alignment framework - WAlign. Fig. 1 illustrates two major components of the proposed framework. First, WAlign generates node embedding representations \( Z_s \) and \( Z_t \) for \( G_s \) and \( G_t \), respectively. The learning is parameterized by a novel lightweight GCN architecture detailed later. Second, we reveal the inherent connection between graph alignment and Wasserstein distance minimization. Thus, based on the learned representations, we generate pseudo correspondence pairs with a well-designed Wasserstein distance discriminator. Then, using the adversarial training method, we update the parameters of embedding and the discriminator in a unified optimization framework.

### 3.1 Node Representation Learning

To model graph topology and node attributes synergistically for the alignment task, we propose to learn node embeddings with prevalent GCNs architectures. However, conventional GCNs are only able to capture the local graph topology information. The reason is that a \( K \)-layer GCN only uses information from \( K \)-order neighborhood (i.e., nodes that are at most \( K \) hops away from the current node). As \( K \) increases in GCNs, the representations of all nodes will converge to similar values, which is known as the over-smoothing issue [24]. Meanwhile, recent studies have shown that global topology patterns are also essential to guarantee accurate alignment, especially when the local structure consistency assumption [43] does not hold. Thus, it motivates us to develop a new GCN architecture that can also capture global topology patterns. Fairly recently, Wu et al. [39] show that nonlinearity between GCN layers is not critical and can be simply removed without jeopardizing learning performance. Motivated by this, we propose a lightweight GCN architecture - LGCN for our problem. Specifically, we remove nonlinear functions and concatenate the embeddings of each layer:

\[
Z = f_e(A, X) = \text{Concat}([X, \hat{A}X, \hat{A}^2X, \ldots, \hat{A}^KX])W_e. \tag{4}
\]

where \( Z \) represents the output embedding matrix and \( W_e \) represents the weight matrix in \( f_e \). Because LGCN avoids layer stacking, we can specify \( K \) as a large number to capture global topology patterns. As the operation \( \text{Concat}([X, \hat{A}X, \hat{A}^2X, \ldots, \hat{A}^KX]) \) can be precomputed, it will greatly reduce the computational cost of the adversarial training phase that will be introduced later. Meanwhile, the flat structure of LGCN makes the model much easier to train and will converge faster. Additionally, a small number of model parameters will make the later adversarial training process more stable.

Considering that the properties of different graphs (both graph topology and node attributes) could be quite different, we introduce an additional transformation matrix \( T \) to accommodate the inherent differences between different graphs. Therefore, we have:

\[
Z_s = T f_e(A_s, X_s) \quad \text{and} \quad Z_t = f_e(A_t, X_t). \tag{5}
\]

### 3.2 Wasserstein Distance Discriminator

Suppose we now have successfully mapped both graphs into the same embedding space. How can we find node correspondence based on node embeddings? How to leverage the generated node

---

**Figure 1**: Illustration of the proposed WAlign framework that consists of two phases: node embedding learning and Wasserstein distance discriminator. In the figure, solid black arrows denote the forward process, while dashed blue arrows denote the backpropagation for model optimization.
We prove that when \( W(V, M) \) determines the Wasserstein Distance between the embeddings of nodes in these two graphs.

Wasserstein Distance [38] is also called Earth-Mover (EM) Distance, and is a widely used metric to measure the difference between two probability distributions.

**Definition 3 (Wasserstein Distance).** Let \( x \) and \( y \) be two random variables that are subject to two distribution \( P_x \) and \( P_y \) respectively, \( x \sim P_x \) and \( y \sim P_y \). Let \( \Pi(P_x, P_y) \) denote the set of all joint distributions for \( x \) and \( y \). Then the Wasserstein-1 Distance between \( P_x \) and \( P_y \) is defined as:

\[
W_1(P_x, P_y) = \inf_{\gamma \in \Pi(P_x, P_y)} \mathbb{E}_{(x,y) \sim \gamma}[\|x - y\|],
\]

where \( \gamma \) indicates a distribution that minimizes \( \mathbb{E}_{(x,y) \sim \gamma}[\|x - y\|] \).

Here, \( y \) indicates an optimal “mass” transportation, which specifies for every \( x \), how many “mass” should be transferred to \( y \) such that \( P_y \) will become \( P_x \).

