CONE-Align: Consistent Network Alignment with Proximity-Preserving Node Embedding

Xiyuan Chen  
University of Michigan, Ann Arbor  
shinech@umich.edu

Mark Heimann  
University of Michigan, Ann Arbor  
mheimann@umich.edu

Fatemeh Vahedian  
University of Michigan, Ann Arbor  
vfatemeh@umich.edu

Danai Koutra  
University of Michigan, Ann Arbor  
dkoutra@umich.edu

ABSTRACT

Network alignment, the process of finding correspondences between nodes in different graphs, has many scientific and industrial applications. Existing unsupervised network alignment methods find suboptimal alignments that break up node neighborhoods, i.e., do not preserve matched neighborhood consistency. To improve this, we propose CONE-Align, which models intra-network proximity with node embeddings and matches nodes across networks by comparing the embeddings after aligning their subspaces. Experiments on diverse, challenging datasets show that CONE-Align is robust and obtains up to 49% greater accuracy than the state-of-the-art graph alignment algorithms.

ACM Reference Format:

1 INTRODUCTION

Graphs or networks are ubiquitous structures for representing complex interconnections between entities. An important problem in mining graph data is network alignment, or the task of finding correspondences between nodes in different graphs. This task has diverse, important, scientific and industrial applications, such as recommendation across social networks, pattern recognition, protein-protein interaction analysis, and database schema matching [15].

This work is inspired by a common limitation of network alignment methods. We find that many unsupervised graph alignment approaches (e.g., FINAL [25], NetAlign [3], REGAL [14]) fail to achieve matched neighborhood consistency: nodes that are close in one graph are often not matched to nodes that are close in the other graph. For example, REGAL [14] matches nodes using node embeddings capturing each node’s structural role in the network. However, neighboring nodes may not have similar structural roles, resulting in very different embeddings that may be matched far apart in the other graph, violating matched neighborhood consistency.

To solve this problem, we propose CONE-Align for consistent network alignment with proximity-preserving node embedding. We use well-known node embedding methods that learn similar embeddings for neighboring nodes in each graph. However, because nodes are not in proximity across graphs, these methods are transductive, and nodes in different graphs will be embedded into different subspaces. Therefore, we align the graphs’ embedding subspaces, and then we can match the nodes using embedding similarity. Since neighboring nodes in each graph will have similar embeddings, they will be matched to similar parts of the other graph. Thus, we have the best of both worlds with CONE-Align: matched neighborhood consistency and cross-graph comparability.

Our contributions can be summarized as follows:

- **Insights for Network Alignment**: We define the principle of matched neighborhood consistency, which motivates us to use node embedding methods with a different kind of objective than what has been used for unsupervised network alignment.

- **Principled New Method**: We propose CONE-Align for unsupervised network alignment, which makes embedding subspaces for different graphs comparable, analogous to machine translation using monolingual word embeddings.

- **Rigorous Experiments**: On challenging datasets, we show that CONE-Align outperforms strong baselines by up to 49% in accuracy, as it better preserves matched neighborhood consistency. Our code is available at https://github.com/GemsLab/CONE-Align.

2 RELATED WORK

Node Embeddings. Node embeddings are latent feature vectors modeling relationships between nodes and/or structural characteristics, learned with various shallow and deep architectures and used for many graph mining tasks [10]. Most embedding objectives model proximity within a single graph: nearby nodes (e.g. neighbors sharing an edge or nodes with mutual neighbors) have similar features. For example, DeepWalk [20] and node2vec [12] perform random walks starting at each node to sample context nodes, using a shallow neural architecture to embed nodes similarly to their context. This process implicitly factorizes a node pointwise mutual information matrix, which NetMF [21] instead directly factorizes.

In contrast, structural embedding methods capture a node’s structural role independent of its positional reference to specific nodes; this independence makes embeddings comparable across graphs [14]. For example, struc2vec [22] resembles DeepWalk and...
node2vec but performs random walks on an auxiliary structural similarity graph. xNetMF embeddings [14] capture local neighborhood connectivity. For more on the distinction between structural and proximity-preserving node embeddings, see [23].

**Network Alignment.** Classic graph alignment approaches often formulate an optimization-based assignment problem. For example, the message-passing algorithm NetAlign [3] tries to preserve “complete squares” by matching two nodes sharing an edge in one graph to counterparts sharing an edge in the other graph. FNAL [25] optimizes a topological consistency objective which may be augmented with node and edge attribute information. Our approach is initialized by the solution to a classic convex optimization formulation [9], but to improve the accuracy, we turn to a different class of methods: those that compare node embeddings.

REGAL [14] matches xNetMF structural embeddings that are comparable across networks. Subsequent work [7] models intranetwork proximity via link prediction, but its cross-network comparison is also based on structural similarity. To use transductive proximity-preserving embedding objectives, workarounds include connecting the two graphs with ground truth “seed” alignments [19] if any are known, or using adversarial training techniques in machine translation [17] as in another recent work [6].

