ELRUNA: Elimination Rule-based Network Alignment

A Thesis
Presented to
the Graduate School of
Clemson University

In Partial Fulfillment
of the Requirements for the Degree
Master of Science
Computer Science

---

by
Zirou Qiu
May 2020

Accepted by:
Dr. Ilya Safro, Committee Chair
Dr. Brian Dean
Dr. Nina Hubig

Abstract

Networks model a variety of complex phenomena across different domains. In many applications, one of the most essential tasks is to align two or more networks to infer the similarities between cross-network vertices and discover potential node-level correspondence. In this thesis, we propose `ELRUNA` (**El**imination **ru**le-based **n**etwork **a**lignment), a novel network alignment algorithm that relies exclusively on the underlying graph structure. Under the guidance of the elimination rules that we defined, `ELRUNA` computes the similarity between a pair of cross-network vertices iteratively by accumulating the similarities between their selected neighbors. The resulting cross-network similarity matrix is then used to infer a permutation matrix that encodes the final alignment of cross-network vertices. In addition to the novel alignment algorithm, we also improve the performance of *local search*, a commonly used post-processing step for solving the network alignment problem, by introducing a novel selection method `RAWSEM` (**Ra**ndom-**w**alk based **se**lection **m**ethod) based on the propagation of the levels of mismatching (defined in the thesis) of vertices across the networks. The key idea is to pass on the initial levels of mismatching of vertices throughout the entire network in a random-walk fashion. Through extensive numerical experiments on real networks, we demonstrate that `ELRUNA` significantly outperforms the state-of-the-art alignment methods in terms of alignment accuracy under lower or comparable running time. Moreover, `ELRUNA` is robust to network perturbations such that it can maintain a close to optimal objective value under a high level of noise added to the original networks. Finally, the proposed `RAWSEM` can further improve the alignment quality with a less number of iterations compared with the naive local search method.

Acknowledgments

First and foremost, I would like to express my sincerest appreciation to my advisor Dr. Ilya Safro who has guided and encouraged me to continue my research on graph algorithms. No matter the circumstance, Dr. Safro is always supportive and provides me with instructive advice. Under his supervision, I have become a better student, and most importantly, a better researcher. I would like to thank Dr. Brian Dean for joining my committee and for offering one of the most comprehensive and elaborate algorithm classes, from which I learned so much. I would like to thank Dr. Nina Hubig for joining my committee and for assisting me through my Master's journey.

Secondly, I would like to thank Dr. Yuri Alexeev at Argonne National Laboratory for his supervision and support on my research projects; Dr. Christopher Henry and Dr. Cẽsar Cardona at the University of Chicago for their collaborations and generous funding.

Thirdly, I would like to thank my friends Ruslan Shaydulin, Justin Sybrandt, Ehsan Sadrfaridpour, Xiaoyuan Liu, and Ilya Tyagin for their company.

Last but not the least, I would like to thank my mother Feichun who always believes in me no matter what.

Table of Contents

List of Tables

List of Figures

Chapter 1

Background and Motivation

Networks encode rich information about the relationships between entities, including friendships, enmities, research collaborations and biological interactions [16]. The Network alignment problem occurs across various domains. Given two networks, many fundamental data mining tasks involve quantifying their structural similarities and discovering potential correspondences between cross-network vertices [34]. For example, by aligning protein-protein interaction networks, we can discover functionally conserved components and identify proteins that play similar roles in networked bio-systems [21]. In the context of marketing, it is often useful for companies to link similar users across different networks in order to recommend products to potential customers [34]. Furthermore, the network alignment problem also exists in fields such as computer vision [2], chemistry [14], social network mining [34], and economy [35].