We start from a simple case that \( m = |V'_s| = |U'_t| \). We assume a perfect alignment from \( V'_s \) to \( U'_t \) exists. It is defined as an alignment that every node in \( V'_s \) is matched to another node in \( U'_t \) according to the optimal correspondence set \( M \) in Definition 2.

Then the problem reduces to identify the optimal set \( M_{opt} \) that minimizes the summation of distance for each pair of \( (u_i, u_j) \) in \( M \). The solution is equivalent to the Wasserstein-1 Distance between the embedding distributions of \( V'_s \) and \( U'_t \).

**Proposition 4.** Suppose \( m = |V'_s| = |U'_t| \). Let \( P_x \) and \( P_y \) be probability distributions that are uniformly distributed on the embeddings of \( V'_s \) and \( U'_t \) respectively. Then we have:

\[
W_1(P_x, P_y) = \frac{1}{m} \min_{M \in \text{Embed}} \sum_{(u_i, u_j) \in M} \|z_{u_i} - z_{u_j}\|.
\]

**Proof.**

We first show an alignment \( M \) uniquely determines a transportation \( \gamma \) from \( P_x \) to \( P_y \), as it can be represented in the form of a joint distribution in Eq. 8:

\[
\mathcal{P}(x = u_i, y = u_j) = \begin{cases} \frac{1}{m} & (u_i, u_j) \in M \\ 0 & (u_i, u_j) \notin M \end{cases}.
\]

Since \( W_1(P_x, P_y) \) defines the minimal transportation distance, the transportation distance of Eq. 8 is at least \( W_1(P', P'_t) \). We have

\[
W_1(P_x, P_y) \leq \frac{1}{m} \min_{M \in \text{Embed}} \sum_{(u_i, u_j) \in M} \|z_{u_i} - z_{u_j}\|.
\]

Then we show \( W_1(P_x, P_y) \geq \frac{1}{m} \min_{M \in \text{Embed}} \sum_{(u_i, u_j) \in M} \|z_{u_i} - z_{u_j}\| \). We prove that when \( W_1(P_x, P_y) \) achieves its optimal, all \( \mathcal{P}(x = u_i, y = u_j) \) in the optimal transport \( \gamma^{*} \) must be either \( \frac{1}{m} \) or 0 so that it can always generate an alignment \( M \). In this case, \( \gamma^{*} \) determines an alignment \( M \).

To prove it, we transform the problem into a minimum cost network flow problem \( G' \) in a directed graph \( G' = (V', E') \), with flow capacity function \( U : E' \rightarrow \mathbb{R} \) and cost \( C : E' \rightarrow \mathbb{R} \). The construction of the flow network is defined through the following equations:

\[
V' = V'_s \cup U'_t \cup \{s, t\},
\]

\[
E' = \{(u_i, u_j)| u_i \in V'_s \wedge u_j \in U'_t\} \cup \{(s, u)| u_i \in V'_s \} \cup \{(u, t)| u_j \in U'_t\}.
\]

\[
U(v, u) = 1.
\]

\[
C(u, v) = \begin{cases} \|z_{u} - z_{v}\| & v \in V'_s \text{ and } u \in U'_t \\ 0 & \text{otherwise} \end{cases}
\]

Clearly, a flow with flow amount \( m \) runs in \( G' \) with capacity \( U \). If we multiple the flow amount of every pair of nodes between \( V'_s \) and \( U'_t \) with \( \frac{1}{m} \), the flow amount uniquely determines a transport between \( P_x \) and \( P_y \).}

According to the totally unimodularity of the constraint matrix [29, 32], it has an optimal solution with integer flow amount on each edge. In this optimal flow, exactly \( m \) nodes in \( V'_s \) and exactly \( m \) nodes in \( U'_t \) will be connected by the flow. Let \( V'_s' \) and \( U'_t' \) be the set of nodes selected by the flow that is in \( V'_s \) and \( U'_t \) respectively, then the Wasserstein Distance (defined by the optimal transport plan) between \( V'_s' \) and \( U'_t' \) is equivalent to \( \frac{1}{m} \min_{M \in \text{Embed}} \sum_{(u_i, u_j) \in M} \|z_{u_i} - z_{u_j}\| \).
the minimum cost network flow problem. Such assumption clearly yields a contradiction. Therefore, for any \( V^\prime_S \) and \( U^\prime_T \) that satisfies |\( V^\prime_S \)| = m and |\( U^\prime_T \)| = m, we have \( W_1(\mathcal{P}^\prime_S, \mathcal{P}^\prime_T) \geq \frac{1}{m} \min_{M \in \mathbb{R}} \sum_{(u, v) \in M} \|z_u - z_v\|\), we have Proposition 5 proved.

