Experiments & Analysis Paper

Comprehensive Evaluation of Algorithms for Unrestricted Graph Alignment

Konstantinos Skitsas
Aarhus University
Aarhus, Denmark
201903630@post.au.dk

Karol Orłowski
Aarhus University
Aarhus, Denmark
karolplg@gmail.com

Judith Hermanns
Aarhus University
Aarhus, Denmark
judith@cs.au.dk

Davide Mottin
Aarhus University
Aarhus, Denmark
davide@cs.au.dk

Panagiotis Karras
Aarhus University
Aarhus, Denmark
panos@cs.au.dk

ABSTRACT

The graph alignment problem calls for finding a matching between the nodes of one graph and those of another graph, in a way that they correspond to each other by some fitness measure. Over the last years, several graph alignment algorithms have been proposed and evaluated on diverse datasets and quality measures. Typically, a newly proposed algorithm is compared to previously proposed ones on some specific datasets, types of noise, and quality measures where the new proposal achieves superiority over the previous ones. However, no systematic comparison of the proposed algorithms has been attempted on the same benchmarks. This paper fills this gap by conducting an extensive, thorough, and commensurable evaluation of state-of-the-art graph alignment algorithms. Our results highlight the value of overlooked solutions and an unprecedented effect of graph density on performance, hence call for further work.

1 INTRODUCTION

Graphs provide a general means to model relationships between entities in diverse areas of society, science and industry [48], where entities are represented as nodes and relationships between entities as edges. Whereas the graph of a social network might picture which users follow each other [10], a protein-protein interaction (PPI) network represents the interaction of proteins associated with a biological species [46, 55].

Despite this diversity of application domains, several graph analytics tasks are universally relevant. Such a task is *graph alignment*, a generalized version of the graph isomorphism problem that aims to find, for each node in a graph $G_A(V_A, E_A)$, a structurally corresponding node in another graph $G_B(V_B, E_B)$, i.e., a function $f : V_A \rightarrow V_B$. This problem is aimed at in many applications, e.g., aligning entities of social networks, biological structures, or the intersections of a road network at different time stamps, as well as a foundation for further analysis of two aligned graphs.

This diversity of application domains also implies a challenge in applying a graph alignment algorithm to a problem: while the problem is fundamentally the same across application domains and graph data, the particular semantics of a good alignment differ across domains. For example, whereas in the case of social networks we may be interested in re-identifying the *same* user in two or more different networks, in the case of protein interaction (PPI) networks of different species, we may be interested to understand which proteins perform *similar roles* in diverse species. Given these multifarious application requirements, we draw a principle distinction between what we call *restricted* and *unrestricted* graph alignment.

Restricted alignment requires domain-specific input information, such as a set of pre-aligned users in a social [23] or biological network [43], knowledge of protein sequences in PPI networks [3, 33, 43, 50], special types of graphs [28], or node attributes [61]. Several domain-specific *restricted* graph alignment algorithms have been developed, which, in principle, could be applied across domains. For example, the requirement for a set of pre-aligned nodes could also be fulfilled in a PPI network if there is prior knowledge available for a set of the proteins. However, in practice this is rarely done. The literature on graph alignment is extensive, but surprisingly disconnected as algorithms from different fields and for slightly different problem definitions are not compared to each other. There is no universal evaluation methodology for graph alignment algorithms. Metrics, the graphs evaluated on and experimental design differ from domain to domain.

Unrestricted alignment, on the other hand, finds correspondences among nodes in two graphs using nodes and edges, but no additional information (e.g., annotations or labels).

Existing experimental evaluations of graph alignment algorithms usually focus on restricted methods for a specific domain, such as biology [9] or neuroscience [36]; these solutions require additional domain information, such as protein affinity or brain functions. A recent comparative study [53] reviews seven graph alignment techniques and features experiments mixing restricted and unrestricted graph alignment, hence cannot be conclusive.

In this experimental study, we go beyond [53] and bridge the gap in previous literature, conducting the *first*, to our knowledge, exhaustive, extensive, and in-depth comparative evaluation of *unrestricted* graph alignment algorithms under three noise types, three assignment algorithms, and diverse datasets; these algorithms were not part of the study in [53], apply on undirected, unattributed graphs, and do not require any prior alignment information, hence can be generalized across domains. We assess techniques, measures, noise generators, and parameters to compare the best versions of each algorithm. Thus, our study complements [53]. In detail, our contributions are the following:

- We perform the first, complete, experimental evaluation of nine undirected, unattributed graph alignment algorithms.

- We carefully tune the algorithms hyperparameters based on network size and using the same assignment algorithm.
- We evaluate the algorithms on real and synthetic graphs, with different levels and types of noise.
- We include memory and time scalability experiments on the algorithms.
- We devise an experimental framework for graph alignment with reproducible experiments and available data and code.

2 PRELIMINARIES

Graph Alignment. Given two undirected graphs $G_A = (V_A, E_A)$ and $G_B = (V_B, E_B)$, a *graph alignment* is a function $f : V_A \rightarrow V_B$ that assigns to each node on G_A a node of G_B. G_A is often referred to as *source graph* and G_B as *target graph*.

Adjacency matrix. $\mathbf{A}$ is the *adjacency matrix* of G_A with $\mathbf{A}_{ij} = 1$ if $(i, j) \in E_A$ and 0 otherwise. Similarly, $\mathbf{B}$ is G_B's adjacency matrix.

Neighborhood. The neighborhood $N(i)$ of node $i \in V$ is the set of node j such that $(i, j) \in E$ for graph $G = (V, E)$.

Graph Laplacian. The normalized *Laplacian* of a graph $G = (V, E)$ is a linear operator, $\mathcal{L} = I - \mathbf{D}^{-\frac{1}{2}}\mathbf{A}\mathbf{D}^{-\frac{1}{2}}$, where $\mathbf{A}$ is the adjacency matrix of G and $\mathbf{D}$ is the degree matrix with $\mathbf{D}_{ii} = \sum_{j=1}^{n} \mathbf{A}_{ij}$. 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.

3 ALIGNMENT ALGORITHMS

We focus on algorithms for *unrestricted graph alignment*, which find a correspondence among the nodes of two graphs without using information other than nodes and edges.

While all graph alignment algorithms aim to find corresponding nodes in different networks, they may start out from a different objective formulation. Typically, the problem is stated as one of finding a correspondence that maximizes a similarity measure or minimizes the cost of transforming the second graph to the first. We outline three prerequisites of a graph alignment algorithms.

Input Data. The most basic input data are the adjacency matrices of two graphs to be aligned. Some algorithms may require additional information as input, such as a similarity matrix or known prior alignments. In the biology domain, such pre-processed input is needed as nodes typically have roles and semantics beyond structural properties.

Similarity Notion. A graph alignment algorithm needs to translate the information in the input graphs to some representation from which a similarity score can be computed. This may be done by computation a graph embedding method [34] or some graph function such as a heat kernel [5]. The computation may be done for each graph separately, or concurrently on two graphs. In all cases, the representations are combined and yield a similarity matrix.

Assignment. The last operation is to extract the alignments based on the provided similarity matrix by solving a Linear Assignment Problem (LAP) on a bipartite graph among nodes of the two graphs on the two sides, weighted according to similarity scores. Previous work considers four assignment algorithms: nearest neighbor (NN) [8, 19, 59, 60] that returns the unmatched node with the highest similarity, SortGreedy (SG) [27, 50] that progressively matches similarity-sorted pairs of nodes, the Maximum Weighted Matching (MWM) [3, 25, 38] and the Jonker-Volkenant (JW) [20] that directly solve the linear assignment problem.

Table 1 collects the characteristics of the algorithms we consider; IsoRank and GRAAL require preprocessing and target biological networks; assignment algorithms are as proposed in the respective papers; CONE optimizes for the MNC measure (Section 5.2.1); time complexity is expressed in the number of nodes n; the presented hyperparameters are obtained via grid search on real graphs.

Algorithm	Year	Prepr.	Bio	Assign	Opt	Time	Parameters
IsoRank [50]	2008	Yes	Yes	SG	Any	$O(n^4)$	$\alpha = 0.9$
GRAAL [29]	2010	Yes	No	SG	Any	$O(n^3)$	$\alpha = 0.8$
NSD [27]	2011	Both	No	SG	Any	$O(n^2)$	$\alpha = 0.8$
LREA [38]	2018	No	No	MWM	Any	$O(n \log n)$	iterations=40
REGAL [19]	2018	No	No	NN	Any	$O(n \log n)$	k=2, p=10 log n
GWL [60]	2019	No	No	NN	Any	$O(n^3)$	epoch=1
S-GWL [59]	2019	No	No	NN	Any	$O(n^2 \log n)$	$\beta \in \{0.025, 0.1\}$
CONE [8]	2020	No	No	NN	MNC	$O(n^2)$	dim=512
GRASP [20]	2021	No	No	JV	Any	$O(n^3)$	q=100, k=20

Table (1) Algorithms considered in the experiments.

3.1 IsoRank

IsoRank [50] is the first network alignment algorithm applied on biological networks; it assumes similarity scores are provided by the Blast algorithm [24]. IsoRank uses neighborhood similarity to extract structural graph information, inspired from the way PageRank [7] uses neighborhood topology (links) to rank nodes. A node from target graph G_B is considered a possible match to a node from source graph G_A if their neighbors are also possible matches to each other. The score of a node pair recursively depends on the score of their neighbors. Equation 1 shows how the similarity for node pairs is calculated, where $\mathbf{R}$ is an iteratively updated matrix of pairwise node similarities between nodes in G_A and those in G_B, while matrix $\mathbf{M}$ captures the topological similarity of the two graphs based on their edges having weight w. $\mathbf{R}_{ij}$ for $i \in V_A$, $j \in V_B$ is:

$$\mathbf{R}_{ij} = \sum_{u \in N(i)} \sum_{v \in N(j)} \frac{w(i,u)w(j,v)}{\sum_{r \in N(u)} w(r,u) \sum_{q \in N(v)} w(q,v)} \mathbf{R}_{uv} \qquad (1)$$

The problem expressed by Equation 1 is an eigenvalue problem solved by the power method that finds the leading eigenvector.

Further, IsoRank incorporates Blast similarity scores in a matrix $\mathbf{E}$ by linear combination $\mathbf{R} = \alpha\mathbf{M}\mathbf{R} + (1 - \alpha)\mathbf{E}$ with parameter $0 \leq \alpha \leq 1$. A value of α close to 1 gives more weight to topological similarity, while a value close to 0 emphasizes Blast similarity. Global alignments are determined by solving a linear assignment problem via a SortGreedy [12] heuristic. An extension of IsoRank, IsoRankN [33], solves the global multiple network alignment problem, i.e., aligns multiple networks instead of just two.