In general, network alignment aims to map [1] vertices in one network to another such that some cost function is optimized and pairs of mapped cross-network vertices are similar [34]. While the exact definitions of similarities are problem dependent, they often reveal some resemblance between structures of two networks an/or additional domain information such as similarities between DNA sequences [21]. Formally, we define the network alignment problem as follows:

[1] We use the terms *map* and *align* interchangeably through out the thesis

Problem 1 (**Network Alignment Problem**). *Given two networks with underlying undirected, unweighted graphs* $G_1 = (V_1, E_1)$ *and* $G_2 = (V_2, E_2)$ *with* $|V_1| = |V_2|$ *(this constraint is trivially satisfied by adding dummy 0-degree nodes to the smaller network)*[2]. *Let* $\boldsymbol{A}$ *and* $\boldsymbol{B}$ *be the adjacency matrices of* G_1 *and* G_2*, respectively. The goal is to find a permutation matrix* $\boldsymbol{P}$ *that minimizes the cost function:*

$$\min_{\boldsymbol{P}} \quad -trace(\boldsymbol{P}^T \boldsymbol{A} \boldsymbol{P} \boldsymbol{B}^T), \tag{1.1}$$

where $\boldsymbol{P}$ *encodes the bijective mappings between* V_1 *and* V_2 *for which* $\boldsymbol{P}_{i,u} = 1$ *if* $i \in V_1$ *is aligned with* $u \in V_2$*, and* $\boldsymbol{P}_{i,u} = 0$ *otherwise.*

An equivalent problem is to maximize the number of ***conserved*** *edges in* G_1*, for which an edge* $(i, j) \in E_1$ *is conserved if* $\boldsymbol{P}_{i,u} = 1$, $\boldsymbol{P}_{j,v} = 1$ *and* $(u, v) \in E_2$.

The above problem is a *quadratic assignment problem* (QAP) which is known to be NP-hard [32]. Moreover, network alignment problem can also be considered as an instance of *subgraph isomorphism problem* [19]. Due to its hardness, many heuristics have been developed to solve the problem by relaxing the integrality constraints. Typically, the existing methods first compute similarity between every pair of cross-network vertices by iteratively accumulating similarities between pairs of cross-network neighbors, then infer the alignment of cross-network nodes by solving variants of maximum weight matching problem [8].

Many existing approaches provide good insights about the potential correspondence between cross-network vertices, however, they still exhibit several limitations. First, under the setting of some previous methods [34, 35, 15, 2, 29, 18, 9, 10, 7, 20], computing similarity between $i \in V_1$ and $u \in V_2$ is a process of accumulating the similarities between **all** pairs of their cross-network neighbors. In other words, while computing the similarity between i and u, each of their neighbors contributes **multiple times**. This might lead to an unwanted case where i has a high similarity score with u simply because u is a high-degree node so

[2]Note that the requirement $|V_1| = |V_2|$ is introduced only to make $\mathbf{P}$ square for simple computation of the objective as 0-degree dummy nodes do not contribute to it. In later discussion and for the implementation of the algorithm, we do not require $|V_1| = |V_2|$.

they have many pairs of cross-network neighbors that can contribute similarities to (i, u). In addition, this setting also makes it difficult to effectively penalize the degree difference between i and u, and after normalization, the resulted similarity is diluted because of the inclusion of many "noisy" similarities.

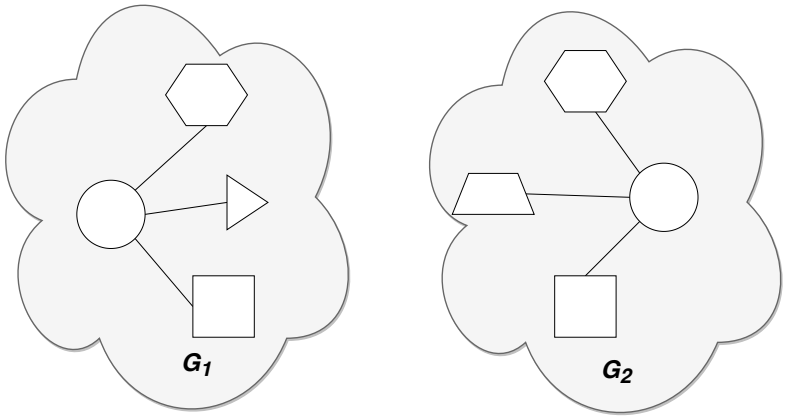

Figure 1.1: An example of the dilution of result

To illustrate the dilution by noisy similarities, consider two **unattributed** graphs shown in Figure 1.1. Let the shape of a node denote its ground-truth identity. For example, the circle in G_1 should have a higher similarity with the other circle in G_2 than nodes with other shapes. When the similarity between two circle nodes gets updated, some exsiting methods not only accumulate the similarities between $(hexagon, hexagon)$, $(square, square)$ and $(triangle, trapezoid)$, but also the similarities between $(hexagon, square)$, $(hexagon, trapezoid)$, $(square, hexagon)$ and so on. However, one could argue that the inclusion of the similarity between dissimilar vertices will dilute the result.