Proposition 5 shows that the node pairs that minimize the Wasserstein distance is the optimal node correspondence to be found. Therefore, Wasserstein distance reflects the closeness between the matched nodes. Then the nodes in \( \mathcal{V}_S \) that maximize \( f_\mathcal{V}_S \) over the \( \mathcal{V}_T \) node pairs that have the minimum Wasserstein distance. According to Eq. (13), given a fixed discriminator \( f_\mathcal{V}_T \), the Wasserstein distance discriminator returns the graph attributes as output. Then we minimize the following loss function:

\[
L_{\text{recon}} = \sum_n ||f_\mathcal{V}_S(z_n) - x_n|| + \sum_{u_i \in \mathcal{U}_i} ||f_\mathcal{U}_T(z_{u_i}) - x_{u_i}||. \tag{16}
\]

Finally, we use Eq. (17) to update the weight matrices in the embedding network, where \( \beta \) is a hyperparameter that balances two loss functions:

\[
L = \beta L_{\text{recon}} + (1 - \beta)L_{\text{wce}}. \tag{17}
\]

As a summary, in each iteration of the training process, our framework first generates node embedding with the embedding network LGCN. After that, our framework updates the parameters of the Wasserstein discriminator by minimizing \( L_{\text{wce}} \) in Eq. (15). Our framework then uses the Wasserstein discriminator to generate pseudo node correspondence pairs following Algorithm 2. Finally at the end of the iteration, the parameters of the embedding network is updated by minimizing Eq. (17). We summarize the detailed training process of our framework in Algorithm 1.

4 EXPERIMENTAL EVALUATIONS

In this section, we aim to assess the effectiveness and efficiency of the proposed framework WAlign.

4.1 Experimental Settings

Before presenting the detailed experimental results, we first show the used datasets, baseline methods, and evaluation metrics. Datasets. We first assess the performance of the proposed WAlign on two semi-synthetic datasets. Then we compare WAlign with different baselines on a real-world graph alignment task, namely, social network alignment. The detailed statistics of these datasets are shown in Table 1.

1. Synthetic Datasets. We create synthetic datasets using node permutation from multiple graphs, i.e., Protein-Protein Interaction (PPI) [47], Arenas Email [21]. For these two datasets, we follow the experimental setup in [16], that for a real-world graph \( G_r = (\mathcal{V}_r, \mathcal{E}_r, X_r) \) with an adjacency matrix \( A_r \), we generate a new graph \( G_t = (\mathcal{U}_t, \mathcal{E}_t, X_t) \) using random permutation. Specifically, we have \( A_t = P^\top A_r P \) and \( X_t = P^\top X_r \), where \( P \) is a random permutation matrix.
Algorithm 1: Unsupervised graph alignment framework WAlign

**Input**: Graphs $G_1(V_1, E_1, X_1)$ and $G_2(V_2, E_2, X_2)$, number of epochs iter, number of epochs for discriminator $\text{iter}_w$, number of epochs for reconstruction network $\text{iter}_r$, number of node correspondence pairs $m$