3.2 GRAph ALigner (GRAAL)

GRAAL [29] is a greedy alignment method that matches nodes using a similarity score based on a dictionary of 73 graphlets with 4 or less nodes. A graphlet is a frequent graph pattern, such as a triangle with three nodes and three edges. In a pre-processing step, GRAAL constructs a *vector signature* for each node, in which each value indicates the number of times a graphlet appears in

the node's neighborhood. This steps takes $O(n^5)$ time. GRAAL aligns a node u in G_A with v in G_B by computing a cost $\mathbf{C}_{uv}$ as a linear combination, with parameter $\alpha \in [0, 1]$, of the vector similarity $S(u, v)$ among signatures and the degrees of u and v:

$$\mathbf{C}_{uv} = 2 - \left((1-\alpha) \cdot \frac{|N(u)| + |N(v)|}{\max\limits_{i \in V_A} |N(i)| + \max\limits_{i \in V_B} |N(i)|} + \alpha \cdot S(u, v) \right) \quad (2)$$

Having created matrix $\mathbf{C}$, GRAAL selects a pair of nodes with the minimum cost as seeds and aligns the nodes induced by the seeds' neighbors up to a certain distance. The alignment proceeds to other seeds by a SortGreedy [12] method until all nodes are aligned.

3.3 Network Similarity Decomposition (NSD)

NSD [27] improves upon both the efficiency and topological similarity capture of IsoRank using the HITS [26] link analysis algorithm. It works well without preprocessing information, yet it can work with preprocessing information by receiving the same Blast similarity matrix as Isorank and getting its singular values by SVD.

The algorithm extends Isorank [6] by approximating the matrix $\mathbf{R}$ in Equation 1 using power iteration, resulting in Equation 3, where α is the importance of preprocessing information, h is the vectorized preprocessing matrix and $\tilde{\mathbf{C}}$=$\tilde{\mathbf{A}} \otimes \tilde{\mathbf{B}}$ is the Kronecker product between the source's degree-normalized adjacency matrix $\tilde{\mathbf{A}} = \mathbf{D}^{-1}\mathbf{A}$ and the target's $\tilde{\mathbf{A}} = \mathbf{D}^{-1}\mathbf{B}$.

$$\mathbf{X}^{(n)} = (1-\alpha) \sum_{k=0}^{n-1} \alpha^k \tilde{\mathbf{C}}^k h + \alpha^n \tilde{\mathbf{C}}^n h \quad (3)$$

Thereafter, by noticing that $\tilde{\mathbf{A}} \otimes \tilde{\mathbf{B}} h = \tilde{\mathbf{B}}^2 \mathbf{H} \tilde{\mathbf{A}}^2$ where $\mathbf{H}$ is the matrix-form of h, NSD decomposes the $\mathbf{C}$ matrix as a linear combination of orthonormal vectors $w_i^{(k)} = \tilde{\mathbf{B}}^k w_i$ and similarly to get the z vector, $z_i^{(k)} = \tilde{\mathbf{A}}^k z_i$, to obtain ranks of $\mathbf{X}^{(n)}$, while integrating preprocessing information expressed by singular values of the Blast similarity matrix in the z vexctor:

$$\mathbf{X}_i^{(n)} = (1-\alpha) \sum_{k=0}^{n-1} \alpha^k w_i^{(k)} z_i^{(k)^\top} + \alpha^n w_i^{(n)} z_i^{(n)^\top} \quad (4)$$

Eventually, it combines ranks to form a dense similarity matrix:

$$\mathbf{X}^{(n)} = \sum_{i=1}^{s} \mathbf{X}_i^{(n)} \quad (5)$$

To extract final alignments, the authors of NSD propose two solutions. One is to enforce a sparse matrix using sparse versions of the outer products w and z; another is to use SortGreedy [12].

3.4 Low-Rank EigenAlign (LREA)

LREA [38] is an alignment method that builds on the EigenAlign [15] algorithm. Whereas the default EigenAlign has very high computational cost, LREA applies a low-rank matrix approximation that dramatically decreases runtime. LREA align graphs of 10 000 nodes in the time EigenAlign needs to align graphs of 1 000 nodes, while achieving similar results in terms of the loss function. LREA aims to solve the following quadratic assignment problem:

$$\begin{aligned} \underset{\mathbf{y}}{\text{maximize}} \quad & \mathbf{y}^\top \boldsymbol{M} \mathbf{y} \\ \text{subject to} \quad & y_i \in \{0, 1\} \\ & \textstyle\sum_u y[u, v] \leq 1 \text{ for all } v \in V_2 \\ & \textstyle\sum_v y[u, v] \leq 1 \text{ for all } u \in V_1 \end{aligned} \quad (6)$$

Matrix $\boldsymbol{M}$ gathers an alignment score based using three elements: overlaps, non-informative, and conflicts (acting as a regularization), and uses the dominant eigenvectors to find the maximum of the objective function. $\boldsymbol{M}$ is rewritten as a linear combination of Kronecker products among the adjacency matrices $\mathbf{A}$ and $\mathbf{B}$ of the two graphs and all-one matrices of the same size $\mathbf{E}$.

Thus, the quadratic assignment problem in Equation 6 is relaxed to the following, using X, the leading eigenvector of $\boldsymbol{M}$:

$$\begin{aligned} \underset{\mathbf{X}}{\text{maximize}} \quad & \mathbf{X} \bullet \left(c_1 \mathbf{A}\mathbf{X}\mathbf{B}^\top + c_2 \mathbf{A}\mathbf{X}\mathbf{E}^\top + c_2 \mathbf{E}\mathbf{X}\mathbf{B}^\top + c_3 \mathbf{E}\mathbf{X}\mathbf{E}^\top \right) \\ \text{subject to} \quad & \|\mathbf{X}\|_F = 1, \mathbf{X} \in \mathbb{R}^{|V_A| \times |V_B|} \end{aligned} \quad (7)$$

This problem is translated to a two-factor low-rank decomposition. Starting from rank 1, a power iteration process calculates the similarity matrix of rank k. To obtain the alignment, LREA separates positive and negative values into two matrices and pairs them based on their sorted positions for each rank, with each rank produces a matching score, eventually reaching the optimal solution to the relaxed problem. The authors also propose a variant that uses a sparse matrix Y with all possible matchings from the ranks.

3.5 REGAL

REGAL [19] aligns graphs fast with high accuracy scores, working in three steps, namely embedding calculation, cross-embedding calculation, and node alignment, without using any pre-processed information. The first step gathers structural and connectivity information about the nodes, taking into account the degree of the neighborhood nodes:

$$\mathbf{d}_u = \sum_{k=1}^{K} \delta^{k-1} \mathbf{d}_u^k \quad (8)$$

Equation 8 iterates over k-hop neighbors of node u and stores neighborhood degrees in the vector $\mathbf{d}_u$; δ is a discount factor that gives less importance to nodes away from u. The authors propose iterating up to a 2-neighborhood ($K = 2$). Besides, degree information is stored in logarithmically scaled buckets. Attribute information is stored in f-dimensional vectors, from which the distance between two nodes' attributes can be estimated; in our study, we focus on aligning graph structures, agnostic to attributes. REGAL evaluates potential node matchings using a cross-network node similarity:

$$\text{sim}(u, v) = \exp\left[-\gamma_s \cdot \|\mathbf{d}_u - \mathbf{d}_v\|_2^2 - \gamma_a \cdot \text{dist}\left(\mathbf{f}_u, \mathbf{f}_v\right) \right] \quad (9)$$

Here, γ_s and γ_α are scalar parameters that weigh the structural and attribute information, respectively. We set γ_α to 0. This cross-network similarity is used to calculate a similarity matrix $\mathbf{S}$ used in the next part, cross-embedding calculation, using the Nyström method for low-rank matrix approximation. The procedure chooses p random landmark nodes from both graphs (p is set to $10 \log_2 n$) and approximates $\mathbf{S}$ as $\mathbf{S}^{\sim} = \mathbf{C}\mathbf{W}^{\dagger}\mathbf{C}^\top$, where $\mathbf{C}$ is

an $n \times p$ node-to-landmark similarity matrix and $\mathbf{W}^{\dagger}$ is a pseudo-inverse landmark-to-landmark similarity matrix. By applying SVD on $\mathbf{W}$ to obtain the decomposition $U\Sigma V^T$, cross-embeddings are derived as $Y' = CU\Sigma^{\frac{1}{2}}$ without fully calcluating the similarity matrix $\mathbf{S}$.

Lastly, REGAL performs node alignment by efficiently querying embeddings, while translating Euclidean distance to the similarity:

$$\text{sim}_{emb}(\tilde{\mathbf{Y}}_1[u], \tilde{\mathbf{Y}}_2[v]) = e^{-\|\overline{\mathbf{Y}}_1[u] - \overline{\mathbf{Y}}_2[v]\|_2^2} \tag{10}$$

The target graph embeddings are stored in a k-d tree for fast nearest-neighbor querying using source graph embeddings. By default, REGAL returns the highest-scoring matching for each source node, hence may return the same target node more than once. In our study, we configure it to return one-to-one matchings for the sake of comparability to other methods.

3.6 Gromov-Wasserstein Learning (GWL)

GWL [60] follows a different approach; instead of first calculating embeddings and then learning alignments from those embeddings, it jointly calculates embeddings $\mathbf{X}_A, \mathbf{X}_B$ and alignments using the dissimilarity notion of Gromov-Wasserstein discrepancy to transport masses from one node to a node in the other graph. The problem is solved collaterally in iterations, using embeddings $\mathbf{X}$ to estimate distances while learning the optimal transport $\mathbf{T}$, and using the learned transport to regularize the learning of embeddings in the next iteration.

$$\min_{\mathbf{X}_A, \mathbf{X}_B} \min_{\mathbf{T} \in \Pi(\boldsymbol{\mu}_A, \boldsymbol{\mu}_B)} \underbrace{\langle L\left(\boldsymbol{C}_A\left(\mathbf{X}_A\right), \boldsymbol{C}_B\left(\mathbf{X}_B\right), \mathbf{T}\right), \mathbf{T}\rangle}_{\text{Gromov-Wasserstein discrepancy}} + \underbrace{\alpha \langle \boldsymbol{K}\left(\mathbf{X}_A, \mathbf{X}_B\right), \mathbf{T}\rangle}_{\text{Wasserstein discrepancy}} + \underbrace{\beta R\left(\mathbf{X}_A, \mathbf{X}_B\right)}_{\text{prior information}} \tag{11}$$

In Equation (11), L is an element-wise loss function, $\mathbf{C}$ is a pairwise node distance computed on edge weights, and $\mathbf{K}$ is a distance matrix on the embeddings. The first part of the equation captures the relational dissimilarity between the two graphs and the second expresses the dissimilarity between nodes of the two graphs. This non-convex problem is solved iteratively by breaking it into two minimization problems. The first is to minimize optimal transport (OT), solved iteratively with the proximal point method. The second problem is to update the embeddings using gradient descent.