**3 PRELIMINARIES**

**Graphs and Embeddings.** We consider two graphs $G_1$ and $G_2$ with node sets $V_1$, $V_2$, edgesets $E_1$, $E_2$ and adjacency matrices $A_1$, $A_2$ containing edges between nodes. As in [14], for simplicity, we assume that both graphs have $n$ nodes (if not, we can add singleton nodes to one graph). For each graph $G_i$, we can create an $n \times d$ matrix $Y_i$ of $d$-dimensional node embeddings.

**Alignment.** An alignment between the nodes of two graphs is a function $\pi : V_1 \rightarrow V_2$, or alternatively a matrix $P$, where $P_{ij}$ is the (real-valued or binary) similarity between node $i$ in $G_1$ and node $j$ in $G_2$. A mapping $\pi$ can be found from $P$, e.g. greedy alignment $\pi(i) = \arg \max_{j} P_{ij}$.

**Neighborhood.** Let $N_{G_i}(i)$ be the neighbors of node $i$ in $G_1$, i.e., nodes that share an edge with $i$. We define node $i$’s “mapped neighborhood” in $G_2$ as the set of nodes onto which $\pi$ maps $i$’s neighbors: $N_{G_2}(\pi(i)) = \{ j \in V_2 : \exists k \in N_{G_1}(i) \ s.t. \pi(k) = j \}$. Also, we denote the neighbors of node $i$’s counterpart $\pi(i)$ in $G_2$ as $N_{G_2}(\pi(i))$. We define the matched neighborhood consistency (MNC) of node $i$ in $G_1$ and $j$ in $G_2$ as the Jaccard similarity of the two sets (visualized for a toy example in Fig. 1):

$$\text{MNC}(i, j) = \frac{|N_{G_2}(\pi(i)) \cap N_{G_2}(j)|}{|N_{G_2}(\pi(i)) \cup N_{G_2}(j)|}$$

**Problem Statement.** Given two graphs $G_1$ and $G_2$ (with meaningful node correspondences), we seek to recover their alignment $\pi$ in an unsupervised setting with no node matchings known a priori, while ideally achieving high MNC.

**Figure 2:** Overview of CONE-Align. Given two graphs $G_1$ and $G_2$, we first use node embedding to model intra-graph node proximity. Second, we align the embedding spaces for cross-graph comparability. Third, we match each node in $G_1$ to the node in $G_2$ with the most similar embedding.

**4 METHOD**

We detail CONE-Align (Fig. 2, with pseudocode in Alg. 2), our proposed method using node embeddings to respect matched neighborhood consistency and identify cross-graph node similarities.

**4.1 Step 1: Node Embedding**

We obtain normalized node embeddings separately for each input graph. CONE-Align is a framework with which we can use many popular embedding methods, graph neural networks, etc. [10], even though they may be designed for a single network. We only need the embeddings to preserve intra-graph node proximity, i.e. neighboring nodes in each graph have similar embeddings and will be mapped close by when using embedding similarity. This preserves matched neighborhood consistency robustly: even when nodes are not neighbors due to missing edges [7], many node embedding algorithms can preserve any higher-order proximities they share.

**4.2 Step 2: Embedding Space Alignment**

Due to the invariance of the embedding objective, the two graphs’ node embeddings $Y_1 \in \mathbb{R}^{n \times d}$ and $Y_2 \in \mathbb{R}^{n \times d}$ may be translated, rotated, or rescaled relative to each other. Thus, to compare them, we must align the embedding subspaces. Inspired by unsupervised word translation [11], we jointly solve two optimization problems:

**Procrustes.** If node correspondences were known, we could find a linear embedding transformation $Q$ from the set of orthogonal matrices $O^d$. $Q$ aligns the columns of the node embedding matrices, i.e. the embedding spaces. It can be obtained by solving an orthogonal Procrustes problem:

$$\min_{Q \in O^d} ||Y_1 Q - Y_2||^2_2 \quad \text{(column permutation)}$$

Its solution is $Q^* = UV^T$, where $U \Sigma V^T$ is the SVD of $Y_1^T Y_2$ [24].