**Output**: Set of node correspondence pairs $M$

**def** WAlign($G_1(V_1, E_1, X_1), G_2(V_2, E_2, X_2), \text{iter}, \text{iter}_w, \text{iter}_r, m$)

- Initialize embedding function $f_e$ and discriminator $f_w$;
- for $i = 1, \ldots, \text{iter}_r$ do
  - $Z_r \leftarrow T_f(A_r, X_r)$;
  - for $j = 1, \ldots, \text{iter}_w$ do
    - $V'_w, U'_w \leftarrow \text{Pseudo}(f_w, Z_r, Z_t, m) ;$ → Algorithm 2
    - Calculate $L_{uw}$ based on Eq. (15);
    - Update($f_w, V'_w, U'_w$);
    - $V'_t, U'_t \leftarrow \text{Pseudo}(f_w, Z_t, Z_t, m) ;$ → Algorithm 2
    - Calculate $L_{uw}$ based on Eq. (14);
  - Calculate $L_{recon}$ based on Eq. (16);
  - Update($f_e, V, f_w, V'_t, U'_t$);
  - $Z_t \leftarrow T_f(A_t, X_t)$;
  - $Z_r \leftarrow f_e(A_r, X_r)$;
  - Generate node correspondence pairs $M$ from the embeddings;
- return $M$;

Algorithm 2: Algorithm of pseudo correspondence pairs prediction

**Input**: $W$. Discriminator Network $f_w$, embedding matrices $Z_r$ and $Z_t$, size of predicted anchor links $l$

**Output**: Predicted pseudo correspondence pairs $V'_t, U'_t$

**def** Pseudo($f_w, Z_r, Z_t, l$)

- $y_s \leftarrow f_w(Z_r)$;
- $y_t \leftarrow f_w(Z_t)$;
- $V'_s \leftarrow \text{arg kmin}(y_s, l)$; → Get Top-$l$ min $q_t$ with $f_w(z_t)$ value.
- $U'_s \leftarrow \text{arg kmax}(y_s, l)$; → Get Top-$l$ max $u_j$ with $f_w(z_j)$ value.

In the synthetic data and social network alignment experiments, we compare WAlign with the following baselines: (1) *K-Nearest Neighbors* — It is a simple baseline alignment method that is based on the $K$-nearest neighbors from the attribute matrix $X_s$ and $X_t$. (2) *IsoRank* [35] — It is a classic unsupervised graph alignment algorithm that uses bipartite graph matching. (3) *FINAL* [43] — It performs unsupervised graph alignment by solving an optimization problem. (4) *REGAL* [16] — It is a fast alignment method based on structural graph embeddings. Meanwhile, we also compare WAlign with the following deep learning based methods: (5) *GCN+Pseudo* — In each iteration, it first uses GCN to generate node embeddings, then synthesizes “pseudo node correspondence pairs” from embeddings based on greedy search, and updates GCN with the synthesized pairs. (6) *GAT+Pseudo* — It is similar to the previous method, but uses Graph Attention Network [37] instead of GCN for embedding learning. (7) *LGCN+Pseudo* — Baselines (6) and (7) are similar, but (7) uses our proposed LGCN for embedding learning. (8) *WAlign w/o T* — It is a variant of WAlign without using the transformation matrix $T$ for the LGCN. Different from the above deep learning based methods, our WAlign generates embeddings with the lightweight GCN architecture LGCN and uses Wasserstein distance discriminator to generate “pseudo node correspondence pairs”.

**Evaluation Metrics**. We evaluate the performance of WAlign and different baselines with Hit@k, which calculates the percentage of the nodes in $V_t$ whose ground truth alignment results in $U_t$ exist in the top-$k$ candidates returned by the model. Additionally, we compare the efficiency of different deep learning based solutions by comparing their running time.

**Implementations**. In the experiments, we use an 8-layer ($K = 8$) LGCN with 512 hidden units each layer ($h = 512$) to generate the node embeddings and match them with the Wasserstein Discriminator. Due to page limit, we put further implementation details, hyperparameter settings of WAlign, and details of baselines into the appendix.

### 4.2 Effectiveness Analysis

We report the Hit@1 and Hit@5 scores of WAlign and baseline methods in Table 2. It should be noted that as IsoRank and FINAL adopt greedy matching and eliminate a corresponding pair after they are matched, thus we can only evaluate them with Hit@1. We make the following observations from the table: (1) Our proposed WAlign achieves better alignment performance than baseline methods in most scenarios. The improvement is more clear on the real-world dataset Douban as the task is much more difficult compared with that using synthetic datasets. (2) In general, LGCN+Pseudo achieves better performance than GCN+Psuedo and GAT+Pseudo, which implies that the superiority of the proposed lightweight GCN.

