Boosting Graph Alignment Algorithms

Alexander Frederiksen Kyster
Aarhus University

Simon Daugaard Nielsen
Aarhus University

Judith Hermanns
Aarhus University

Davide Mottin
Aarhus University

Panagiotis Karras
Aarhus University

ABSTRACT

The problem of graph alignment is to find corresponding nodes between a pair of graphs. Past work has treated the problem in a *monolithic* fashion, with the graph as input and the alignment as output, offering limited opportunities to adapt the algorithm to task requirements or input graph characteristics. Recently, node embedding techniques are utilized for graph alignment. In this paper, we study two state-of-the-art graph alignment algorithms utilizing node representations, CONE-Align and GRASP, and describe them in terms of an overarching modular framework. In a targeted experimental study, we exploit this modularity to develop enhanced algorithm variants that are more effective in the alignment task.

CCS CONCEPTS

• **Computing methodologies** → **Spectral methods**; • **Information systems** → **Similarity measures**; *Nearest-neighbor search*; • **Mathematics of computing** → *Graph theory*.

KEYWORDS

Graph Alignment; Graph Mining; Graph Matching; Network Alignment

ACM Reference Format:
Alexander Frederiksen Kyster, Simon Daugaard Nielsen, Judith Hermanns, Davide Mottin, and Panagiotis Karras. 2021. Boosting Graph Alignment Algorithms. 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, 5 pages. https://doi.org/10.1145/3459637.3482067

1 INTRODUCTION

Graphs expressively represent real-world entities and their interrelationships. Graph alignment [12, 16] finds the correspondence of nodes a pair of graphs, whereby nodes in one graph are aligned to those in another by a similarity criterion. The identification of such correspondences finds applications in science and industry, as in identifying functionally similar proteins [1] and matching [30] or de-anonymizing [19, 29] users across social networks.

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

ACM ISBN 978-1-4503-8446-9/21/11…$15.00
https://doi.org/10.1145/3459637.3482067

Early graph alignment methods were *monolithic*, i.e., hard to adapt to the diverse graph types. Such algorithms, employing matrix factorization [16, 17] or integer programming [12], are typically designed for specific graph types, e.g., biological [1] or bipartite networks [13] and fare poorly on other types [9]. Recent developments in node embeddings [22, 24] have led to a new breed of *modular* graph alignment methods [4, 9, 10], which, as we observe, may be described by the following *modular framework*:

(1) **Embed.** Compute an embedding for each node.
(2) **Align.** Align the embedding spaces of both graphs so that similar nodes are close to each other in the common space.
(3) **Assign.** Match the transformed embeddings by some linear assignment algorithm.

In this paper, we study ways to enhance each part of this modular framework. We interpret three modular alignment algorithms, REGAL [9], CONE-Align [4], and GRASP [4] in terms of the framework. We propose improvements on the two most recent algorithms, CONE-Align and GRASP, interchanging, enhancing, and adding to framework components. In a targeted experimental study, we compare CONE-Align and GRASP to each other for the first time, and show that our enhancements improve upon the state-of-the-art effectiveness with nearly no impact on efficiency.

2 RELATED WORK

Here, we discuss *monolithic* alignment methods, that cannot be broken into atomic operations, and *modular* ones, that can.

2.1 Monolithic Alignment

In monolithic methods, the path from the graph as input to the alignment as output is non-adaptable; intermediate steps solve objectives designed for assumed characteristics of the graph. FINAL [30] optimizes an objective that takes into account topological and attribute information, yielding a pairwise similarity matrix between nodes. IsoRank [25] aligns protein-protein-interaction networks relying on a similarity matrix encoding pairwise protein similarity. EigenAlign [7] interprets graph alignment as a quadratic assignment problem, solved by spectral-decomposing the adjacency matrix of a matching graph. As this method is computationally prohibitive, LowRankEigenAlign [17] proposes a low rank approximation, but its objective remains inflexible. Klau's algorithm [12] interprets graph alignment as an integer program and presents a Lagrangian relaxation thereof. BigAlign [13] is restricted to bipartite graphs.