**Wasserstein.** If the embedding space transformation were known, we could solve for the optimal node correspondence $P$ from the set of permutation matrices $P^n$. $P$ aligns the rows of the node embedding matrices, i.e. the nodes. It can be obtained using the Sinkhorn algorithm [5] to minimize the squared Wasserstein distance:

$$\min_{P \in P^n} \min_{Q \in O^d} ||Y_1 Q - PY_2||^2_2 \quad \text{(row permutation)}$$

**Wasserstein Procrustes.** As we know neither the correspondences nor the transformation, we combine the problems:

$$\min_{P \in P^n} \min_{Q \in O^d} ||Y_1 Q - PY_2||^2_2$$

---

MLG ’20, KDD Workshop on Mining and Learning with Graphs, Online

Xiyuan Chen, Mark Heimann, Fatemeh Vahedian, and Danai Koutra
We equivalently solve \( \max_{\mathbf{P} \in \mathcal{P}} \max_{\mathbf{Q} \in \mathcal{O}} \text{trace}(\mathbf{Q}^\top \mathbf{Y}_1 \mathbf{P} \mathbf{Y}_2) \) with a stochastic optimization scheme [11], alternating between the Wasserstein and Procrustes problems. For \( T \) iterations, we use the current embedding transformation \( \mathbf{Q} \) to find a matching \( \mathbf{P}_t \) for minibatches \( \mathbf{Y}_1, \mathbf{Y}_2 \) of \( b \) embeddings each, using Sinkhorn [5] with regularization parameter \( \lambda \). We then use the gradient of the Wasserstein Procrustes distance \( ||\mathbf{Y}_1 \mathbf{Q} - \mathbf{P}_t \mathbf{Y}_2||_2^2 \), evaluated on the minibatches \( \mathbf{Y}_1, \mathbf{Y}_2 \), to update \( \mathbf{Q} \) with gradient descent (Alg. 1).

**Convex Initialization.** To initialize the above nonconvex procedure, we turn to a classic convex graph matching formulation [9]:

\[
\min_{\mathbf{P} \in \mathcal{P}} \left\| (\mathbf{A}_1 \mathbf{P} - \mathbf{P}_0 \mathbf{A}_2) \right\|_2^2
\]

where \( \mathcal{P}^n \) is the convex hull of \( \mathcal{P}^n \). We can find the global minimizer \( \mathbf{P}^* \) with the Frank-Wolfe algorithm [8] for \( n_0 \) iterations and Sinkhorn [5] with regularization parameter \( \lambda_0 \). Using \( \mathbf{Y}_1 \) and \( \mathbf{P}^* \mathbf{Y}_2 \), an initial \( \mathbf{Q} \) can be generated with orthogonal Procrustes (Eq. (5)).

**Complexity Considerations.** Our subspace alignment procedure (Alg. 1) uses SVD and Sinkhorn’s algorithm [5] on the full data. Although these algorithms have quadratic time complexity, recent superlinear approximations [1, 2] can further scale up CONE-Align.

**4.3 Step 3: Matching Nodes with Embeddings**

After aligning the embeddings with the final transformation matrix \( \mathbf{Q} \), we match each node in \( G_1 \) to its nearest neighbor in \( G_2 \) based on Euclidean distance. We could use scaling corrections to mitigate “hubness” whereby many nodes are mapped to the same counterpart [11], but we did not find this necessary. Following [14], we use a \( k \)-d tree for fast nearest neighbor search between \( \mathbf{Y}_1 \mathbf{Q} \) and \( \mathbf{Y}_2 \).

**5 EXPERIMENTS**

In this section, we analyze CONE-Align’s accuracy and matched neighborhood consistency in network alignment.

**Configuration of CONE-Align.** We use NetMF [21] node embeddings, which we find obtain higher accuracy than the related DeepWalk and node2vec [12, 20], possibly because the latter use random walks that increase variance [14]. We use default values [21], approximating the normalized graph Laplacian with 256 eigenpairs, and set embedding dimension \( d = 128 \), context window size \( w = 10 \), and \( \alpha = 1 \) negative samples [21]. For the subspace alignment, we use parameters which yield good accuracy and speed: \( n_0 = 10 \) iterations and regularization \( \lambda_0 = 1.0 \) for the initial convex matching, and \( T = 50 \) iterations of Wasserstein Procrustes optimization with batch size \( b = 10 \), learning rate \( \eta = 1.0 \), and regularization \( \lambda = 0.05 \).

**Data.** Following prior works [6, 14, 25], we simulate a network alignment scenario with known ground truth: a graph with adjacency matrix \( \mathbf{A} \) is aligned to a noisy permutated copy \( \mathbf{A}^* \). We generate a random permutation matrix \( \mathbf{P} \) and set \( \mathbf{A}^* = \mathbf{P} \mathbf{A}^\top \); we then randomly remove edges from \( \mathbf{A}^* \) with probability \( p \in [0.05, 0.10, 0.15, 0.20, 0.25] \). We perform this procedure on graphs representing various phenomena as shown in Table 1, all of which are on par with graph sizes considered in existing works.