<table>
<thead>
<tr>
<th>Data</th>
<th># Nodes</th>
<th># Edges</th>
</tr>
</thead>
<tbody>
<tr>
<td>Synthetic</td>
<td></td>
<td></td>
</tr>
<tr>
<td>PPI [47]</td>
<td>1,767</td>
<td>32,318</td>
</tr>
<tr>
<td>Arenas Email [21]</td>
<td>1,133</td>
<td>5,451</td>
</tr>
</tbody>
</table>

| Real-world |         |         |
| Real | -Online | 3,906  |
| -Offline | 1,118   | 3,022   |

### Table 1: Dataset statistics.
architecture LGCN over conventional GCNs for the alignment task. (3) Our proposed WAlign also outperforms the baseline method LGCN+Pseudo, which implies that the proposed Wasserstein distance discriminator indeed can generate more accurate "pseudo node correspondence pairs" compared with the greedy strategy used in other baselines. (4) For the Douban dataset, the alignment performance of all methods improve when a certain ratio of prior alignment results are incorporated. Our proposed WAlign achieves the best alignment performance on both cases, i.e., with and without prior information. (5) The PPI dataset has a 50-dimensional sparse node feature matrix. REGAL cannot well handle sparse node attributes, thus its performance is much worse than other methods. (6) In the Arena and PPI datasets, the performance of WAlign is not sensitive to the usage of transformation matrix $T$. It is because these datasets are synthesized from the same graph, which makes both graphs share the similar structure and attributes. However, in the Douban dataset when the two graphs become different, adding the transformation matrix $T$ makes the result significantly better.

4.3 Efficiency Analysis

Now, we also compare the efficiency of different deep learning based graph alignment methods. We compare the efficiency of deep learning based alignment methods in Fig. 2. The number of training epochs for different methods are fixed to ensure a fair comparison. From the figure, we have three major observations: (1) WAlign is much faster than all other deep learning based methods. For example, WAlign gets a 20x speedup on PPI and gets a 2x speedup on Arenas Email, compared to GCN+Pseudo. It is because LGCN reduces the number of layers in the GCN architecture and has a much smaller number of parameters. The flatten structure of LGCN also saves time in back-propagation. In the "pseudo node correspondence pairs" generation step, our proposed Wasserstein distance discriminator also greatly reduces the computation cost. (2) The methods that use the lightweight GCN architecture LGCN is much faster than the methods that rely on vanilla GCN and GAT, demonstrating that LGCN indeed can reduce the computational cost. (3) PPI is the densest graph among these three while Douban contains the largest number of nodes. LGCN scales well on graphs that are dense, while Wasserstein training scales well with graphs that have a lot of nodes. WAlign greatly improves the efficiency of the training process of graph alignment by combining these two techniques together.

4.4 Sensitivity Analysis

Here we study the effects of several important parameters. Due to space limit, we only perform the studies on the Douban dataset.

Number of Layers of LGCN ($K$). Firstly, we study how the number of layers in LGCN affect the performance and efficiency of WAlign. We vary the number $K \in \{1, 2 \ldots 8\}$ and report the result in the left subfigure of Fig. 3. As we can see, the alignment performance improves when $K$ becomes larger, but also demands more training time with larger $K$.

Hidden Layer Size of LGCN ($h$). We study the effects of hidden layer size $h$ in LGCN. We vary $h \in \{128, 256, 512, 1,024, 2,048\}$ and report the result in the middle subfigure of Fig. 3. The algorithm achieves the best performance when $h$ is around 512 or 1,024. In fact, WAlign achieves a fair good performance and efficiency trade-off when $h = 128$.

Hyperparameter of Prior Information ($\alpha$). Douban dataset includes a prior alignment term that is controlled by $\alpha$. Unlike baseline methods [25, 43], we do not train with the prior information in WAlign, but instead interpolate with it after the node embeddings are learned. We report the result of varying $\alpha$ in the right subfigure of Fig. 3. As $\alpha$ increases, the alignment performance first increases and then decreases. It is possible that we can get a better alignment performance by integrating the prior information into the training process. However, since we mainly focus on the case with minimal supervision, we leave it as a future direction.