2.2 Modular Alignment

These methods break the alignment problem into three basic operations: first, they compute a representation of each node of both graphs, exploiting advances in neural graph embeddings [22, 23, 26,

27], then align embeddings, and eventually match nodes. REGAL [9] extracts embeddings based on node ego-networks and, optionally, attributes. To align such embeddings across graphs, it factorizes a similarity matrix of distances from each node to a set of randomly chosen landmark nodes. For the final matching, it returns the top-k matches to each node by Euclidean distance. CONE-Align [4] uses an off-the-shelf embedding, NetMF [23], aligns the embedding subspaces, and matches nearest neighbors by Euclidean distance. GRASP [10] computes embeddings based on the heat kernels of graph Laplacians after aligning the eigenvector spaces and matches nodes using a variant of the Hungarian algorithm [11].

3 PRELIMINARIES

Graph Alignment. Given two undirected graphs $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$, a *graph alignment* is a function that assigns to each node on G_1 a node of G_2.

Node Embedding. An *embedding* $f(v) \in \mathbb{R}^k$ of node v is a vector representation of v.

Graph Laplacian. The normalized *Laplacian* of a graph $G = (V, E)$ is a linear operator, $\mathcal{L} = I - D^{-\frac{1}{2}}AD^{-\frac{1}{2}}$, where A is the adjacency matrix of G and D is the degree matrix with $D_{ii} = \sum_{j=1}^{n} A_{ij}$. The eigendecomposition of this matrix is $\mathcal{L} = \Phi\Lambda\Phi^\top$, where Λ is a diagonal matrix of eigenvalues; its diagonal, $\{\lambda_1, \ldots, \lambda_n\}$ is the *spectrum* of $\mathcal{L}$ and $\Phi_{\mathcal{L}} = [\phi_1\phi_2 \ldots \phi_n]$ is a matrix of eigenvectors. Eigenvectors can be interpreted as functions [20] that assign a real value to each node.

4 MODULAR ALIGNMENT FRAMEWORK

Our modular graph alignment framework comprises three steps:

Step 1: Embed. Compute vector representations of nodes. A good representation encodes the connectivity structure or neighborhood [22].

Step 2: Align. Refit the node representations among two graphs, so that corresponding nodes have comparable representations.

Step 3: Assign. Match node representations so as to minimize a cost function; this step corresponds to a *linear assignment* between the nodes in one graph to those nodes in the other graph.

We interpret three modular alignment methods, REGAL [9], CONE-Align [4], and GRASP [10], using this modular framework.

4.1 REGAL

Embed. Embeddings represent the weighted degree histograms of ego-networks. Optionally, they encode node attributes.

Align. REGAL aligns embeddings by factorizing a pairwise similarity matrix, employing a modified Nyström method.

Assign. REGAL returns top-k-alignments by Euclidean distance, assisted by a k-d tree [2].

4.2 CONE-Align

Embed. CONE-Align supports any *neighborhood preserving* embedding; as a default, it uses NetMF [23].

Align. CONE-Align refits embeddings to preserve *neighborhood consistency* across graphs, i.e., the embeddings of corresponding nodes across the two graphs should preserve the corresponding neighbors by jointly optimizing row-wise node correspondence and column-wise embedding space rotation by means of the objective $\min_{Q \in \mathcal{O}^d} \min_{P \in \mathcal{P}^n} ||Y_1 Q - PY_2||_2^2$, where Y_1 and Y_2 are the two initial embedding matrices, Q is a linear embedding transformation, P a permutation matrix, and $\mathcal{O}$ and $\mathcal{P}$ the set of admissible linear transformation matrices and permutation matrices.

Assign. The final embeddings are matched by Euclidean distance.

4.3 GRASP

Embed. GRASP builds embeddings from graph's Laplacian eigenvectors, $\phi_1, \ldots, \phi_n$; the embedding of v_i is $[\phi_1(i), \phi_2(i), \ldots, \phi_k(i)]$ for some $k \leq n$.

Align. GRASP interprets the embeddings as linear combinations of eigenvectors and maps the coefficients of the respective *corresponding functions* across the two graphs via a diagonal matrix C, hence aligning eigenvectors to preserve orthogonality.

