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:

Xiyuan Chen, Mark Heimann, Fatemeh Vahedian, and Danai Koutra. 2020. CONE-Align: Consistent Network Alignment with Proximity-Preserving Node Embedding. In *Proceedings of (MLG '20).* ACM, New York, NY, USA, 4 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

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 **CON**sistent network alignment with proximity-preserving node **E**mbedding. 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

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

ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

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. FINAL [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 intra-network 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 nodesets $\mathcal{V}_1, \mathcal{V}_2$, edgesets $\mathcal{E}_1, \mathcal{E}_2$ and adjacency matrices $\mathbf{A}_1, \mathbf{A}_2$ 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 $\mathbf{Y}_i$ of d-dimensional node embeddings.

Alignment. An alignment between the nodes of two graphs is a function $\pi : \mathcal{V}_1 \rightarrow \mathcal{V}_2$, or alternatively a matrix $\mathbf{P}$, where $\mathbf{P}_{ij}$ is the (real-valued or binary) similarity between node i in G_1 and node j in G_2. A mapping π can be found from $\mathbf{P}$, e.g. greedy alignment $\pi(i) = \arg\max_j \mathbf{P}_{ij}$.

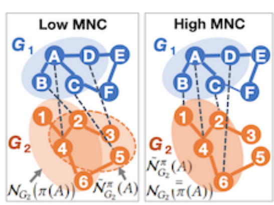


Figure 1: Partial alignments between G_1 & G_2: varying matched neighborhood consistency for node A in G_1 and its counterpart in G_2, node 4.

Neighborhood. Let $\mathcal{N}_{G_1}(i)$ 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 π maps i's neighbors: $\tilde{\mathcal{N}}^{\pi}_{G_2}(i) = \{j \in \mathcal{V}_2 : \exists k \in \mathcal{N}_{G_1}(i) \text{ s.t. } \pi(k) = j\}$. Also, we denote the neighbors of node i's counterpart $\pi(i)$ as $\mathcal{N}_{G_2}(\pi(i))$. 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):

$$MNC(i,j) = \frac{|\tilde{\mathcal{N}}^{\pi}_{G_2}(i) \cap \mathcal{N}_{G_2}(j)|}{|\tilde{\mathcal{N}}^{\pi}_{G_2}(i) \cup \mathcal{N}_{G_2}(j)|} \quad (1)$$

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

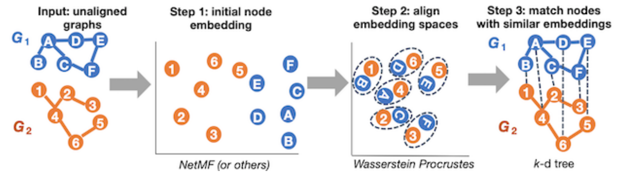


Figure 2: Overview of CONE-Align. Given two graphs G_1 and G_2, we first use node embedding to model intra-graph node proximity. Second, we align the embedding spaces for cross-graph comparability. Third, we match each node in G_1 to the node in G_2 with the most similar embedding.

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 $\mathbf{Y}_1 \in \mathbb{R}^{n \times d}$ and $\mathbf{Y}_2 \in \mathbb{R}^{n \times d}$ 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 $\mathbf{Q}$ from the set of orthogonal matrices O^d. $\mathbf{Q}$ aligns the columns of the node embedding matrices, i.e. the embedding spaces. It can be obtained by solving an orthogonal Procrustes problem:

$$\min_{\mathbf{Q} \in O^d} ||\mathbf{Y}_1\mathbf{Q} - \mathbf{Y}_2||_2^2 \quad \textit{(column permutation)} \quad (2)$$

Its solution is $\mathbf{Q}^* = \mathbf{U}\mathbf{V}^\top$, where $\mathbf{U}\Sigma\mathbf{V}^T$ is the SVD of $\mathbf{Y}_1^\top\mathbf{Y}_2$ [24].