As mentioned, the two problems are solved collaterally in alternations: first find the OT, then update the embedding $\mathbf{X}$ using the OT. GWL can thereby align multiple networks.

S-GWL [59] addresses the scalability drawback of GWL by adopting a partitioning method on the input graphs. While the S-GWL objective is the same as GWL, S-GWL recursively decomposes the graphs into K small graphs and calculates a common intermediate graph with K-nodes called a barycenter graph. The algorithm matches the nodes in the barycenter graph with the subgraphs of the graph using GWL and repeats the partitioning process over the subgraphs in a divide-and-conquer fashion, thus achieving a logarithmic speedup over GWL. S-GWL further reduces the times employing a proximal gradient scheme [58] that decomposes the original GWL non-convex objective into smaller convex problems, achieving an additional speedup on sparse graphs.

3.7 CONE

CONE [8] finds alignments by focusing on matched neighborhood consistency (MNC), defined in Section 5.2.1 as the Jaccard similarity among node neighbohoods. It calculates node embeddings $\mathbf{Y}_A$, $\mathbf{Y}_B$ for each graph separately using an off-the-shelf method, and aligns embedding sub-spaces by combining two optimization problems, Procrustes and Wasserstein, to create the following problem:

$$\min_{\mathbf{Q} \in O^d} \min_{\mathbf{P} \in \mathcal{P}^n} \|\mathbf{Y}_A\mathbf{Q} - \mathbf{P}\mathbf{Y}_B\|_2^2 \tag{12}$$

$\mathbf{P}$ is a row permutation found by solving a Wasserstein problem and $\mathbf{Q}$ is a column permutation found by solving a Procrustes problem. The overall problem is initilized by a Frank-Wolf algorithm for 10 iterations. CONE minimizes Wasserstein distance by the Sinkhorn algorithm and updates the Procrustes orthogonal matrix by SVD. This procedure is repeated around 50 times to return the embeddings. The last step is to align nodes to their nearest neighbor by Euclidean distance, by default using a k-d tree as REGAL does.

3.8 GRASP

GRASP [20] solves the alignment problem using the spectral properties of the graphs grounded on the eigenvectors of their normalized Laplacian matrices, in a manner reminiscent of NetLSD [54]. The normalized Laplacian, $\mathcal{L} = I - \mathbf{D}^{-\frac{1}{2}}\mathbf{A}\mathbf{D}^{-\frac{1}{2}}$, converges to the eigenfunctions of Laplace-Berltrami operator [4]. The Heat Kernel, computed based on Laplacian eigenvectors, provides meaningful information about the structure of the graph in a permutation-invariant manner robust to small perturbations:

$$H_t = \Phi e^{-t\Lambda}\Phi^{\top} = \sum_{j=1}^{n} e^{-t\lambda_j}\phi_j\phi_j^{\top} \tag{13}$$

GRASP performs eigendecomposition and calculates heat kernels of the two graphs for different values of t, forms matrices $\mathbf{F}$ and $\mathbf{G}$ from the diagonals of those heat kernels; it produces linear combinations of the two graphs' Laplacian eigenvectors, contained in matrices Φ and Ψ, using $\mathbf{F}$ and $\mathbf{G}$, and aligns the eigenvector matrices Φ and Ψ via a diagonal base alignment matrix $\mathcal{M}$ that minimizes the objective:

$$\min_{\mathcal{M} \in \mathcal{S}(n,n)} \text{off}\left(\mathcal{M}^{\top}\Lambda_2\mathcal{M}\right) + \mu\left\|\mathbf{F}^{\top}\Phi - \mathbf{G}^{\top}\Psi\mathcal{M}\right\|_F^2 \tag{14}$$

This objective strives for the diagonality of $\mathcal{M}$ by penalizing off-diagonal elements via the first (diagonalization) term. Next, it approximates a diagonal matrix $\mathbf{C}$ that maps those linear combinations of Laplacian eigenvectors from the one graph to the other, using the top-k aligned eigenvectors, $\bar{\Phi}$ and $\hat{\Psi} = \bar{\Psi}\mathcal{M}$.

The matching problem is solved as a Linear Assignment Problem (LAP); the authors chose the JV [22] algorithm for this purpose.

4 EXCLUDED ALGORITHMS

In our preparatory assessment, we initially considered a larger collection of algorithms. However, we soon realized that some algorithms cannot adapt to the unrestricted scenario, or cannot run in reasonable time on graphs with more than 100 nodes.

There is an extensive body of work on graph alignment algorithms for biological networks, which includes GHOST [44], MAGNA [49], and NETAL [42]. We exclude these methods as they require $O(n^4)$ or higher time complexity that hinders their application outside biology, or additional information, such as protein-protein sequence similarity. We exclude two popular alignment algorithms, Klau [25] due to its $O(n^5)$ time complexity and NetAlign [3] as we observed inadequate quality even after we applied the enhancements granted to the rest of algorithms, including the IsoRank similarity notion described in Section 6.1 and the JV assignment algorithm described in Section 6.2. Likewise, we exclude methods in which additional information is part of the algorithm definition, such as FINAL [61], BigAlign [28], and the extensive literature on supervised algorithms that require pre-aligned nodes.

5 EXPERIMENTAL SETUP

All algorithms other than GRAAL are implemented in Python 3.8.5. We use the authors' original Python codes, whether available, and re-implement algorithms without an available implementation; in the case of GRAAL, we use the authors' executable. Our code[1] uses the Sacred framework [18] to easily test and tune algorithms.

We use a 28-core Intel Core CPU i9-10940X machine at 3.30 GHz with 256Gb RAM on Linux 5.4.0-74.

5.1 Datasets

We evaluate the algorithms with both real and synthetic graphs where the edges have been perturbed with a fixed amount of noise and nodes have been permuted. Moreover, to further investigate the the behaviour of the algorithms under different conditions, we employ three types of noise. For each graph, we generate 10 noisy graphs, perform the alignment on each noisy graph, and report the average result. We also test the methods on graphs with ground-truth alignment.

5.1.1 Noise types. In line with previous work [8, 19, 20, 38, 60], we perturb the adjacency matrix of the graph by removing or adding edges. In the literature, several strategies have been proposed, such as removing edges with uniform probability [8, 19], adding and removing edges with uniform probability [29, 35, 37, 38, 61], removing and adding nodes [29], generating noise based on the distance between nodes [27] or sampling from the Poisson distribution [60]. Typically, the authors test their methods using only one strategy. We instead adopt three noise strategies to evaluate the algorithms under different regimes:

- **One-Way**: Remove edges from the target (G_2) graph.
- **Multi-modal**: Remove and add the same number of edges from the target G_2 graph.
- **Two-Ways**: Remove edges from both G_1 and G_2.

5.1.2 Synthetic Graphs [3, 38]. We generate graphs using popular four graph models to evaluate the algorithms under various characteristics of real graphs.

[1] https://github.com/constantinosskitsas/Framework_GraphAlignment

- **Erdös-Rényi (ER)** [14]: random graphs where edges form with a fixed probability. We generate graphs with probability $p = 0.009$
- **Barabasi-Albert (BA)** [2]: scale-free graphs under the preferential attachment model. We generate graphs with initial node degree $m = 5$.
- **Watts-Strogats (WS)** [57]: graphs with small-world properties and high clustering coefficient. We generate graphs with number of neighbors per node $k = 10$ and rewiring probability $p = 0.5$.
- **Newmann-Watts (NW)** [40]: similar to WS but edges are not removed. We generate graphs with number of neighbors per node $k = 7$ and rewiring probability $p = 0.5$
- **Powerlaw cluster (PL)** [21]: a Holme-Kim model similar to BA with a tunable probability of forming triangles. We generate graph with number of random edges $m = 5$ and probability of forming a triangle $p = 0.5$.

5.1.3 Real Graphs. We also use real-world data. Table 2 shows their characteristics. Social and communication graphs are typically power-law, infrastructure networks are grids with power-law distributions; collaboration networks have many triangles; biological and proximity networks are dense.

Dataset	n	m	ℓ	Type
Arenas [30]	1 133	5 451	0	communication
Facebook [32]	4 039	88 234	0	social
CA-AstroPh [32]	17 903	197 031	0	collaboration
inf-euroroad [1, 47]	1 174	1 417	200	infrastructure
inf-power [47, 57]	4 941	6 594	0	infrastructure
fb-Haverford76 [47, 51]	1 446	59 589	0	social
fb-Hamilton46 [47, 51]	2 314	96 394	2	social
fb-Bowdoin47 [47, 52]	2 252	84 387	2	social
fb-Swarthmore42 [47, 52]	1 659	61 050	2	social
soc-hamsterster [47]	2 426	16 630	400	social
bio-celegans [13, 47]	453	2 025	0	biological
ca-GrQc [31, 47]	4 158	14 422	0	collaboration
ca-netscience [39, 47]	379	914	0	collaboration
MultiMagna [56]	1 004	8 323	0	biological
HighSchool [16]	327	5 818	0	proximity
Voles [11]	712	2 391	0	proximity

Table (2) Information about the real graphs, number of nodes n, number of edges m, number of nodes left out of the largest connected component ℓ, and network type.

The last three data sets in Table 2 are *evolving* graphs that naturally provide *ground-truth* alignment. These graphs represent the most challenging scenario, since the real noise distribution is unknown. Only few previous works [20, 38] besides methods designed for biological networks [3, 25, 50] evaluate their performance on ground-truth alignment preferring synthetically generated noise.

5.2 Quality measures

The evaluation of an alignment method requires a measure for alignment quality.

5.2.1 Matched Neighborhood Consistency (MNC) [8, 38]. is the Jaccard similarity of the mapped neighborhood $\widetilde{N}^f_{G_B}(i) = \{j \in V_B : \exists k \in N_{G_A}(i) \text{ s.t. } f(k) = j\}$ of a node $i \in V_A$ and a

node $j \in V_B$.

$$MNC(i,j) = \frac{\left|\widetilde{N}^f_{G_B}(i) \cap N_{G_B}(j)\right|}{\left|\widetilde{N}^f_{G_B}(i) \cup N_{G_B}(j)\right|} \quad (15)$$

The final score is the average MNC among all nodes.

5.2.2 Accuracy or Node Correctness (NC). The most popular measure is Accuracy, that calculates the number of correctly aligned nodes given the true alignment. The accuracy score is the count of corrected alignments normalized by the total number of such alignments. Accuracy assumes that a node in the source graph corresponds to a single node in the target graphs. As such, accuracy disregards multiple alignments or partial alignments.