Assign. GRASP matches nodes via the Jonker-Volgenant algorithm [11] for the weighted assignment problem.

5 ENHANCEMENTS

We introduce alternative components for the modular framework.

PageRank (PR): We develop an embedding based on PageRank [21]. PR is real-valued vector that represents the importance of each node, defined as $\mathbf{r}_j = \sum_{(i,j) \in E} \alpha \frac{\mathbf{r}_i}{d_i} + (1-\alpha)\mathbf{p}$, where $\alpha \in [0, 1]$ is the damping factor and $p_i = 1/n$ for each i. We generate corresponding functions sampling PR at different α values. We also use *Personalized PageRank* (PPR), which defines a non-uniform restart probability vector $\mathbf{p}$, and construct corresponding functions by Algorithm 1.

Algorithm 1 PPR-based Corresponding Function

1: Compute PageRank for graph G.
2: Sort PageRank values in descending order.
3: Split nodes into q groups according to PageRank values.
4: For each group, compute PPR with a $\mathbf{p}$ personalized per group.
5: Set the corresp. func. of v_i as its vector of group-PPR values.

Iterative Closest Point aligns two sets of data points in matrices A, B. In each iteration, it assumes each column in A corresponds to one in B and finds an alignment matrix C that minimizes pairwise distances between corresponding columns, until convergence:

(1) Find, for each point x in $C_0 A$, the closest point y in B by Euclidean distance.
(2) Compute an orthonormal C minimizing $\sum_{x,y} ||Cx - y||$.
(3) Set $C_0 = C$.

Voting Heuristic: Instead of fixing the number of eigenvectors k, we apply GRASP in k iterations, increasing the number of eigenvectors. In each iteration, we collect a vote for the match of each node v_i, and determine the final alignment by majority voting.

6 EXPERIMENTS

We study the effect of different components in CONE-Align [4] and GRASP [10]. Table 1 gathers the characteristics of the data sets we use. Unless stated otherwise, the settings for Arenas Email, Facebook-ego, Hamsterer and PPI is as follows: we permute the original graph and delete edges with probability (noise level) p. For

Dataset	$\|V\|$	$\|E\|$	Network type
Arenas Email [14]	1 133	5 451	communication
Facebook-ego [15]	4 039	88 234	social
Hamsterer [14]	2 000	16 097	social
PPI [3]	3 852	38 705	biological
MultiMagna [28]	1004	8323	biological
HighSchool [8]	327	5818	proximity
Voles [5]	712	2391	proximity

Table 1: Datasets used in our evaluation, $|V|$ number of nodes, $|E|$ number of vertices.

each noise level, we create 5 graphs and report average accuracy in terms of matching ground truth nodes, as in [9, 13]. Experiments ran on an Intel Core i9 3.3GHz 14-Core CPU with 256GB RAM.

We first compare GRASP and CONE-Align in their default forms on two data sets. Figure 1 shows the results. GRASP outperforms CONE on Facebook and CONE outperforms GRASP on Arenas.

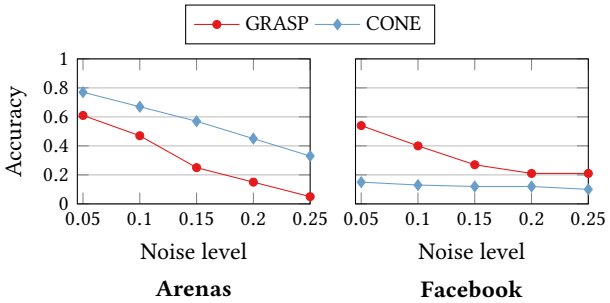

Figure 1: Accuracy of GRASP and CONE-Align in their default settings on the Arenas and Facebook datasets.

6.1 Changing components

We study the impact of different component choice on accuracy.

Boosting Embed. We first investigate the impact of different node embeddings on alignment accuracy, trying out Heat Kernel (HK), PageRank (PR), Personalized PageRank (PPR) and NetMF. In GRASP, embeddings are corresponding functions, thus they affect both Embed and Align. The default embedding in GRASP is the diagonal of the *heat kernel* $H_t = \Phi e^{-t\Lambda}\Phi^\top = \sum_{j=1}^{n}$ at different values of t. The default embedding in CONE is NetMF [23].