Wasserstein. If the *embedding space transformation* were known, we could solve for the optimal node correspondence $\mathbf{P}$ from the set of permutation matrices $\mathcal{P}^n$. $\mathbf{P}$ 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:

$$\min_{\mathbf{P} \in \mathcal{P}^n} ||\mathbf{Y}_1 - \mathbf{P}\mathbf{Y}_2||_2^2 \quad \textit{(row permutation)} \quad (3)$$

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

$$\min_{\mathbf{Q} \in O^d} \min_{\mathbf{P} \in \mathcal{P}^n} ||\mathbf{Y}_1\mathbf{Q} - \mathbf{P}\mathbf{Y}_2||_2^2 \quad (4)$$

Algorithm 1 Align Embeddings $(\mathbf{Y}_1, \mathbf{Y}_2, \mathbf{A}_1, \mathbf{A}_2)$

Input: node embeddings $\mathbf{Y}_1, \mathbf{Y}_2$, adjacency matrices $\mathbf{A}_1, \mathbf{A}_2$
/* Convex Initialization */
$\mathbf{P}^* = \arg\min_{\mathbf{P}\in\mathcal{B}^n} ||(\mathbf{A}_1\mathbf{P} - \mathbf{P}\mathbf{A}_2)||_2^2$ ▷ Initial node correspondences: ▷ based on Franke-Wolfe [8] and Sinkhorn [5]
$\mathbf{U}\Sigma\mathbf{V}^\top = \text{SVD}(\mathbf{Y}_1^\top\mathbf{P}^*\mathbf{Y}_2)$
$\mathbf{Q} = \mathbf{U}\mathbf{V}^\top$ ▷ Compute initial embedding space transformation
/* Stochastic Alternating Optimization */
for $t = 1 \to T$ **do** ▷ T: # iter (e.g., 50)
  $\mathbf{P}_t = \arg\max_{\mathbf{P}\in\mathcal{P}^b} \text{trace}(\mathbf{Q}^\top\mathbf{Y}_{1_t}^\top\mathbf{P}\mathbf{Y}_{2_t})$ ▷ Via Sinkhorn, compute ▷ optimal matching for size-b (e.g., 10) minibatches $\mathbf{Y}_{1_t}, \mathbf{Y}_{2_t}$
  $\mathbf{G}_t = -2\mathbf{Y}_{1_t}^\top\mathbf{P}_t\mathbf{Y}_{2_t}$ ▷ Compute gradient of WP distance wrt Q
  $\mathbf{U}\Sigma\mathbf{V}^\top = \text{SVD}(\mathbf{Q} - \eta\mathbf{G}_t)$ ▷ Update orthogonal transform. matrix
  $\mathbf{Q} = \mathbf{U}\mathbf{V}^\top$ ▷ η: learning rate (e.g., 1.0)
return $\mathbf{Q}$ ▷ orthogonal transformation Q

Algorithm 2 CONE-Align $(\mathbf{A}_1, \mathbf{A}_2)$

Input: adjacency matrices $\mathbf{A}_1, \mathbf{A}_2$
/* STEP 1. Model Intra-Network Proximities with Embeddings */
$\mathbf{Y}_1 = \text{proximity-emb-method}(\mathbf{A}_1)$, $\mathbf{Y}_2 = \text{proximity-emb-method}(\mathbf{A}_2)$
$\mathbf{Y}_1 = \frac{\mathbf{Y}_1}{||\mathbf{Y}_1||_2}$, $\mathbf{Y}_2 = \frac{\mathbf{Y}_2}{||\mathbf{Y}_2||_2}$ ▷ Normalize the node embeddings
/* STEP 2. Align Embedding Spaces for Cross-Graph Comparability */
$\mathbf{Q} = \text{Align Embeddings}(\mathbf{Y}_1, \mathbf{Y}_2, \mathbf{A}_1, \mathbf{A}_2)$
/* STEP 3. Match Nodes with Similar Embeddings */
$\mathbf{Y}_1 = \mathbf{Y}_1\mathbf{Q}$ ▷ Align embedding spaces and greedily match nodes with
$\mathbf{P} = \text{QueryKDTree}(\mathbf{Y}_1, \mathbf{Y}_2)$ ▷ sim. embeddings via k-d tree (NN search)
return P ▷ permutation matrix with aligned nodes across input graphs