5.2.3 EC, ICS, S^3. An alternative way to assess the quality of an alignment is to count the number of edges that are correctly aligned. *Edge correctness (EC)* is the percentage of edges in the source graph $G_A = (V_A, E_A)$ correctly aligned in the target graph $G_B = (V_B, E_B)$. Given an alignment f, and the set of correctly aligned edges $f(E_A) = \{(f(i), f(j)) \in E_B : (i,j) \in E_A\}$ EC is defined as

$$\mathrm{EC}(f) = \frac{|f(E_A)|}{|E_A|}$$

EC does not penalize sparse regions in the source graphs, matched to dense region in the target graph. The *Induced Conserved Structure (ICS) [45]* normalizes over the graph in G_B induced by the source graph's nodes $f(V_A) = \{f(v) \in V_B : v \in V_A\}$ correctly aligned

$$ICS(f) = \frac{|f(E_A)|}{|E(G_B[f(V_A)])|}$$

Similarly to EC, ICS does not penalize dense regions in the source graph matched to sparse regions in the target graph.

The *symmetric substructure score (S^3)* [49] corrects the deficiencies of EC and ICS with a normalization over both the source and the target graphs

$$\mathrm{S}^3(f) = \frac{|f(E_A)|}{|E_A| + |E(G_B[f(V_A)])| - |f(E_A)|} \quad (16)$$

	Graph models			*Time (<3h)*		*Mem. (<256Gb)*	
Algorithm	ER	BA/PL	WS/NW	$n>2^{14}$	$\Delta>10^3$	$n>2^{14}$	$\Delta>10^3$
IsoRank	–	–	–	✘	✔	✘	✔
GRAAL	–	–	–	✘	✔	✘	✔
NSD	–	–	–	✔	✔	✔	✔
LREA	–	–	–	✔	✔	✔	✔
REGAL	–	–	–	✔	✘	✔	✘
GWL	–	🏆	–	✘	✘	✘	✘
S-GWL	🏆	🏆	🏆	✘	✘	✘	✘
CONE	🏆	–	🏆	✘	✘	✘	✘
GRASP	–	–	–	✘	✔	✘	✔

Table (3) Summary results vs. graph model (first and second best method marked with 🏆), graph size and density.

6 RESULTS

Here we present the results of our experimental study. Table 3 provides a concise view of our results with different graph models, indicating time and space efficiency in terms of working with graphs of more than 2^{14} nodes and average degree Δ higher than 10^3 in less than 3 hours and within 256Gb.

6.1 Similarity Notion

To address the needs of IsoRank, we devised *our own* weight schema that takes into account node degrees. In particular, the node similarity between nodes u, v is $sim(u,v) = 1 - \frac{|deg(u) - deg(v)|}{\max\{deg(u), deg(v)\}}$, where $deg(u) = |N(u)|$ is the degree of node u. Prior works have used binary weights that had a negative effect the performance of IsoRank. We tried this enhancement on NetAlign, to no avail.

6.2 Assignment Algorithms

We first investigate the performance of assignment algorithms used in the final alignment step. As discussed, some techniques use a heuristic that greedily selects the most attractive one-to-one match; we call this method SortGreedy (SG) [12]; GRAAL performs SG integrally, rendering the adaptation to other methods hard; others use assignment to Nearest Neighbor, that allows for many-to-one matches; lastly, many techniques execute an optimal algorithm for the minimum cost one-to-one linear assignment problem (LAP), i.e., the Hungarian algorithm and its variations for sparse matrices (MWM) [3, 17, 38] or the JV algorithm [20]. These LAP algorithms have higher runtime than heuristics, yet may produce results of better quality. To create a level playing field, we should use a common method for all algorithms in the rest of our study.

We evaluate algorithms with all assignment methods and select the ones that yield highest accuracy on a real dataset, Arenas, and a random graph with power-law degree distribution; we generate noise by permuting the source graph and removing edges with uniform probability {0, 0.01, 0.02, …, 0.05} while keeping the graph connected. Figure 1 shows the results. We try out all assignment methods, yet do not report solutions that produce worse results than the method the authors proposed or need many hours to finish. As MWM produces results similar to those of JV, we show it only with LREA as the author-proposed method.

GWL, **REGAL**, **CONE** and **S-GWL** extract alignments by Nearest-Neighbor (NN). CONE and REGAL do so using a kd-tree to return the top-k most similar matches. While all four methods may produce good results, they return many-to-one assignments. As a matter of principle, we consider one-to-one matchings, hence restrict these four methods to such outputs. They all benefit by applying SG or JV instead of NN. While for CONE, REGAL and S-GWL the improvements due to SG or JV is only between 1% and 4%, for GWL on Arenas SG and JV improve the results from 60% to close to 100%.

GRASP and **LREA** use variants of the Hungarian algorithm in their proposed form. LREA creates a sparse matching matrix called union of matchings and runs the MWM algorithm [3] thereon, a variation of Hungarian that works well with sparse matrices. GRASP implements JV. Our test on GRASP confirms that JV achieves better quality than SG albeit in higher runtime. For LREA, we evaluate MWM against SG and JV. As JV and MWM yield comparable quality results, we opt for JV's multi-threaded implementation.

NSD and **IsoRank** are proposed with SG as alignment method, yet benefit significantly from using JV instead.

In conclusion, JV is our assignment method of choice as it improves alignment accuracy with all algorithms. Linear assignment gracefully leads to one-to-one alignments that maximize the sum of the similarities across nodes. Henceforward, as all algorithms use JV, we report runtime excluding the assignment step. Still, as the density of the similarity matrix affects JV's runtime, more lightweight methods, SG and even NN, are recommendable

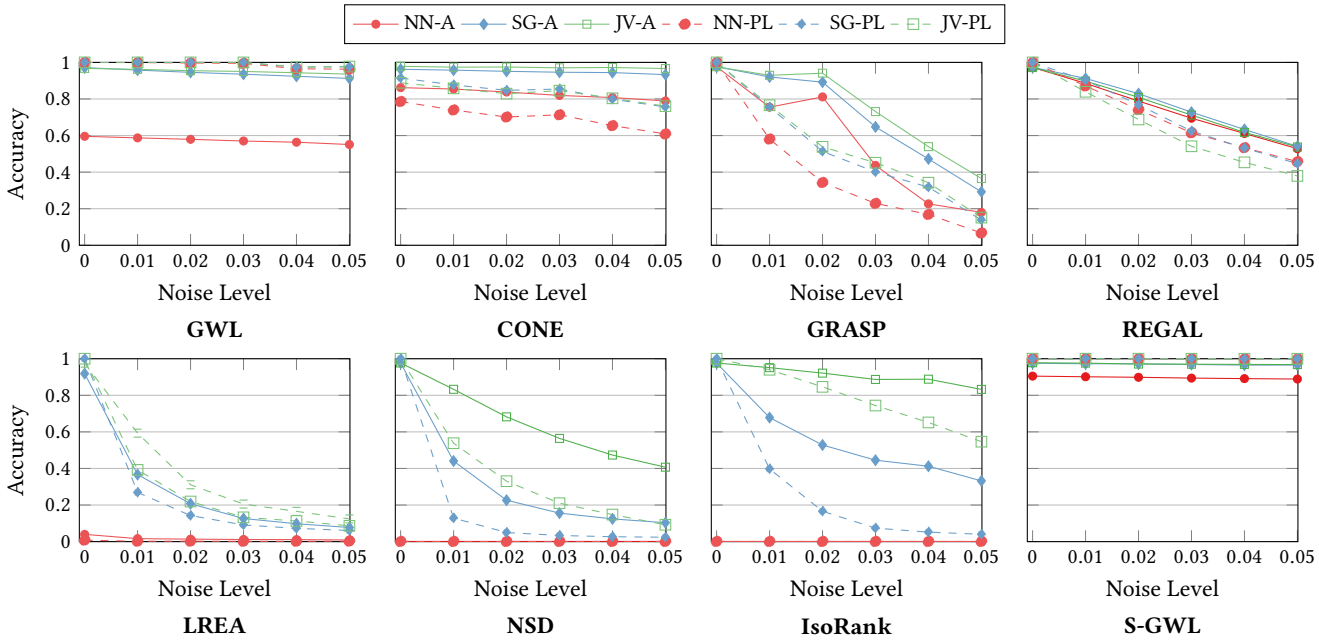

Figure (1) Different assignment methods; solid lines: Arenas real data; dashed lines: Power-Law (PL) synthetic data.

on large graphs. The ability to produce quality assignment with SG or NN is instrumental for scalability purposes. In addition, for algorithms like CONE, REGAL and S-GWL, where JV only brings slight improvements on the cost of a large increase in running time, JV might not be worth the small improvement even for smaller graphs.

6.3 Evaluation of Synthetic Random Graphs

We evaluate the algorithms on five random graph models, to comprehend whether characteristics of the graphs, such as degree distribution or topology, affect the algorithms and how. We choose the models presented in Section 6.3: Erdös-Rényi (ER) (Figure 2), Barabasi-Albert (BA) (Figure 3), Watts-Strogats (WS) (Figure 4), Newmann-Watts (NW) (Figure 5) and Power Law (PL) (Figure 6). For all models, we fix the graph size $n = 1133$ and the degree distribution. In the case of powerlaw graph generators (BA and PL) the degree distribution simulates the one of Facebook, Arenas and Ca-AstroPh. In the case of Gaussian degree distribution as generated by BA, WS, and NW, we mimic the degree distribution of graphs with real alignments, such as HighSchool. To reduce variance across noise levels we generate 10 noisy graphs and report the average.

CONE performs well on all graph models, returning nearly perfect alignments in nearly all models. CONE faces some difficulty with NW graphs, exhibiting some sensitivity to strongly small-world graphs as NW. CONE deficiencies are mostly prominent in PL graphs. This is also confirmed on the Facebook dataset that exhibits a skewed degree distribution (Figure 7). Another interesting insight is the CONE's susceptivity to different types of noise; on multi-modal and two-way noise CONE's quality drops faster than the more common one-way. CONE, that introduced the MNC score, effectively attains better results with the MNC score than with accuracy and S^3.