Figure 2 shows the accuracy for CONE on Arenas and Facebook. CONE exhibits high sensitivity to the choice of embeddings, especially on Arenas. While NetMF performs best on Arenas on Facebook, PR and HK outpeforms NetMF on low noise.

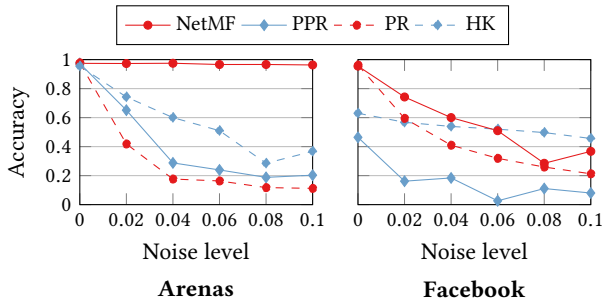

Figure 2: Accuracy of CONE-Align with different embeddings on the Arenas and Facebook datasets.

Figure 3 juxtaposes different node embeddings for GRASP, using the more stable version of GRASP with ICP, voting heuristic and JV. PPR dominates over the other embeddings, while NetMF displays alternating performance in Arenas and Facebook. Henceforward, we use GRASP-PPR as the variant of choice.

Boosting Align. Next, we investigate the impact of the enhancements in Section 5 on GRASP-PPR. Figure 4 shows the accuracy for Iterative Closest Point (ICP) and Base Alignment (BA). The combination of both ICP and BA yields the best performance.

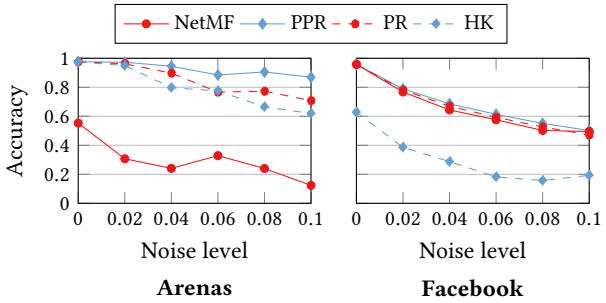

Figure 3: Accuracy of GRASP (ICP, voting and JV) with PPR, PR, HK and NetMF embeddings.

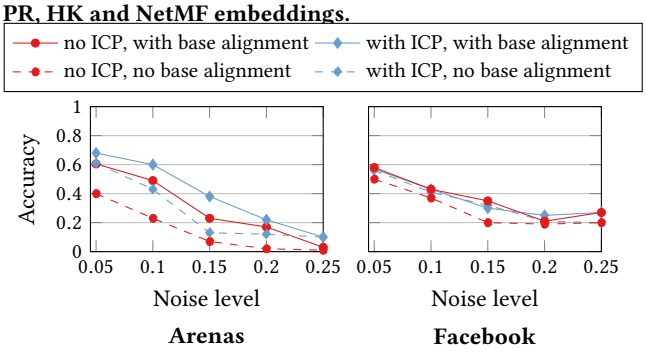

Figure 4: Accuracy of GRASP-PPR with ICP and BA.

Figure 5 showcases the performance of voting with different value intervals for k, using SortGreedy (SG) [6] both for node assignment while collecting votes and for the final assignment based on collected votes; other, more computationally demanding choices brought negligible improvements.