We equivalently solve $\max_{\mathbf{P}\in\mathcal{P}^n}\max_{\mathbf{Q}\in\mathcal{O}^d} \text{trace}(\mathbf{Q}^\top\mathbf{Y}_1^\top\mathbf{P}\mathbf{Y}_2)$ 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 $\mathbf{Y}_{1_t}, \mathbf{Y}_{2_t}$ of b embeddings each, using Sinkhorn [5] with regularization parameter λ. We then use the gradient of the Wasserstein Procrustes distance $||\mathbf{Y}_{1_t}\mathbf{Q} - \mathbf{P}_t\mathbf{Y}_{2_t}||_2^2$, evaluated on the minibatches $\mathbf{Y}_{1_t}, \mathbf{Y}_{2_t}$, 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]:

$$\min_{\mathbf{P}\in\mathcal{B}^n} ||(\mathbf{A}_1\mathbf{P} - \mathbf{P}\mathbf{A}_2)||_2^2 \tag{5}$$

where $\mathcal{B}^n$ 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 λ_0. Using $\mathbf{Y}_1$ and $\mathbf{P}^*\mathbf{Y}_2$, an initial $\mathbf{Q}$ can be generated with orthogonal Procrustes (Eq. (2)).

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 permuted copy $\mathbf{A}^*$. We generate a random permutation matrix $\mathbf{P}$ and set $\mathbf{A}^* = \mathbf{P}\mathbf{A}\mathbf{P}^\top$; 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.

Name	Nodes	Edges	Description
Arenas [16]	1 133	5 451	communication network
Hamsterster [16]	2 426	16 613	social network
PPI [4]	3 890	76 584	protein-protein interaction
Facebook [18]	4 039	88 234	social network

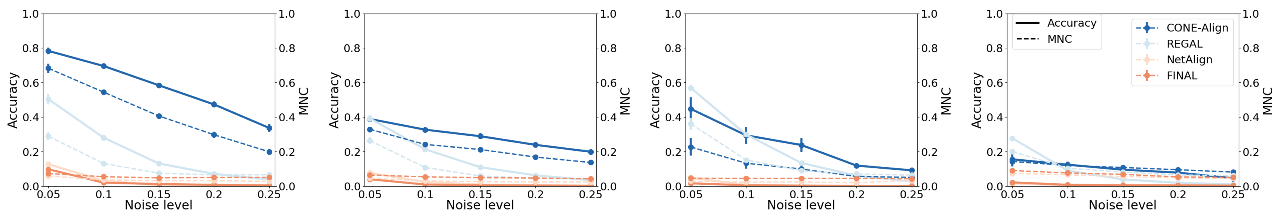
Baselines. Our baselines are unsupervised methods using diverse techniques (belief propagation, spectral methods, and embeddings): **(1) NetAlign** [3] and **(2) FINAL** [25], and **(3) REGAL** [14]. We configure each method following the literature. NetAlign and FINAL require a matrix of prior alignment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

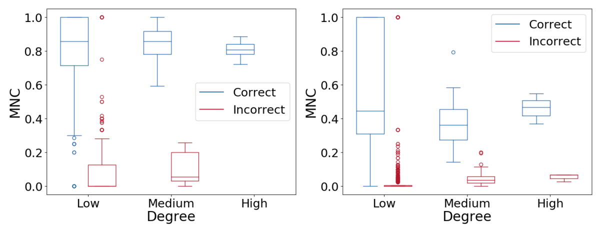
5.1.2 Results. In Fig. 3, we report average and standard deviation for each metric over five trials for each experimental setting.

CONE-Align largely outperforms baselines. We study 5× 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.



(a) Arenas (b) Hamsterster (c) PPI (d) Facebook

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.



(a) MNC with CONE-Align (b) MNC with REGAL

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.

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: $[0, \frac{\Delta^*}{3})$, $[\frac{\Delta^*}{3}, \frac{2\Delta^*}{3})$, $[\frac{2\Delta^*}{3}, \Delta^*]$, where Δ^* 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.