GWL exhibits good performance only on powerlaw graphs, such as BA (Figure 3) and PL (Figure 6). On other graph types GWL fails to find the correct alignment, scoring close to 0 in all measures even with low noise levels. This behaviour indicates

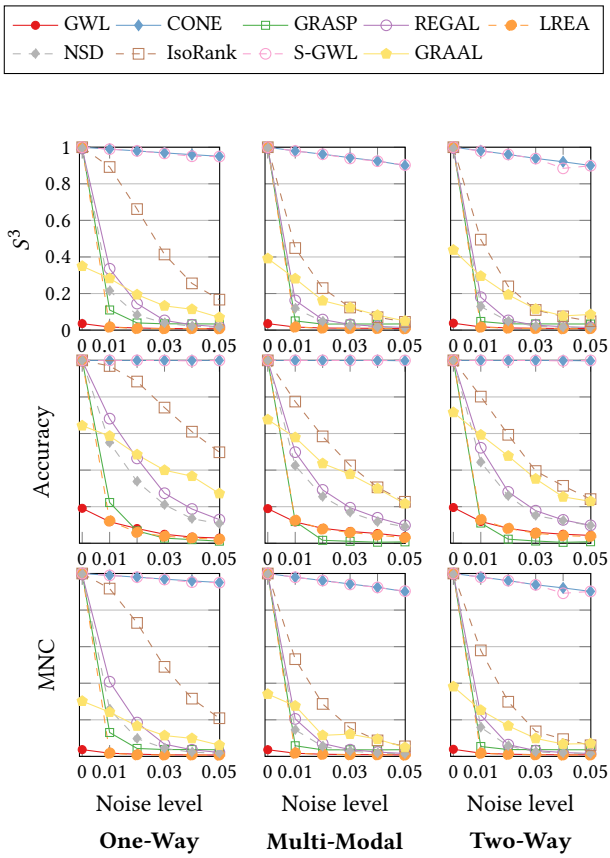

Figure (2) Accuracy, S^3, and MNC for Erdős-Rényi (ER) random graphs; noise up to 5% and different noise types.

that the exact transport objective that GWL optimizes might not be able to discriminate nodes with similar degree distribution. This is also confirmed on BA and PL, in which the S^3 and MNC

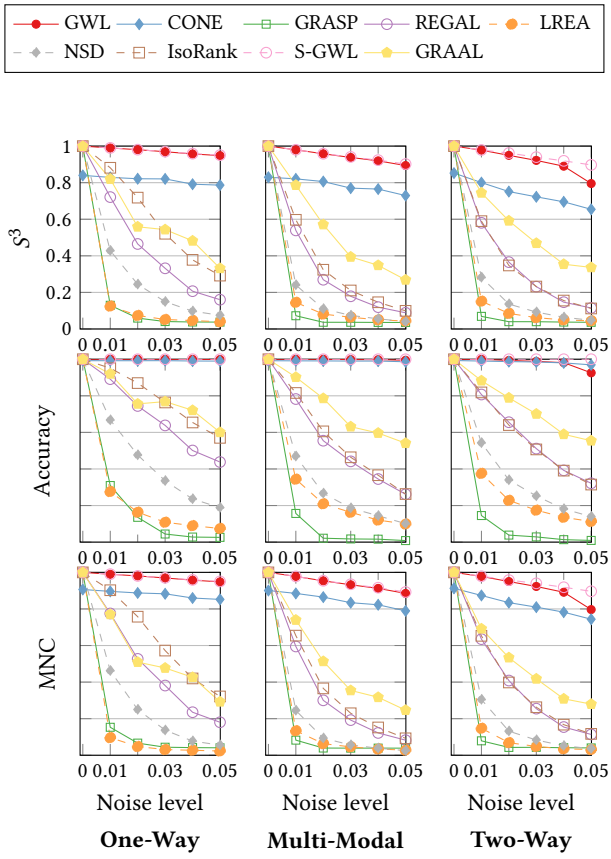

Figure (3) Accuracy, S^3, and MNC for Barabasi-Albert (BA) synthetic graphs; noise up to 5% and different noise types.

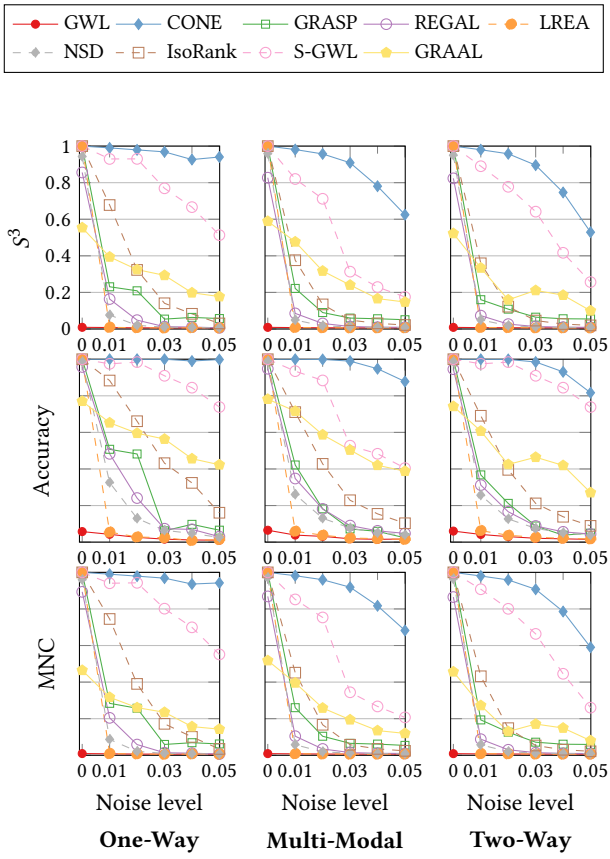

Figure (4) Accuracy, S^3, and MNC for Watts-Strogats (WS) synthetic graphs; noise up to 5% and different noise types.

scores, both based on neighbor matches, are lower than accuracy. On the other hand, GWL is more robust to different noise types than CONE.

IsoRank is among the most competitive algorithms, as opposed to previous comparisons. This is due to an appropriate choice of the weights in Equation (9) that capture the degree distribution of the neighbors of each node. Yet, IsoRank is sensitive to the type and level of noise; for multi-modal and two-way noise accuracy drops by 10 – 30%. We surmise that multi-modal and two-way noise alter the neighbor structure significantly, weakening IsoRank's random walk approach. Nevertheless, IsoRank shows consistent results across graph models. These results render IsoRank a competitive approach compared to more recent methods.

NSD improves upon IsoRank by not requiring prior information to return alignments. The incorporation of prior information boosts quality, but is detrimental to the running time. As expected, NSD exhibits comparable performance to IsoRank, yet its quality drops faster than IsoRank's. NSD inherits IsoRank's sensitivity to noise. Thus, NSD is only preferable to IsoRank when time is critical.

LREA, as expected from the objective, consistently finds the correct alignment on graphs with no noise (i.e., isomorphic). Yet, the performance drops close to 0 on graphs with only 1% noise. This behaviour is consistent across all graph models and potentially reveal the local nature of LREA objective that considers mismatch of single edges rather than structures. This intuition is corroborated by a 40% quality in PL graphs in which node connectivity is more easily determined by the skewed degree distribution.

GRASP, by its spectral nature, performs better on PL graphs with community structure and skewed degree distributions, although it also performs generally well in all graph models and noise types. GRASP exhibits superior performance to REGAL on small-world WS and NWS graphs. Similar to LREA, GRASP almost consistently returns the best alignment on graphs with no noise.

REGAL performs best on powerlaw PL and BA graphs, less so on WS and NWS graphs. REGAL is robust to noise types, although it delivers inferior results than IsoRank by 10%–20%. REGAL is consistent across measures, indicating that the method does not optimize to any of the measures explicitly. All in all, REGAL is a stable algorithm with average performance in most degree distributions.

S-GWL exhibits performance comparable or superior to CONE. Although approximating GWL, S-GWL is competitive in most datasets. This phenomenon in which an approximation outperforms the exact method is not surprising, as approximate methods often remove noise from the data. S-GWL is stable across graph models and noise types. S-GWL is less affected than GWL by the degree distribution and is therefore preferable.

GRAAL has a mediocre position, while it is more robust than others of similar caliber as noise grows.

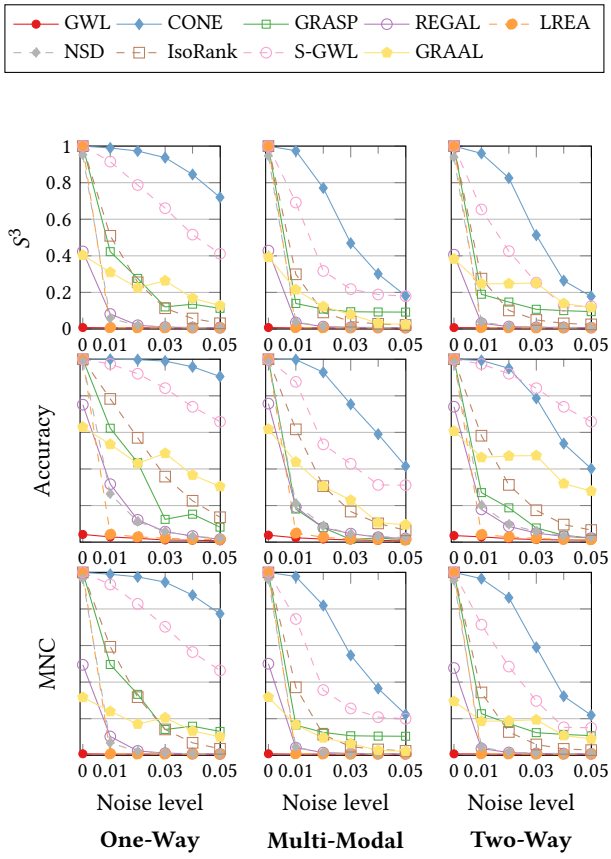

Figure (5) Accuracy, S^3, and MNC for Newman-Watts (NW) synthetic graphs; noise up to 5% and different noise types.

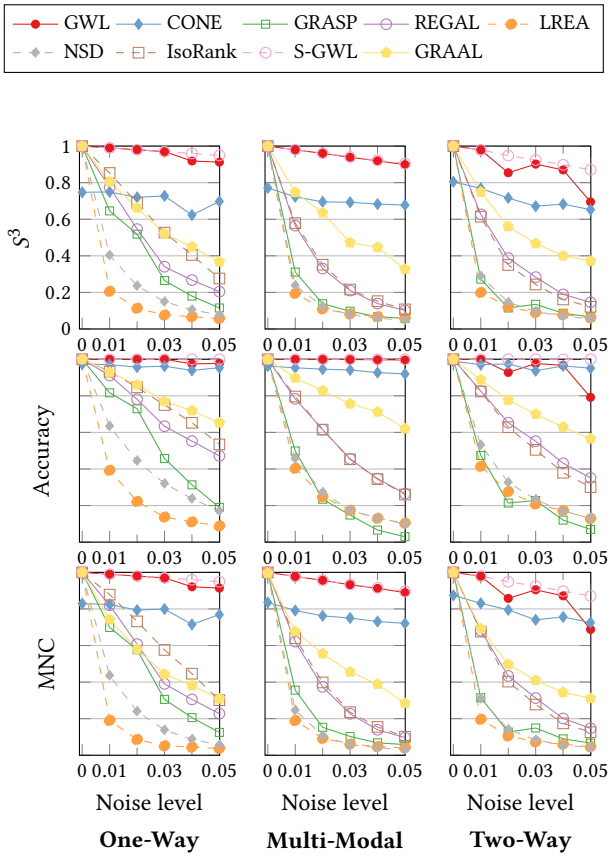

Figure (6) Accuracy, S^3, and MNC for Powerlaw (PL) synthetic graphs; noise up to 5% and different noise types.