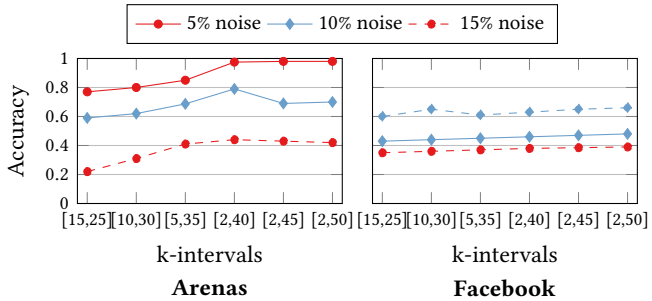

Figure 5: Accuracy of GRASP-PPR with ICP using the SortGreedy voting procedure with different intervals of k

Boosting Assign. We now study the impact of the linear assignment algorithm. We try out nearest-neighbor (NN), SortGreedy (SG) [6] and Jonker-Volgenant (JV). Figure 6 shows the results; performance is similar across datasets. SG fares similar to JV, while yielding better runtime. We observe that GRASP-PPR is sensitive on high noise levels, independent of the assignment algorithm, but outperforms CONE on lower noise levels.

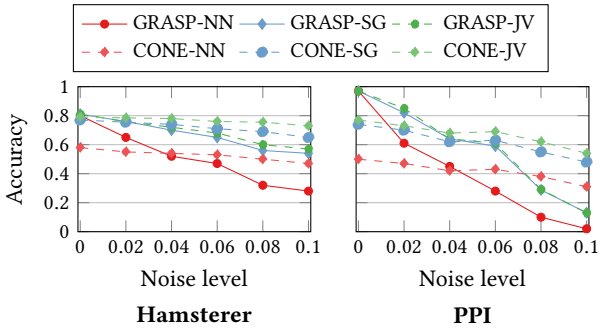

Figure 6: Accuracy with different assignment algorithms.

6.2 Impact of degree distribution

The graphs used in the experiments approximately follow a power law distribution. We estimate the power law exponents using the method in [18]. Table 2 lists the results.

Graph	*Power Law Exponent (γ)*
Arenas	1.56
Facebook	1.32
Hamsterster, PPI	1.45
Voles	1.64
MultiMagna	1.46
HighSchool	1.36

Table 2: Estimated power law exponents.

We study the impact of the power-law exponent on alignment accuracy. Figure 7 shows the performance for three synthetic power-law graphs vs. noise. GRASP achieves nearly 100% accuracy with low exponent, which explains the good performance in the previous experiments. GRASP falls short on graphs with highly skewed distribution, while CONE exhibits a more consistent performance.

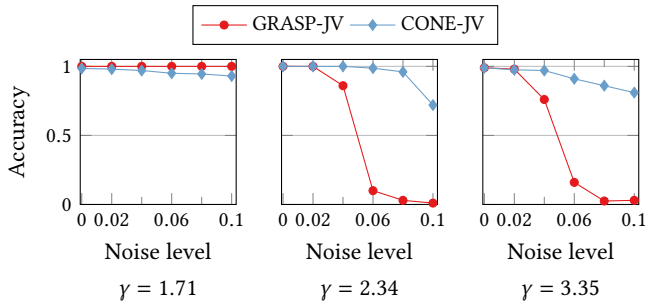

Figure 7: Accuracy of GRASP-PPR and CONE with JV assignment on generated power law graphs.

6.3 Real World Networks

So far we have analyzed behavior on synthetic noise. Here we consider networks in which noisy modifications occur naturally:

- **Voles** [5]. A time-evolving wildlife contact network. The full graph is matched to snapshots at time steps where 80%, 85%, 90%, and 99% of edges have been added.
- **HighSchool** [8]. A time-evolving graph of contact patterns among high school students. The full graph is matched to snapshots with 80%, 85%, 90% and 99% of edges added.
- **MultiMagna** [28]. Different versions of the same protein-protein-interaction network.

Figure 8 shows results on these real world networks. CONE-Align consistently outperforms GRASP-PRR with ICP and Voting.

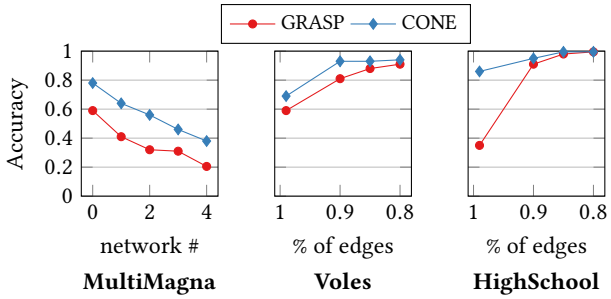

Figure 8: Accuracy on real data sets.