5 RELATED WORK

Graph alignment, which aims to identify the node correspondence across graphs, is one of the most fundamental tasks in graph mining. In the past decade, graph alignment has attracted much research attention. Existing graph alignment methods can be mainly divided as: (1) alignment with plain graphs; and (2) alignment with attributed graphs. Here, we further divide the second category as supervised methods and unsupervised methods.

Alignment on Plain Graphs. Early graph alignment methods find node correspondence mainly based on the graph topology information. For example, IsoRank [35] calculates a pairwise topology similarity matrix in a Kronecker product graph and then solves the problem with bipartite graph matching. Later on, it is extended in IsoRankN [25] for aligning multiple graphs. NetAlign [2] formulates graph alignment as an integer quadratic programming problem and solves the problem in a distributed manner. UniAlign [20] proposes a gradient descent approach to solve the graph alignment problem efficiently. UMA [42] first investigates the alignment problem in multiple anonymized social networks and considers the constraints of transitivity law. However, these methods are mainly based on the local topology consistency assumption [43], which often does not hold in real-world scenarios.

Supervised Alignment on Attributed Graphs. Lately, many supervised graph alignment algorithms have been proposed for attributed graphs. These algorithms aim to train a classifier to fit observed anchor links, which is later used to predict unknown anchor links for the alignment task. For example, IONE [26] projects nodes into an embedding space and minimizes the probability of the provided anchor links together with the KL divergence between the source distribution and target distribution. COSNET [44] designs an energy-based model to capture both local and global consistency for the supervised alignment task. Deeplink [45] proposes to use random walks to sample graph and learns node embeddings for supervised graph alignment with a deep neural network architecture. CrossMNA [5] learns two types of node embeddings with the help of anchor links, one for the alignment task and the other one for downstream network analysis tasks. DGMC [13] proposes a graph convolutional network based two step matching algorithm that defines a loss function based on neighbor resemblance. SNPNA [22] uses Wasserstein distance in social network alignment. However, it does not utilize the graph information explicitly. Gromov-Wasserstein Learning (GWL) [40] minimizes a Wasserstein Distance based loss function towards a graph corresponding matrix, whose optimal
Table 2: Unsupervised graph alignment performance comparison.

<table>
<thead>
<tr>
<th></th>
<th>PPI</th>
<th>Arena Email</th>
<th>Douban (w/ prior)</th>
<th>Douban (w/o prior)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Hit@1</td>
<td>Hit@5</td>
<td>Hit@1</td>
<td>Hit@5</td>
</tr>
<tr>
<td>KNN</td>
<td>17.71%</td>
<td>33.16%</td>
<td>0.18%</td>
<td>0.88%</td>
</tr>
<tr>
<td>IsoRank</td>
<td>73.85%</td>
<td>\</td>
<td>72.33%</td>
<td>\</td>
</tr>
<tr>
<td>FINAL</td>
<td>76.80%</td>
<td>50.90%</td>
<td>54.35%</td>
<td>34.35%</td>
</tr>
<tr>
<td>REGAL</td>
<td>29.30%</td>
<td>50.90%</td>
<td>94.00%</td>
<td>98.14%</td>
</tr>
<tr>
<td>GCN+Pseudo</td>
<td>85.80%</td>
<td>91.91%</td>
<td>78.17%</td>
<td>90.29%</td>
</tr>
<tr>
<td>GAT+Pseudo</td>
<td>83.47%</td>
<td>90.15%</td>
<td>33.78%</td>
<td>55.60%</td>
</tr>
<tr>
<td>LGCN+Pseudo</td>
<td>86.36%</td>
<td>91.45%</td>
<td>94.94%</td>
<td>99.60%</td>
</tr>
<tr>
<td>WAlign w/o T</td>
<td>86.83%</td>
<td>91.57%</td>
<td>97.16%</td>
<td>99.58%</td>
</tr>
<tr>
<td>WAlign</td>
<td>86.76%</td>
<td>91.79%</td>
<td>96.81%</td>
<td>99.60%</td>
</tr>
</tbody>
</table>

Figure 2: Running time comparison between deep learning based methods (left: PPI; Middle: Arenas Email; Right: Douban).

Figure 3: Performance of our method with different hyperparameters (left: number of layers $K$; middle: hidden layer size $h$; right: hyperparameter $\alpha$ for prior information).

solution could be non-integer, where the optimal solution of our problem is always integer.