Another limitation is that many existing network alignment algorithms rely on non-network information (prior-similarity matrices) to generate high-quality alignments [2, 9, 7, 14, 34, 33, 23, 5]. However, these methods are limited when no such information is available.

Our contribution: We address the network alignment problem by focusing on overcoming the above limitations. The main contributions of this work are as follows:

1. `ELRUNA`**: Network Alignment Algorithm**. We propose a novel network alignment

algorithm ELRUNA which identifies *globally most similar* pairs of vertices based on the growing *contribution threshold* (both defined in the later section). Such a threshold is used to determine the pairs of cross-network vertices that can contribute similarities while **eliminating others**. To the best of our knowledge, ELRUNA is the first network alignment algorithm that introduces the elimination rule into the process of accumulating similarities. Another novelty is that ELRUNA solves network alignment problem by iteratively solving smaller subproblems at the neighborhood scale. Extensive experiment results show that the proposed ELRUNA significantly outperforms the state-of-the-art counterparts under lower or comparable running time. Moreover, ELRUNA can maintain a close to optimal objective value under a high level of noise added to the original networks. What makes ELRUNA even more competitive is that it discovers high-quality alignment results solely relying on the topology of the networks (without the help of non-network infromation).

2. RAWSEM**: Selection Method for Local Search.** We introduce a novel selection method RAWSEM for local search procedure which narrows the search space by locating mismatched vertices. The proposed method first quantifies the initial amount of mismatching of each vertex and then propagates the values of mismatching throughout the network in a PageRank [22] fashion. After convergence, vertices with high level of mismatching are selected into the search space. To the best of our knowledge, RAWSEM is the first random-walk based **selection method** for the local search scheme in solving the network alignment problem.

3. **Evaluations**. We conducted extensive experiments to analyze the effectiveness and efficiency of ELRUNA. We compare ELRUNA againt 8 state-of-the-art baselines on 21 networks. Our experiments cover three real-world scenarios: (1) *self-alignment without and under noise* (2) *alignment between homogeneous networks*; and (3) *alignment between heterogeneous networks*. The results show that the proposed ELRUNA (before applying local search) already significantly outperforms all the baseline methods. At

the same time, the proposed RAWSEM can further improve the objective values to an optimum with a drastically decreased number of iterations compared with the naive local search.

The rest of the thesis is organized as follows. Chapter 2 discusses related works. Chapter 3 presents the proposed network alignment algorithm ELRUNA. Chapter 4 introduces the selection method RAWSEM. Chapter 5 presents the experimental results. The conclusion and future works are given in chapter 6. Additional experimental results can be found in the appendix section.

Chapter 2

Related Work

Extensive research has been conducted in solving the network alignment problem. In most methods, the underlying intuition is that *two cross-network vertices are similar if their cross-network neighbors are similar*. The limitations of the existing works are discussed in the introduction section.

`IsoRank` [29] is an alignment algorithm which is equivalent to PageRank on the Kronecker product of two networks. `IsoRankN` [18] extends `IsoRank` by applying spectral graph partitioning to align multiple networks simultaneously. Koutra et al. [15] formulate the bipartite network alignment problem and propose an iterative improvement algorithm to find the optimum. The `Klau` method formulates the problem using the maximum weight trace and suggests a Lagrangian relaxation approach [14]. Wang et al. introduce `NetAlign` which treats the network alignment problem as an integer quadratic program and solve it using the belief propagation. `EigenAlign` [7] formulates the network work alignment problem with respect to not only the number of conserved edges, but also non-conserved edges and neutral edges. The authors then solve it as an eigenvector computation problem which finds the eigenvector with the largest eigenvalue. Finally, `C-GRAAL` [20] is a member of `GRAAL` family which iteratively computes the cross-network similarities based on the *combined neighborhood density* of each node.