6.4 Efficiency

We now compare the efficiency of best-of-breed versions, namely GRASP-PPR with ICP and voting and CONE with NetMF embeddings, across Assign variants. Figure 9 shows the runtime for each steps. For GRASP, we report precomputation including Embed and Align, and the time for voting. For CONE-Align, we report Embed, Align and Assign, separately. While CONE-Align is faster than GRASP-PPR on small graphs, its Align is slower than BA and ICP; therefore, GRASP-PPR outperforms CONE-Align on large enough graphs, especially when using nearest neighbor matching.

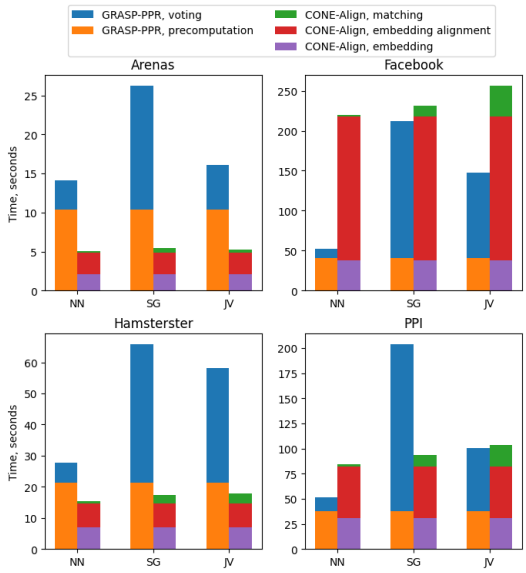

Figure 9: Time with GRASP-PPR (ICP, voting) and CONE with three matching methods on four datasets.

7 CONCLUSION

We revisited state-of-the-art graph alignment algorithms and identified an overarching modular framework that allows for components exchange. By virtue of this framework, these algorithms are amenable to several enhancements. We offered and analyzed alternatives that improve performance vs. that of default variants, as our experiments with synthetic and real noise corroborate. Our results pave the way to further advances in graph alignment.

REFERENCES

[1] Ahmet E Aladağ and Cesim Erten. 2013. SPINAL: scalable protein interaction network alignment. *Bioinformatics* 29, 7 (2013), 917–924.

[2] Jon Louis Bentley. 1975. Multidimensional binary search trees used for associative searching. *CACM* 18, 9 (1975), 509–517.

[3] Paul J Besl and Neil D McKay. 1992. Method for registration of 3-D shapes. In *Sensor fusion IV: control paradigms and data structures*, Vol. 1611. International Society for Optics and Photonics, 586–606.

[4] Xiyuan Chen, Mark Heimann, Fatemeh Vahedian, and Danai Koutra. 2020. CONE-Align: Consistent Network Alignment with Proximity-Preserving Node Embedding. In *CIKM*. 1985–1988.

[5] Stephen Davis, Babak Abbasi, Shrupa Shah, Sandra Telfer, and Mike Begon. 2015. Spatial analyses of wildlife contact networks. *Journal of the Royal Society, Interface* 12, 102 (2015).

[6] Katerina Doka, Mingqiang Xue, Dimitrios Tsoumakos, and Panagiotis Karras. 2015. k-Anonymization by freeform generalization. In *AsiaCCS*. 519–530.

[7] Soheil Feizi, Gerald Quon, Mariana Mendoza, Muriel Medard, Manolis Kellis, and Ali Jadbabaie. 2019. Spectral alignment of graphs. *IEEE Trans. Netw. Sci.* (2019).

[8] Julie Fournet and Alain Barrat. 2014. Contact patterns among high school students. *PloS one* (2014).

[9] Mark Heimann, Haoming Shen, Tara Safavi, and Danai Koutra. 2018. REGAL: Representation Learning-based Graph Alignment. In *CIKM*. 117–126.

[10] Judith Hermanns, Anton Tsitsulin, Marina Munkhoeva, Alexander Bronstein, Davide Mottin, and Panagiotis Karras. 2021. GRASP: Graph Alignment through Spectral Signatures. In *APWeb-WAIM*.