Concluding, CONE shows consistently good performance, due to the use of embeddings that capture local and global structures. However, no algorithm is overall the best; CONE is deficient with power-law graphs, while GWL excels only in those. S-GWL does consistently well, yet does not stand out. REGAL, GRASP, GWL, and GRAAL are robust to noise type. IsoRank, with appropriate choice of weights, is a formidable competitor. Although CONE and GWL outperforms competitors, they also present scalability drawbacks, as we see in Section 6.6.

6.4 Evaluation on Graphs with Synthetic Noise

In this section, we evaluate all algorithms on real world graphs with synthetic noise of One-Way, Multi-Modal, and Two-ways type. We report runtime results within 3 hours.

6.4.1 Low Noise. Figure 7 presents results for noise levels in the domain $\{0, 0.01, 0.02, \ldots, 0.05\}$ on three real world graphs: Arenas, Facebook and CA-AstroPh.

GWL performs nearly optimal on Arenas with a slight drop in the accuracy scores with increasing noise levels. For Facebook and CA-AstroPh, GWL exceeds the runtime limit of 3 hours, as its runtime increases significantly for dense or big graph. Thus we do not report accuracy results for these graphs.

CONE also performs nearly optimal on Arenas and is robust to noise. On CA-AstroPh, CONE attains 80% accuracy for One-Way and Two-Way noise; yet, CONE is less effective with Multi-modal noise. We note a similar behaviour on Facebook, where CONE gives ground to S-GWL due to the powerlaw degree distribution.

REGAL follows the same pattern as in Section 5.1.2: uninfluenced by the type of noise, but losing accuracy at higher noise levels. On all three datasets, REGAL achieves comparable accuracy.

GRASP falters on graphs with several connected components, which may arise if the random edge removals disconnect the graph. For instance, on Arenas and CA-AstroPh, sparsity induces disconnected components with noise above 3%. Besides, the Multi-Modal and Two-Way noise affects GRASP's performance due to increased chances to disconnect the graph. On the other hand, GRASP performs well in dense graphs such as Facebook, on which the noise does not generate disconnected components.

LREA performs better on real graphs with synthetic noise than on the synthetic graphs reported in Section 5.1.2, but still not well. Yet it is not too affected by noise type, hence it outperforms GRASP and NSD with Multi-Modal noise on Facebook.

IsoRank is the best algorithm on Facebook and the second-best in the other graphs with One-Way noise. We reiterate that, as prior information, we provide a "naive" similarity score based on degrees. The algorithm is influenced by the type and level of noise, especially Multi-Modal; this is expected, as the more noise is included, the less relevant prior information becomes.

NSD performs poorly with Multi-Modal noise; however, with One-Way and Two-Way noise, it is usually the third-best performer, achieving results comparable to GRASP or REGAL.

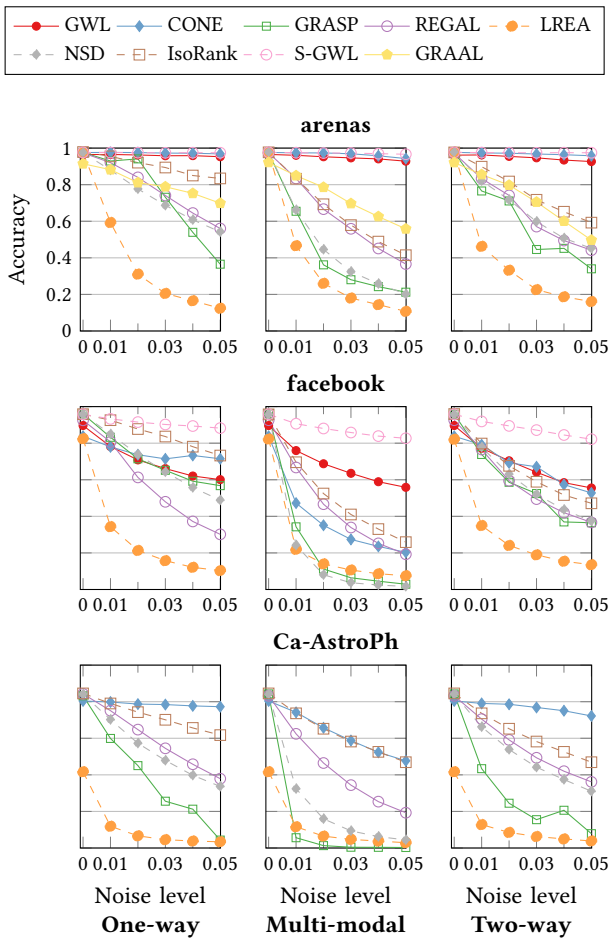

Figure (7) Accuracy on real graphs; noise up to 5%.

S-GWL is one of the best algorithms on Arenas and Facebook. Yet, although it is more scalable than GWL, its complexity is cubic in dense networks, hence it is unsuitable for large graphs. On the other hand, on Arenas S-GWL has performance comparable to CONE.

GRAAL stands again in mediocre positions among the best and worst performers, in those data where it runs within 3 hours; it stands out in terms of its robustness to growing noise, yet never matches the performance of GWL, CONE, and S-GWL.

6.4.2 High Noise. In this experiment we use graphs from the network-repository website [47] shown in Table 2. We run experiments with One-Way noise in the domain value $\{0, 0.05, 0.1, \ldots, 0.25\}$ and report average accuracy of 5 runs. Results are in Figure 8.

CONE is least influenced by the noise level, performing well even with 25% noise. On social networks, it achieves close to optimal accuracy at the highest noise level, except for Hamilton46, where accuracy drops swiftly after 15% noise. For all other graphs, there is slight drop of the accuracy score with increasing noise levels. Infrastructure graphs are most challenging for CONE.

GWL is challenged on infrastructure networks, where its accuracy drops. On other graphs, CONE and GWL exhibit comparable behavior and achieve the highest accuracy, except for the collaboration network CA-GrQc, where GWL performs significantly worse.

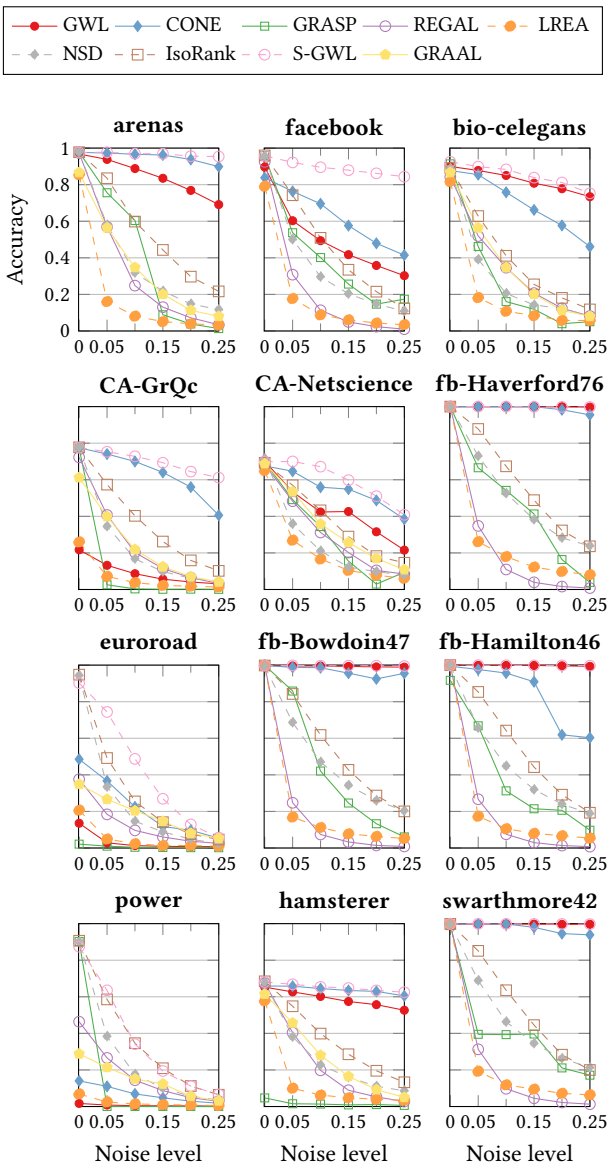

Figure (8) Accuracy on real graphs; one-way noise up to 25%.

REGAL struggles at noise levels larger than 5%. The only datasets where REGAL achieves more than 30% accuracy for noise 10% are Ca-Netscience and Bio-celegans, our two smallest networks.

GRASP is affected by the number of connected components, which, in turn, is influenced by the noise, since deleting edges may disconnect parts of the graph. If a graph consists of more than one connected component, the accuracy score for this graph stays below 15% regardless of noise level. If the noise does not render the graph disconnected, GRASP delivers competitive results, as observed in the previous section. If both source and target graphs are disconnected, but the largest connected component of the graphs is close to the full graph, GRASP can align the networks well. This is the case for Hamilton46, Bowdoin47 and Swarthmore42. Euroroad and hamsterer comprise several connected components in their original form, even without adding noise. In this case, GRASP fails to align the graphs even without noise.

IsoRank succeeds in aligning all the networks and is constantly the third best; its accuracy scores are similar on all the datasets and seem to not be affected by distribution, size, clustering, or density characteristics. It also performs best on infrastructure graphs, where algorithms falter. However, the accuracy of IsoRank drops significantly with increasing noise level. This is due to the fact that with more noise, degree change significantly across graphs and our prior information model becomes less meaningful.

NSD shows a pattern similar to IsoRank, but achieves accuracy between 10% and 20% less than IsoRank for low noise levels. Accuracy drops with noise, but the effect is less pronounced. With noise between 20% and 25%, IsoRank and NSD have similar scores.

S-GWL consistently achieves accuracy close to the best, even with 25% noise. Its main drawback is sensitivity to hyperparameters. We manually set $\beta = 0.025$ on sparse data sets (e.g., inf-power, ca-netscience) and $\beta = 0.1$ on dense dasets (e.g., fb-datasets).

GRAAL performs similarly to NSD in those cases where it runs within the time limit of 3 hours; its previously observed robustness to growing noise levels is attenuated at these high noise levels.

Figure 9 visualizes the results on the NetScience data, as a representative case, juxtaposing accuracy to runtime. CONE and S-GWL stand out on resolving the time-accuracy tradeoff. We obtained similar results with other noise types at high noise levels. We include GRAAL for illustration, albeit not implemented in Python.