Xu et al. [19] solve the network alignment problem by projecting the problem into

the domain of computational geometry while preserving the topology of the graphs. Then, they use the rigid transformations approach to compute the similarity scores. `REGAL` [12] tackles the problem from a node representation learning prospective. By leveraging the low-rank matrix approximation method, they extract the node embeddings, then construct the alignment of vertices based on the similarity between embeddings of cross-network vertices. In addition to finding the one-to-one mapping of vertices, `REGAL` can also identify the top-α potential mappings for each vertex. `HashAlign` [11] solves the multiple network alignment problem also based on node representation learning for which each node-feature vector encodes topological features and attributed features.

Saraph and Milenkovic introduce `MAGANA` which uses genetic algorithm based local search to solve the alignment problem. Later, they introduce `MAGANA++` which extends `MAGANA` with parallelization. Finally, Milenkovic et al. address the dynamic network alignment problem where the structures of the networks evolve over time. They propose `DynaMAGNA++` that conserves dynamic edges and nodes [30]. There exist other local search approaches for this problem including such stochastic versions as simulated annealing which are very slow.

Zhang and Tong [34] tackle the attributed network alignment problem for which vertices have different labels. They drop the topological consistency assumption and solve the problem by using attributes as alignment guidance. Du et al. [5] also address the attributed network alignment problem where the underlying graphs is evolving (revising). They formulate the problem as a Sylvester equation and solve it in a incremental fashion with respect to the updates of networks. In another work by Zhang et al. [35], they solve the multilevel network alignment problems based on the coarsening and un-coarsening scheme. By coarsening the network into multiple levels, they not only discover the node correspondence of the original network but also cluster-level correspondence with different granulairties. Such coarsening-uncoarsening methods have been very successful in solving various cut-based optimization problems on graphs [24, 26] but, to the best of our knowledge, are used for the first time for network alignment.

Hashemifar and Xu's [10] method involves computing topological importance for each node. A pair of cross network vertices have similar scores if they play similar roles, such as hubs or nodes with high betweenness centralties, in the networks. `NETAL` [21] introduces the concept of interaction scores between each pair of cross-network vertices which are estimations of the number of conserved edges. `ModuleAlign` [9] combines the topological information with non-network information such as protein sequence for each vertex and produces an alignment that resemble both topological similarities and sequence similarities. `GHOST` [23] is another biological network alignment algorithm based on the graph spectrum. `GHOST` determines the similarities between cross-network vertices based on the similarities between the topological signatures of each vertex, and such signature is obtained by computing the spectrum of the k-egocentric subgraph of each vertex.

Chapter 3

ELRUNA : Elimination Rule-based Network Alignment

In this section, we introduce the network alignment algorithm ELRUNA. We first provide notation used throughout this thesis in Table 3.1. Then we define three rules that serve as a guidance for our algorithm. Finally, we introduce the pseudocode of ELRUNA and analyze its running time.

Given two undirected networks with underlying graphs $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$. Let $|V_1| = n_1$, $|V_2| = n_2$, $|E_1| = m_1$ and $|E_2| = m_2$. Without loss of generality, assume $n_1 \leq n_2$. Nodes in each network are labeled with consecutive integers starting from 1.

Throughout the thesis, we use bold uppercase letters to represent matrices and bold lowercase letters to represent vectors. We use subscript over a node to refer the network it belongs to, for example, $i_{(1)} \in V_1$. We use superscript over vectors/matrices to denote the number of iterations. Let $N(i_{(1)})$ denote the set of neighbors of vertex $i_{(1)}$. Let $\mathbf{S}$ be the $n_1 \times n_2$ cross-network similarity matrix where $\mathbf{S}_{i,u}$ encodes the similarity score between $i_{(1)}$ and $u_{(2)}$. Note that $i_{(1)}$ and $u_{(2)}$ do not carry superscripts for matrix / vector indexing. Let $f : V_1 \to V_2$ denote the injective alignment function for which $f(i_{(1)}) = u_{(2)}$ if $\mathbf{P}_{i,u} = 1$ ($\mathbf{P}$ is defined in Eq. (1.1)). Function f is injective because n_1 could be less than n_2 (f is bijective

when $n_1 = n_2$). Let t_{max} denote the maximum number of iterations of our algorithm. We always set t_{max} equal to the larger diameter of the two networks.