[11] Roy Jonker and Anton Volgenant. 1987. A shortest augmenting path algorithm for dense and sparse linear assignment problems. *Computing* 38, 4 (1987), 325–340.

[12] Gunnar W. Klau. 2009. A new graph-based method for pairwise global network alignment. *BMC Bioinf.* 10, S-1 (2009).

[13] Danai Koutra, Hanghang Tong, and David Lubensky. 2013. BIG-ALIGN: Fast Bipartite Graph Alignment. In *ICDM*. 389–398.

[14] Jérôme Kunegis. 2013. Konect: the Koblenz network collection. In *WWW*. ACM, 1343–1350.

[15] Jure Leskovec and Andrej Krevl. 2014. SNAP Datasets: Stanford Large Network Dataset Collection. http://snap.stanford.edu/data.

[16] Chung-Shou Liao, Kanghao Lu, Michael Baym, Rohit Singh, and Bonnie Berger. 2009. IsoRankN: spectral methods for global alignment of multiple protein networks. *Bioinformatics* 25, 12 (2009), i253–i258.

[17] Huda Nassar, Nate Veldt, Shahin Mohammadi, Ananth Grama, and David F. Gleich. 2018. Low Rank Spectral Network Alignment. In *WWW*. 619–628.

[18] Mark EJ Newman. 2005. Power laws, Pareto distributions and Zipf's law. *Contemporary physics* 46, 5 (2005), 323–351.

[19] Sadegh Nobari, Panagiotis Karras, HweeHwa Pang, and Stéphane Bressan. 2014. L-opacity: Linkage-Aware Graph Anonymization. In *EDBT*. 583–594.

[20] Maks Ovsjanikov, Mirela Ben-Chen, Justin Solomon, Adrian Butscher, and Leonidas J. Guibas. 2012. Functional maps: a flexible representation of maps between shapes. *ACM Trans. Graph.* 31, 4 (2012), 30:1–30:11.

[21] Lawrence Page, Sergey Brin, Rajeev Motwani, and Terry Winograd. 1999. *The PageRank citation ranking: Bringing order to the web.* Technical Report. Stanford InfoLab.

[22] Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. 2014. Deepwalk: Online learning of social representations. In *KDD*. 701–710.

[23] Jiezhong Qiu, Yuxiao Dong, Hao Ma, Jian Li, Kuansan Wang, and Jie Tang. 2018. Network embedding as matrix factorization: Unifying DeepWalk, LINE, PTE, and node2vec. In *WSDM*. 459–467.

[24] Leonardo FR Ribeiro, Pedro HP Saverese, and Daniel R Figueiredo. 2017. struc2vec: Learning node representations from structural identity. In *KDD*. 385–394.

[25] Rohit Singh, Jinbo Xu, and Bonnie Berger. 2008. Global alignment of multiple protein interaction networks with application to functional orthology detection. *PNAS* 105, 35 (2008), 12763–12768.

[26] Anton Tsitsulin, Davide Mottin, Panagiotis Karras, and Emmanuel Müller. 2018. VERSE: Versatile graph embeddings from similarity measures. In *TheWebConf*. 539–548.

[27] Anton Tsitsulin, Marina Munkhoeva, Davide Mottin, Panagiotis Karras, Ivan V. Oseledets, and Emmanuel Müller. 2021. FREDE: Anytime Graph Embeddings. *Proc. VLDB Endow.* 14, 6 (2021), 1102–1110.

[28] Vipin Vijayan and Tijana Milenković. 2017. Multiple network alignment via multiMAGNA++. *IEEE/ACM Transactions on Computational Biology and Bioinformatics* 15, 5 (2017), 1669–1682.

[29] Mingqiang Xue, Panagiotis Karras, Chedy Raïssi, Panos Kalnis, and Hung Keng Pung. 2012. Delineating social network data anonymization via random edge perturbation. In *CIKM*. 475–484.

[30] Si Zhang and Hanghang Tong. 2016. FINAL: Fast Attributed Network Alignment. In *KDD*. 1345–1354.