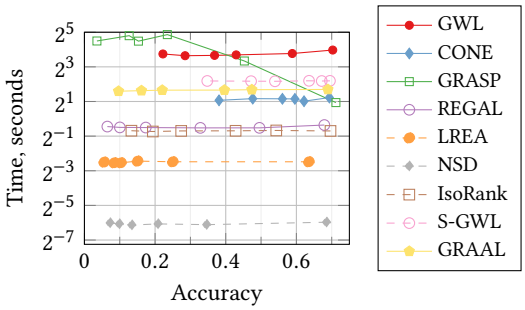

Figure (9) Time vs. accuracy on NetScience; marks show results with One-way noise in $[0.25, 0.2, 0.15, 0.1, 0.05, 0]$**.**

6.5 Evaluation on Graphs with Real Noise

We now evaluate all algorithms on three real-world networks with respect to accuracy, MNC, and S^3. HighSchool and Voles are temporal proximity networks; we match the last version of the graph to versions with 80%, 85%, 90%, and 99% of edges. MultiMagna is a base yeast network with five variants, with edges indicating a possible protein-protein interaction. We match the original to each variant. Figure 10 shows our results. **GWL** and **CONE** do well in all three measures. GWL perfectly aligns Highschool, whereas CONE achieves only 55% accuracy on this network with 80% of edges. Roles are reversed with Voles, where CONE achieves around 80% accuracy on the graph with 80% of edges, while GWL achieves only 32%. On MultiMagna, both algorithms perform similarly; CONE obtains 10% better results on the first graph variants. Accuracy drops consistently across variants to 40% for the last variants. Overall, CONE and GWL perform the best. The next best algorithms are **IsoRank** and **GRAAL**,

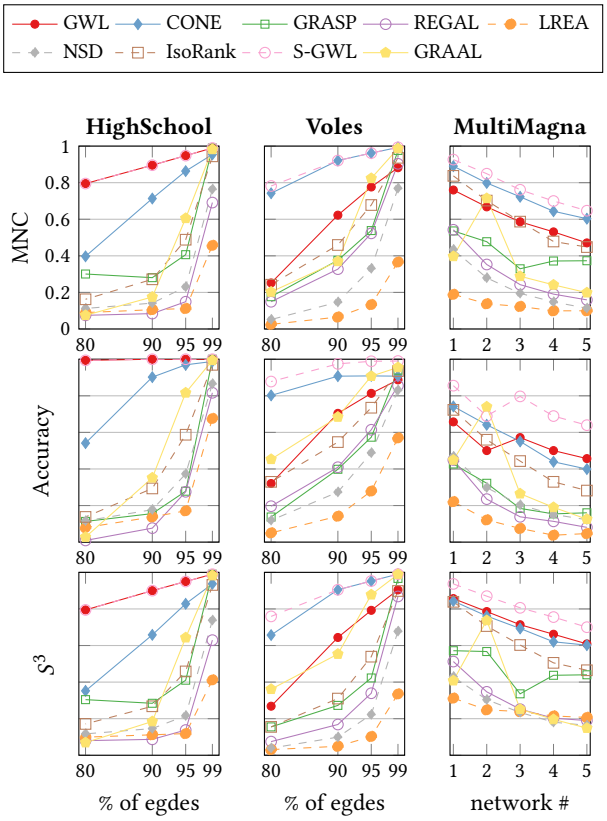

Figure (10) Accuracy, MNC and S^3 results for the real graphs *HighSchool*, *Voles* and *MultiMagna*.

albeit they fare poorly on the HighSchool data. IsoRank's good performance on MultiMagna is expected, as it is designed for protein-protein interaction networks. The remaining algorithms perform well only when the networks to be matched do not differ much, i.e., aligning graphs with 99% of edges on HighSchool and Voles and the first MultiMagna variants; GRASP has a slight advantage, followed by NSD, REGAL and LREA. In terms of MNC and S^3, GRASP approaches IsoRank.

6.6 Scalability

Lastly, we study runtime and memory usage vs. network size and average degree on *configuration model* [41] graphs with normal degree distribution; we exclude the runtime for linear assignment and average results over 5 runs. All algorithms use all 28 cores and a memory of 256Gb, with a maximum runtime allowance of 3 hours. We exclude GRAAL due to its *quintic* ($O(n^5)$) pre-processing time. Figures 11 and 13 show results when tuning the number of nodes from 2^{10} to 2^{16} with average degree 10, while Figures 12 and 14 show results for average degree in $[10, 10^2, 10^3, 10^4]$ with size 2^{14}.

By algorithmic complexity, we expect LREA, NSD, and REGAL to be fastest and IsoRank and GWL slowest. The results in Figures 11–12 confirm our expectation. IsoRank has lower runtime than expected as we let it return a similarity matrix after 100 iterations even if it has not converged. REGAL could not run within the available memory for the highest number of nodes. We let algorithms use sparse array representations as in their implementations. Thus, with CONE using a sparse representation,

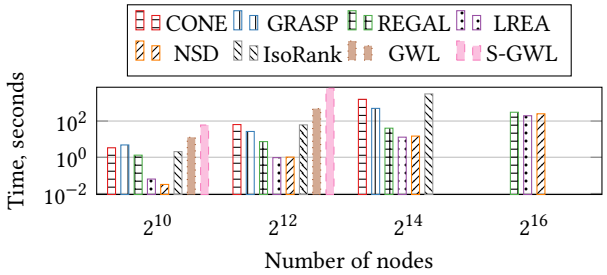

Figure (11) Time vs. # nodes; configuration model graphs.

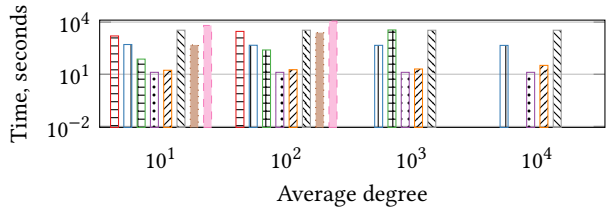

Figure (12) Time vs. avg degree, uniform distr.; 2^{14} nodes.

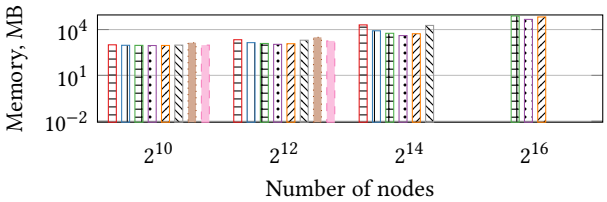

Figure (13) Memory vs. # nodes; configuration model.

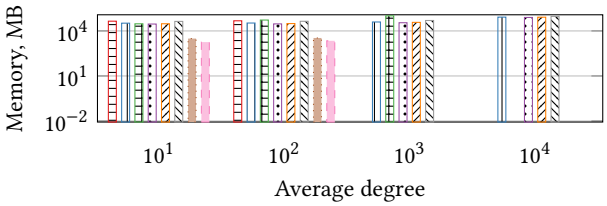

Figure (14) Memory vs. avg degree, uniform distr.; 2^{14} nodes.

even when the number of edges grows, its memory usage does not.

6.7 Varying Density

Our preceding analysis reveals that S-GWL and CONE perform best on small and medium graphs, while REGAL does well on large graphs. In this section, we investigate whether such conclusions hold under varying graph density characteristics. While the graph models we employ do not explicitly allow for controlling density, in NWS graphs, the rewiring probability p implicitly affects the edge density of the sampled graphs for a fixed number of nodes n; besides, the k parameter, i.e., the number of nearest neighbors for each node, implicitly controls the minimum and expected degree.

Figure 15 presents the **impact of density** with NWS graphs of $n = 2000$ nodes. While CONE and S-GWL outperform other algorithms, they face a difficulty when handling sparse graphs with $p = 0.2$. A flatter degree distribution with $k = 100$ accentuates this problem. Besides, GRASP's performance is unstable due to its sensitivity to the disjoint components appearing in the NWS model, reconfirming our observation that, due to its spectral basis, GRASP does not handle disconnected graphs well. We also vary the minimum k while fixing the rewiring probability

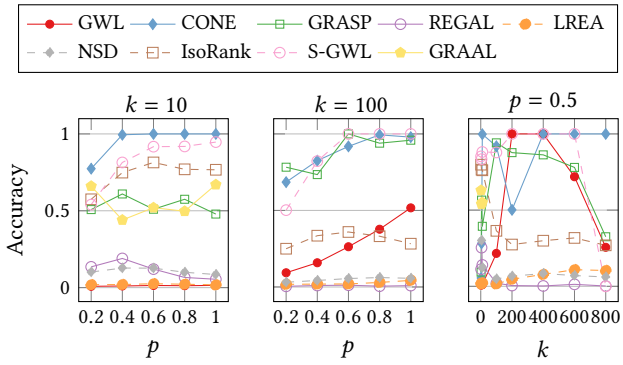

Figure (15) Accuracy for 1% One-way noise on Newman–Watts–Strogatz synthetic graphs with 2000 nodes.

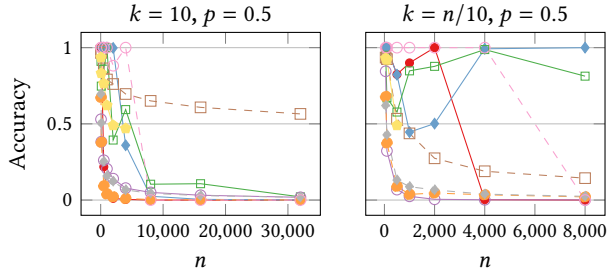

Figure (16) Accuracy for 1% One-way noise on Newman–Watts–Strogatz synthetic graphs of increasing size.

to $p = 0.5$; we see that GWL and, to a lesser extend, S-GWL, cannot align graphs with either too low (0–100) or too high (> 600) average degree. Contrariwise, IsoRank performs comparatively well on low-degree graphs, reconfirming our observations with the Multimagna graph in Figure 10. CONE falters with average degree $k = 200$; this effect seems to arise from a uniform degree distribution rendering nodes indistinguishable to the embedding that forms CONE's backbone.

Figure 16 shows the effect of size on quality. We first examine a constant average degree $k = 10$ and $p = 0.5$ and increasing graph size, hence decreasing density. Remarkably, as the graph becomes progressively sparser, alignment quality drops, except with IsoRank. We conclude that, by virtue of its weighted **M** matrix, IsoRank can easily align small-degree nodes. We also experiment with density fixed to 10%, setting $k = n/10$ and increasing n. The right side of Figure 16 shows our results. GRASP and CONE manage graphs with variable degree, while S-GWL and GWL, as in Figure 15, fail to align networks of either too low or too high average degree.