Table 3.1: Notation

Symbol	Definition
$G_1 = (V_1, E_1), G_2 = (V_2, E_2)$	the two networks
$\mathbf{A}, \mathbf{B}$	the adjacency matrices of G_1 and G_2
n_1, n_2	number of nodes in G_1 and G_2
m_1, m_2	number of edge in G_1 and G_2
$i_{(1)}$, $j_{(1)}$	example nodes in G_1
$u_{(2)}$, $v_{(2)}$	example nodes in G_2
$N(i)$	the set of neighbors of node i
$\mathbf{S}$	the $n_1 \times n_2$ cross-network similarity matrix
$\mathbf{P}$	the $n_1 \times n_2$ alignment matrix
$f : V_1 \to V_2$	the injective alignment function.
t_{max}	the maximum number of iterations

The proposed `ELRUNA` relaxes the combinatorial constraints of $\mathbf{P}$ defined in Problem 1.1, that is, it iteratively computes a similarity matrix $\mathbf{S}$ instead of directly finding a permutation matrix $\mathbf{P}$. In general, our proposed algorithm `ELRUNA` is a two-step procedure:

(1) **Similarity computation:** Based on the elimination rules, `ELRUNA` iteratively updates the *cross-network similarity matrix* $\mathbf{S}$ which encodes similarities between cross-network vertices.

(2) **Alignment:** Based on the converged $\mathbf{S}$, `ELRUNA` computes the 0-1 alignment matrix $\mathbf{P}$ which encodes the final alignment of corss-network vertices.

Note that the main differences between most alignment algorithms is how they compute the similarities between vertices. The alignment task does not require a complex alignment method (the second step) if its similarity computation step produces high-quality

alignment matrices. Therefore, the main focus of ELRUNA is to compute high-quality alignment matrices.

3.1 Step 1: Similarity Computation

We introduce three rules which serve as the guidance of the proposed algorithm. We then present the pseudocode for computing the similarity matrix $\mathbf{S}$. Overall, the proposed ELRUNA computes the similarities between cross-network vertices by updating $\mathbf{S}$ iteratively.

We first provide an intuitive concept of what it means for a vertex to ***contribute*** to the similarity computation process: Given a pair of cross-network vertices $(i_{(1)}, u_{(2)})$, let $j_{(1)}$ be a neighbor of $i_{(1)}$. At the kth iteration of the algorithm, $j_{(1)}$ is said to ***contribute*** to the computation of $\mathbf{S}_{i,u}^{(k)}$ if we accumulate (will be defined later) the similarity between $j_{(1)}$ and a neighbor $v_{(2)}$ of $u_{(2)}$ into $\mathbf{S}_{i,u}^{(k)}$. By the same token, the pair $(j_{(1)}, u_{(2)})$ is also said to ***contribute*** to the computation of $\mathbf{S}_{i,u}^{(k)}$.

We now provide three essential definitions that are the backbone of ELRUNA:

Definition 1 (**Conserved Vertices and Edges**). *Given a vertex $i_{(1)}$ and its aligned vertex $u_{(2)} = f(i_{(1)})$. Let $j_{(1)} \in N(i_{(1)})$ be a neighbor of $i_{(1)}$, and $v_{(2)} = f(j_{(1)})$ be the the aligned vertex of $j_{(1)}$. Node $j_{(1)}$ is a **conserved** neighbor of $i_{(1)}$ if $v_{(2)} \in N(u_{(2)})$. Under this scenario, $v_{(2)}$ is also a conserved neighbor of $u_{(2)}$. An edge is **conserved** if its incident verteices are conserved neighbors of each other.*

Definition 2 (**Best Matching**). *A vertex $u_{(2)}$ is the **best matching** of a vertex $i_{(1)}$ if aligning $i_{(1)}$ to $u_{(2)}$ maximizes the number of conserved neighbors of $i_{(1)}$ in comparison with aligning $i_{(1)}$ to other nodes in V_2. The best matching of $u_{(2)}$ is defined in the same fashion.*

Definition 3 (**Globally Most Similar**). *At the kth iteration, a vertex $u_{(2)}$ is **globally most similar** to a vertex $i_{(1)}$ if $\mathbf{S}_{iu}^{(k)} \geq \mathbf{S}_{iv}^{(k)}$, $\forall v_{(2)} \in V_2$. Likewise, $i_{(1)}$ is globally most similar to $u_{(2)}$ if $\mathbf{S}_{iu}^{(k)} \geq \mathbf{S}_{ju}^{(k)}$, $\forall j_{(1)} \in V_1$.*

Note that the definitions of *best matching* and *globally most similar* vertices are

one-way. In other words, $i_{(1)}$ being globally most similar to $u_{(2)}$ it does not imply that $u_{(2)}$ is also globally most similar to $i_{(1)}$.