**Unsupervised Alignment on Attributed Graphs.** As anchor links are costly to obtain in many real-world scenarios, unsupervised algorithms have attracted a surge of research interests recently. Among them, FINAL [43] optimizes a quadratic programming problem, which maximizes the similarity between the nodes that are structurally similar or sharing the same attribute. REGAL [16] generates structural embedding vectors for each node in the graph, and then the alignment results are obtained from the embeddings. HashAlign [15] develops a hash-based framework for multiple graph alignment. One recent work UAGA [3] proposes a two-step method, it first generates an unsupervised deep learning based graph embedding and then performs graph alignment on the learned embedding. Another recent work CONE-align [4] defines the concept of neighborhood consistency and generates graph matching by maximizing the consistency. Finally, the recent work of DANA [8] directly runs Generative Adversarial Network (specifically, CycleGAN [46]) on the graph embedding matrices to generate a graph alignment.

6 CONCLUSION

In this paper, we propose a novel unsupervised graph alignment framework WAlign. To capture the inherent correlations between graph topology and node attributes for the alignment task, we develop a lightweight graph neural network architecture LGCN, which speeds up the embedding computation and fits the alignment task better. We then prove the equivalence between graph matching problem and the Wasserstein distance minimization in the embedding space. Thus, given the node embedding representations, we propose a novel Wasserstein distance discriminator that can quickly determine the candidate sets of node correspondence
pairs without involving explicit calculation that could be expensive. The obtained correspondence pairs are further leveraged to update the embeddings. The whole optimization scheme can be formulated as a two-player game and is iterated until we find discriminative embeddings that are suitable for the alignment task. Extensive experiments on synthetic and real-world datasets validate the effectiveness and efficiency of the proposed framework.

7 ACKNOWLEDGEMENTS

The authors would like to thank anonymous reviewers for their constructive feedback.

REFERENCES

8 APPENDIX - REPRODUCING DETAILS

8.1 Source Code
Our code is available at https://github.com/gaoji7777/walign.

8.2 More Details of Experiments
8.2.1 Download Links of the Dataset.
- PPI dataset: We use the dataset from Pytorch-geometric [12], which is originally released by [21]. https://github.com/rusty1s/pytorch_geometric.
- Arenas Email: We use the dataset released by [16]. http://snap.stanford.edu/node2vec/
- Douban: We use the dataset from [43]. https://github.com/maffia92/FINAL-network-alignment-KDD16

8.2.2 Dataset Split. As it is an unsupervised task, we did not split the dataset as training/testing/validation.

8.2.3 Hyperparameter Selection. In the experiments, we set the embedding vector size as $h = 512$. We apply ADAM Optimizer [17] to update the model parameters, with an $L_2$ penalty that has $5 \times 10^{-4}$ as the weight.

Also, we train 20 epochs on our embedding network $f_e$. During each iteration of embedding model, we train 5 epochs on the discriminator $f_w$ to make sure the discriminator will have optimal performance each time. Therefore, the discriminator is trained for 20 times, each with 5 epochs. Similarly, we train 5 epochs on the reconstruction networks each iteration.

The learning rate for Douban dataset is set as $lr = 0.001$, while for other datasets it is set as $lr = 0.00001$. The learning rate of discriminator network is $lr = 0.01$ for Douban dataset and 0.001 for other datasets.

In the experiments, we set the hyperparameter of all the deep learning methods to be the same.

8.3 Implementation of Baselines
We use validate baseline implementations for the performance comparison. Specifically,
- IsoRANK: We use the implementation of IsoRANK from the Github repository of [43].
- FINAL: We use the original implementation of FINAL from the Github repository of [43].
- REGAL: We use the original implementation of REGAL from the Github repository of [16].

Other compared deep learning baselines are included in our code.

8.4 Infrastructure
All experiments are performed on one single server machine. The server machine contains 16 AMD Opteron Processors at 3.10GHz with 256GB RAM.

We use python 3.6 for our experiments. Also, PyTorch 1.4 is used for the deep learning infrastructure. The dependencies of our code include PyTorch-Geometric 1.4.3, and all related dependencies for PyTorch and PyTorch-geometric.