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

[1] Zeyuan Allen-Zhu and Yuanzhi Li. Lazysvd: Even faster svd decomposition yet without agonizing pain. In *NeurIPS*, 2016.
[2] Jason Altschuler, Francis Bach, Alessandro Rudi, and Jonathan Niles-Weed. Massively scalable sinkhorn distances via the nyström method. In *NeurIPS*, 2019.
[3] Mohsen Bayati, Margot Gerritsen, David F Gleich, Amin Saberi, and Ying Wang. Algorithms for large, sparse network alignment problems. In *ICDM*, 2009.
[4] Bobby-Joe Breitkreutz, Chris Stark, et al. The biogrid interaction database: 2008 update. *Nucleic acids research*, 36(suppl 1):D637–D640, 2008.
[5] Marco Cuturi. Sinkhorn distances: Lightspeed computation of optimal transport. In *NeurIPS*, 2013.
[6] Tyler Derr, Hamid Karimi, Xiaorui Liu, Jiejun Xu, and Jiliang Tang. Deep adversarial network alignment. *arXiv preprint arXiv:1902.10307*, 2019.
[7] Xingbo Du, Junchi Yan, and Hongyuan Zha. Joint link prediction and network alignment via cross-graph embedding. In *IJCAI*, 2019.
[8] Marguerite Frank and Philip Wolfe. An algorithm for quadratic programming. *Naval research logistics quarterly*, 3(1-2):95–110, 1956.
[9] Steven Gold and Anand Rangarajan. A graduated assignment algorithm for graph matching. *TPAMI*, 18(4):377–388, 1996.
[10] Palash Goyal and Emilio Ferrara. Graph embedding techniques, applications, and performance: A survey. *Knowledge-Based Systems*, 151:78–94, 2018.
[11] Edouard Grave, Armand Joulin, and Quentin Berthet. Unsupervised alignment of embeddings with wasserstein procrustes. In *AISTATS*, 2019.
[12] Aditya Grover and Jure Leskovec. node2vec: Scalable feature learning for networks. In *KDD*, 2016.
[13] Mark Heimann, Tara Safavi, and Danai Koutra. Distribution of node embeddings as multiresolution features for graphs. In *ICDM*, 2019.
[14] Mark Heimann, Haoming Shen, Tara Safavi, and Danai Koutra. Regal: Representation learning-based graph alignment. In *CIKM*, 2018.
[15] Ehsan Kazemi. Network alignment: Theory, algorithms, and applications. Technical report, EPFL, 2016.
[16] Jérôme Kunegis. Konect: the koblenz network collection. In *WWW*, 2013.
[17] Guillaume Lample, Alexis Conneau, Marc'Aurelio Ranzato, Ludovic Denoyer, and Hervé Jégou. Word translation without parallel data. In *ICLR*, 2018.
[18] Jure Leskovec and Andrej Krevl. SNAP Datasets: Stanford large network dataset collection. http://snap.stanford.edu/data, June 2014.
[19] Li Liu, William K Cheung, Xin Li, and Lejian Liao. Aligning users across social networks using network embedding. In *IJCAI*, 2016.
[20] Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. Deepwalk: Online learning of social representations. In *KDD*, 2014.
[21] Jiezhong Qiu, Yuxiao Dong, Hao Ma, Jian Li, Kuansan Wang, and Jie Tang. Network embedding as matrix factorization: Unifying deepwalk, line, pte, and node2vec. In *WSDM*, 2018.
[22] Leonardo FR Ribeiro, Pedro HP Saverese, and Daniel R Figueiredo. struc2vec: Learning node representations from structural identity. In *KDD*, 2017.
[23] Ryan A Rossi, Di Jin, Sungchul Kim, Nesreen K Ahmed, Danai Koutra, and John Boaz Lee. On proximity and structural role-based embeddings in networks: Misconceptions, methods, and applications. *TKDD*, 2020.
[24] Peter H Schönemann. A generalized solution of the orthogonal procrustes problem. *Psychometrika*, 31(1):1–10, 1966.
[25] Si Zhang and Hanghang Tong. Final: Fast attributed network alignment. In *KDD*, 2016.