Theorem 1 below shows that the objective defined at Equation (1.1) is optimized when all nodes are aligned to their best matchings (if possible).

Theorem 1. *Given an alignment matrix* $\mathbf{P}$ *for which all nodes are aligned to their best matchings,* $\mathbf{P}$ *is the optimal solution of equation (1.1).*

Proof. Suppose for the sake of contradiction, there exists a better alignment matrix $\bar{\mathbf{P}} \neq \mathbf{P}$ such that

$$trace(\bar{\mathbf{P}}^T \mathbf{A} \bar{\mathbf{P}} \mathbf{B}^T) > trace(\mathbf{P}^T \mathbf{A} \mathbf{P} \mathbf{B}^T). \tag{3.1}$$

Let $\mathbf{B}'$ and $\mathbf{B}''$ denote $\bar{\mathbf{P}}^T \mathbf{A} \bar{\mathbf{P}}$ and $\mathbf{P}^T \mathbf{A} \mathbf{P}$, respectively. Inequality (3.1) implies that

$$\mathbf{B}'_{i,*} \mathbf{B}^T_{*,i} > \mathbf{B}''_{i,*} \mathbf{B}^T_{*,i}, \ \exists \, i \in V_1 \tag{3.2}$$

where $\mathbf{B}^T_{i,*}$ and $\mathbf{B}^T_{*,i}$ denotes the ith row and column of $\mathbf{B}^T$, respectively. However, the inequality (3.2) implies that in $\mathbf{P}$, there exists a vertex $i_{(1)}$ that is not aligned with its best matching which is a contradiction. □

By Theorem 1, to optimize the objective value, nodes are desired to be aligned with their best matchings. **Algorithm `ELRUNA` heuristically attempts to ensure that the node which is globally most similar to $i_{(1)}$, provided by the similarity matrix S, corresponds to the best matching of $i_{(1)}$.** Then the alignment process is simply to map each node $i_{(1)}$ in G_1 to its globally most similar vertex $u_{(2)}$ in G_2.

3.1.1 Rule 1 - level-one elimination

While computing similarity between a pair of cross-network vertices, under the setting of `ELRUNA`, a neighbor cannot contribute twice. Given a pair of vertices $(i_{(1)}, u_{(2)})$, we consider computing their similarity as a process of aligning their neighbors. In other words, **a pair of cross-network neighbors $(j_{(1)}, v_{(2)})$ can contribute its similarity**

to $\mathbf{S}_{i,u}$ only if $j_{(1)}$ is qualified to be aligned with $v_{(2)}$ (in the section below, we explain the way to determine whether a pair of neighbors are qualified to be aligned or not). As a result, $i_{(1)}$ and $u_{(2)}$ have a higher similarity if they have more neighbors that can be aligned, which also optimize the objective defined in Equation (1.1). The injective nature of alignments leads to our first rule:

> **Rule 1.** At the kth iteration of the algorithm, given a pair of cross-network vertices $(i_{(1)}, u_{(2)})$, a neighbor $j_{(1)}$ of $i_{(1)}$ can contribute its similarity (with a neighbor of $u_{(2)}$) to $\mathbf{S}_{i,u}^{(k)}$ **at most once**. Similarly, a neighbor $v_{(2)}$ of $u_{(2)}$ can contribute its similarity (with a neighbor of $i_{(1)}$) to $\mathbf{S}_{i,u}^{(k)}$ **at most once**.