**Table 1: Description of the datasets used.**

<table>
<thead>
<tr>
<th>Name</th>
<th>Nodes</th>
<th>Edges</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Arenas [16]</td>
<td>133</td>
<td>5451</td>
<td>communication network</td>
</tr>
<tr>
<td>Hamsterster</td>
<td>2426</td>
<td>16613</td>
<td>social network</td>
</tr>
<tr>
<td>PPI [4]</td>
<td>3890</td>
<td>76584</td>
<td>protein-protein interaction</td>
</tr>
<tr>
<td>Facebook [18]</td>
<td>4039</td>
<td>88234</td>
<td>social network</td>
</tr>
</tbody>
</table>

**Baselines.** Our baselines are unsupervised methods using diverse techniques (belief propagation, spectral methods, and embeddings): (1) NetAlign [3] and (2) FINAL [25], and (3) REGAL [14]. We configure each method following the literature. NetAlign and FINAL require a matrix of prior alignment information, for which we take the top \( k = \lceil \log_2 n \rceil \) most similar nodes by degree for each node [6, 14]. For REGAL we use recommended embedding dimension \( 10 \log_2 (2n) \), maximum neighbor distance \( 2 \) with discount factor \( \alpha = 0.1 \), and resolution parameter \( y_{struc} = 1 \) [14].

**5.1 Alignment Performance**

**5.1.1 Evaluation.** We measure alignment accuracy, or the proportion of correctly aligned nodes, as well as the average matched neighborhood consistency (MNC) using Eq. (1) across all nodes.

**5.1.2 Results.** In Fig. 3, we report average and standard deviation for each metric over five trials for each experimental setting. CONE-Align largely outperforms baselines. We study 5X higher noise levels than prior work [14]; in this challenging setting, NetAlign and FINAL achieve <10% accuracy. On the denser PPI and Facebook networks with low noise, REGAL is most accurate; it may be hard to model node proximities distinctly in large neighborhoods. However, CONE-Align outperforms it above 10% noise.
were thought not to be applicable [13]. We can also explore other multi-network tasks such as graph classification, where they should not neglect intra-network proximity information. With embedding subspace alignment, we obtain compatibility while capturing proximity. In the future, this may allow transductive node embeddings to improve other multi-network tasks such as graph classification, where they were thought not to be applicable [13]. We can also explore other node embeddings, particularly methods using node/edge attributes.

5.2 Matched Neighborhood Consistency

To further understand MNC, we analyze it on a node-level basis.

5.2.1 Setup & Evaluation. For brevity, we show only REGAL and CONE-Align on the Arenas dataset with 5% noise. We split the nodes into three groups by degree: \( [\Delta^*, \Delta^*], (\frac{\Delta}{2}, \Delta^*], [\frac{\Delta}{2}, \Delta^*] \), where \( \Delta^* \) is the maximum degree, and plot the distribution of MNC for both correctly and incorrectly aligned nodes.

5.2.2 Results. Fig. 4 shows that for both methods, MNC is much higher for correctly aligned nodes across degree levels, but they misalign a few lower degree nodes with high MNC, whose smaller neighborhoods may be misaligned together. (However, CONE-Align correctly aligns all high-degree nodes.)

6 CONCLUSION

CONE-Align’s success offers the following takeaway: the quest for cross-network embedding comparability should not neglect intra-network proximity information. With embedding subspace alignment, we obtain compatibility while capturing proximity. In the future, this may allow transductive node embeddings to improve other multi-network tasks such as graph classification, where they were thought not to be applicable [13]. We can also explore other node embeddings, particularly methods using node/edge attributes.

Figure 3: Average accuracy (solid lines) and MNC (dashed lines), with standard deviation in error bars, vs. different noise levels.

CONE-Align significantly outperforms baselines and better preserves MNC across datasets, particularly as noise increases.

Figure 4: MNC of CONE-Align and REGAL on the Arenas dataset with 5% noise. Compared to REGAL, CONE-Align generates significantly higher MNC for almost all nodes.

CONE-Align is more robust to noise. CONE-Align’s accuracy declines less sharply than REGAL as noise increases, and it is the only method to measurably align any datasets at 25% noise.

Accuracy and MNC are closely related. They trend similarly, and more accurate methods (esp. CONE-Align) have higher MNC.

Runtime. CONE-Align’s average runtime per dataset ranges from 5 sec to 4 min: slower than the famously scalable methods NetAlign and REGAL, but at least twice as fast as FINAL.

6 CONCLUSION

CONE-Align’s success offers the following takeaway: the quest for cross-network embedding comparability should not neglect intra-network proximity information. With embedding subspace alignment, we obtain compatibility while capturing proximity. In the future, this may allow transductive node embeddings to improve other multi-network tasks such as graph classification, where they were thought not to be applicable [13]. We can also explore other node embeddings, particularly methods using node/edge attributes.

ACKNOWLEDGEMENTS

We would like to thank the reviewers for their feedback. This work is supported by the NSF under Grant No. IIS 1845491, Army Young Investigator Award No. W911NF1810397, and Adobe, Amazon, and Google Faculty awards.

REFERENCES