7 CONCLUSION

We evaluated a gamut of graph alignment algorithms in terms of efficiency and quality. Our study comprised algorithms never compared before, exhaustive parameter tuning, and datasets with real and generated ground-truth alignments; for the latter, we corrupt graph structure with noise of diverse types and at different levels. Our experiments suggest that S-GWL is an algorithm of choice on most counts; yet, if scalability is a concern, REGAL offers a viable alternative. Graph density and degree distribution affect performance. As these are inherent graph properties, we conclude that future graph alignment algorithms should consider these parameters in pre-processing. Our study therefore calls for further efforts for development in graph alignment.

REFERENCES

[1] David A Bader, Henning Meyerhenke, Peter Sanders, and Dorothea Wagner. 2012. Graph Partitioning and Graph Clustering. In *10th DIMACS Implementation Challenge Workshop*.

[2] Albert-László Barabási and Réka Albert. 1999. Emergence of scaling in random networks. *science* 286, 5439 (1999), 509–512.

[3] Mohsen Bayati, David F. Gleich, Amin Saberi, and Ying Wang. 2013. Message-Passing Algorithms for Sparse Network Alignment. *TKDD* 7, 1, Article 3 (2013).

[4] M. Berger. 2007. *A Panoramic View of Riemannian Geometry*. Springer Berlin Heidelberg. https://books.google.dk/books?id=d_SsagQckaQC

[5] Nicole Berline, Ezra Getzler, and Michele Vergne. 2003. *Heat kernels and Dirac operators*. Springer Science & Business Media.

[6] Claude Brezinski and Michela Redivo-Zaglia. 2006. The PageRank Vector: Properties, Computation, Approximation, and Acceleration. *SIAM J. Matrix Anal. Appl.* 28, 2 (June 2006), 551–575. https://doi.org/10.1137/050626612

[7] Sergey Brin and Lawrence Page. 1998. The anatomy of a large-scale hypertextual web search engine. *Computer networks and ISDN systems* 30, 1-7 (1998), 107–117.

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

[9] Joseph Crawford, Yihan Sun, and Tijana Milenković. 2015. Fair evaluation of global network aligners. *Algorithms for Molecular Biology* 10, 1 (2015), 1–17.

[10] Mahashweta Das and Gautam Das. 2015. Structured analytics in social media. *PVLDB* 8, 12 (2015), 2046–2047.

[11] 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).

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

[13] J. Duch and A. Arenas. 2005. Community identification using Extremal Optimization. *Phys Rev. E* 72 (2005), 027104.

[14] P. Erdös and A. Rényi. 1959. On Random Graphs I. *Publicationes Mathematicae Debrecen* 6 (1959), 290.

[15] Soheil Feizi, Gerald Quon, Mariana Recamonde-Mendoza, Muriel Medard, Manolis Kellis, and Ali Jadbabaie. 2019. Spectral alignment of graphs. *IEEE Trans. Netw. Sci. Eng.* 7, 3 (2019), 1182–1197.

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

[17] M. Gerritsen, M. Bayati, D. F. Gleich, A. Saberi, and Y. Wang. 2009. Algorithms for Large, Sparse Network Alignment Problems. In *IEEE ICDM*. 705–710.

[18] Klaus Greff, Aaron Klein, Martin Chovanec, Frank Hutter, and Jürgen Schmidhuber. 2017. The Sacred Infrastructure for Computational Research. In *Proceedings of the 16th Python in Science Conference*. 49–56.

[19] Mark Heimann, Haoming Shen, Tara Safavi, and Danai Koutra. 2018. Regal: Representation learning-based graph alignment. In *CIKM*. 117–126.

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

[21] Petter Holme and Beom Jun Kim. 2002. Growing scale-free networks with tunable clustering. *Physical review E* 65, 2 (2002), 026107.

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

[23] Ehsan Kazemi, S Hamed Hassani, and Matthias Grossglauser. 2015. Growing a graph matching from a handful of seeds. *PVLDB* 8, 10 (2015), 1010–1021.

[24] Brian P Kelley, Bingbing Yuan, Fran Lewitter, Roded Sharan, Brent R Stockwell, and Trey Ideker. 2004. PathBLAST: a tool for alignment of protein interaction networks. *Nucleic acids research* 32, suppl_2 (2004), W83–W88.

[25] Gunnar W Klau. 2009. A new graph-based method for pairwise global network alignment. *BMC bioinformatics* 10, 1 (2009), 1–9.

[26] Jon M. Kleinberg. 1999. Authoritative Sources in a Hyperlinked Environment. *J. ACM* 46, 5 (Sept. 1999), 604–632. https://doi.org/10.1145/324133.324140

[27] Giorgos Kollias, Shahin Mohammadi, and Ananth Grama. 2011. Network similarity decomposition (nsd): A fast and scalable approach to network alignment. *TKDE* 24, 12 (2011), 2232–2243.

[28] Danai Koutra, Hanghang Tong, and David Lubensky. 2013. Big-align: Fast bipartite graph alignment. In *ICDM*. 389–398.

[29] Oleksii Kuchaiev, Tijana Milenković, Vesna Memišević, Wayne Hayes, and Nataša Pržulj. 2010. Topological network alignment uncovers biological function and phylogeny. *Journal of the Royal Society Interface* 7, 50 (2010), 1341–1354.

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

[31] Jure Leskovec, Jon Kleinberg, and Christos Faloutsos. 2007. Graph Evolution: Densification and Shrinking Diameters. *ACM Trans. Knowl. Discov. Data* 1, 1 (March 2007), 2–es. https://doi.org/10.1145/1217299.1217301

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

[33] 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.

[34] Ilya Makarov, Dmitrii Kiselev, Nikita Nikitinsky, and Lovro Subelj. 2021. Survey on graph embeddings and their applications to machine learning problems on graphs. *PeerJ Computer Science* 7 (2021).

[35] Hermina Petric Maretic, Mireille El Gheche, Giovanni Chierchia, Matthias Minder, and Pascal Frossard. 2022. Wasserstein-based graph alignment. *IEEE Transactions on Signal and Information Processing over Networks* (2022).

[36] Marianna Milano, Pietro Hiram Guzzi, Olga Tymofieva, Duan Xu, Christofer Hess, Pierangelo Veltri, and Mario Cannataro. 2017. An extensive assessment of network alignment algorithms for comparison of brain connectomes. *BMC bioinformatics* 18, 6 (2017), 31–45.

[37] Shahin Mohammadi, David F Gleich, Tamara G Kolda, and Ananth Grama. 2016. Triangular alignment (TAME): A tensor-based approach for higher-order network alignment. *TCBB* 14, 6 (2016), 1446–1458.

[38] Huda Nassar, Nate Veldt, Shahin Mohammadi, Ananth Grama, and David F. Gleich. 2018. Low Rank Spectral Network Alignment. In *TheWebConf*. ACM.

[39] Mark EJ Newman. 2006. Finding community structure in networks using the eigenvectors of matrices. *Physical review E* 74, 3 (2006), 036104.

[40] Mark EJ Newman and Duncan J Watts. 1999. Renormalization group analysis of the small-world network model. *Physics Letters A* 263, 4-6 (1999), 341–346.

[41] Mark E. J. Newman. 2003. The Structure and Function of Complex Networks. *SIAM Rev.* 45, 2 (2003), 167–256.

[42] Behnam Neyshabur, Ahmadreza Khadem, Somaye Hashemifar, and Seyed Shahriar Arab. 2013. NETAL: a new graph-based method for global alignment of protein-protein interaction networks. *Bioinformatics* 29, 13 (05 2013), 1654–1662.

[43] Thanh Toan Nguyen, Minh Tam Pham, Thanh Tam Nguyen, Thanh Trung Huynh, Quoc Viet Hung Nguyen, Thanh Tho Quan, et al. 2021. Structural representation learning for network alignment with self-supervised anchor links. *Expert Systems with Applications* 165 (2021), 113857.

[44] Rob Patro and Carl Kingsford. 2012. Global network alignment using multiscale spectral signatures. *Bioinformatics* 28, 23 (2012), 3105–3114.

[45] Rob Patro and Carl Kingsford. 2012. Global network alignment using multiscale spectral signatures. *Bioinformatics* 28, 23 (2012), 3105–3114.

[46] Georgios A Pavlopoulos, Maria Secrier, Charalampos N Moschopoulos, Theodoros G Soldatos, Sophia Kossida, Jan Aerts, Reinhard Schneider, and Pantelis G Bagos. 2011. Using graph theory to analyze biological networks. *BioData mining* 4, 1 (2011), 1–27.

[47] Ryan A. Rossi and Nesreen K. Ahmed. 2015. The Network Data Repository with Interactive Graph Analytics and Visualization. http://networkrepository.com.

[48] Siddhartha Sahu, Amine Mhedhbi, Semih Salihoglu, Jimmy Lin, and M Tamer Özsu. 2017. The ubiquity of large graphs and surprising challenges of graph processing. *PVLDB* 11, 4 (2017), 420–431.

[49] Vikram Saraph and Tijana Milenković. 2014. MAGNA: maximizing accuracy in global network alignment. *Bioinformatics* 30, 20 (2014), 2931–2940.

[50] 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.

[51] Amanda L Traud, Eric D Kelsic, Peter J Mucha, and Mason A Porter. 2011. Comparing Community Structure to Characteristics in Online Collegiate Social Networks. *SIAM Rev.* 53, 3 (2011), 526–543.

[52] Amanda L Traud, Peter J Mucha, and Mason A Porter. 2012. Social structure of Facebook networks. *Phys. A* 391, 16 (Aug 2012), 4165–4180.

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

[54] Anton Tsitsulin, Davide Mottin, Panagiotis Karras, Alexander Bronstein, and Emmanuel Müller. 2018. NetLSD: hearing the shape of a graph. In *KDD*. 2347–2356.

[55] Alexei Vázquez, Alessandro Flammini, Amos Maritan, and Alessandro Vespignani. 2003. Modeling of protein interaction networks. *Complexus* 1, 1 (2003).

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

[57] Duncan J Watts and Steven H Strogatz. 1998. Collective dynamics of 'small-world' networks. *nature* 393, 6684 (1998), 440–442.

[58] Yujia Xie, Xiangfeng Wang, Ruijia Wang, and Hongyuan Zha. 2020. A fast proximal point method for computing exact Wasserstein distance. In *UAI*. PMLR.

[59] Hongteng Xu, Dixin Luo, and Lawrence Carin. 2019. Scalable Gromov-Wasserstein learning for graph partitioning and matching. *NeurIPS* 32 (2019).

[60] Hongteng Xu, Dixin Luo, Hongyuan Zha, and Lawrence Carin Duke. 2019. Gromov-Wasserstein Learning for Graph Matching and Node Embedding. In *ICML (PMLR, Vol. 97)*. 6932–6941.

[61] Si Zhang and Hanghang Tong. 2016. Final: Fast attributed network alignment. In *KDD*. 1345–1354.