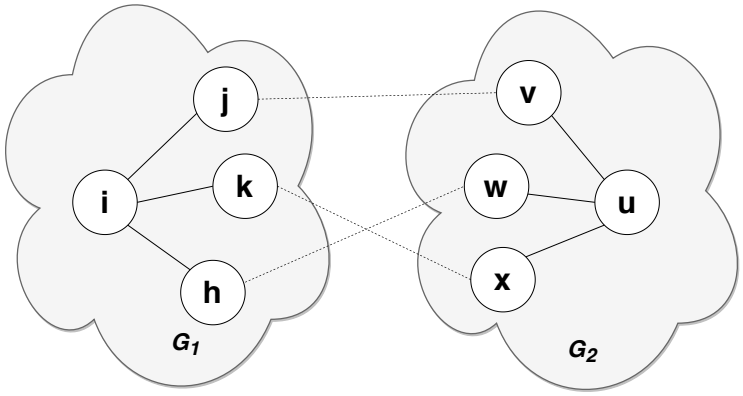

Figure 3.1: Example of rule 1.

Rule 1 allows each neighbor to contribute at most once while eliminating the contribution of similarities from all other pairs. Figure (3.1) illustrates an example graph under rule 1 where dashed lines indicate the pairs of cross-network neighbors that contribute to the computation of $\mathbf{S}_{i,u}^{(k)}$. For example, if the similarity between $j_{(1)}$ and $v_{(2)}$ is accumulated into $\mathbf{S}_{i,u}^{(k)}$, then the similarity between $(j_{(1)}, w_{(2)})$, $(j_{(1)}, x_{(2)})$, $(k_{(1)}, v_{(2)})$, and $(h_{(1)}, v_{(2)})$ can no longer contribute to $\mathbf{S}_{i,u}^{(k)}$.

Formally, rule 1 decreases the number of pairs of cross-network contributing neighbors from $|N(i_{(1)})| \times |N(u_{(2)})|$ to at most $\min\{|N(i_{(1)})|, |N(u_{(2)})|\}$. This setting provides an

effective way to penalize the degree differences as shown in the later section. Additionally, it reduces the amount of "noisy" similarities included during the iteration process.

3.1.2 Rule 2 - level-two elimination

Prior to defining rule 2, we assume that rule 1 is satisfied. How to determine whether a pair of cross-network neighbors are qualified to contribute? Ideally, **a pair of cross-network neighbors can be aligned** (and therefore, qualified to contribute, as stated in rule 1) **if at least one of them is globally most similar to the other**. Because of the iterative nature of `ELRUNA`, however, during the first several iterations of the algorithm, the computed similarities are less revealing of the true similarities between vertices. In other words, we are less certain about whether the node that is globally most similar to $j_{(1)}$ is indeed its best matching. Therefore, for each iteration, only allowing globally most similar pairs of neighbors to contribute while discarding others might fail to accumulate valuable information.

We want to relax the above constraint. We observe that as we proceed with more iterations, the reliability of similarities increases. To model this increase of confidence, we define *growing* thresholds such that under rule 1, **pairs of neighbors whose similarities are greater than their thresholds are allowed to contribute**. Such thresholds are low at the first iteration and grow gradually as we proceed with more iterations.

We define vectors **b1** and **b2** for two networks respectively such that

$$\mathbf{b1}_i^{(k)} = \max_{u \in V_2} \mathbf{S}_{i,u}^{(k)} \text{ and } \mathbf{b2}_u^{(k)} = \max_{i \in V_1} \mathbf{S}_{i,u}^{(k)}.$$

Informally, $\mathbf{b1}_i^{(k)}$ is the similarity between $i_{(1)}$ and its globally most similar vertex at the kth iterations. Additionally, we define ***contribution-threshold vectors***, **c1** and **c2**, for two networks. Given a pair of vertices $(i_{(1)}, u_{(2)})$, a pair of their cross-network neighbors $(j_{(1)}, v_{(2)})$ can only contribute similarity to $\mathbf{S}_{i,u}^{(k+1)}$ if $\mathbf{S}_{j,v}^{(k)} \geq \min\{\mathbf{c1}_j^{(k)}, \mathbf{c2}_v^{(k)}\}$. At the same time, such thresholds grow as the algorithm proceeds with more iterations.

Computing the similarity between $(i_{(1)}, u_{(2)})$ iteratively can be seen as a process of gathering information (regarding the similarity) from other nodes in a breadth-first search (BFS) manner such that $\mathbf{S}_{i,u}^{(k)}$ is computed based on similarities between cross-network nodes that are within distance k away from $i_{(1)}$ and $u_{(u)}$. A node $j_{(1)}$ is said to be *visited* by $i_{(1)}$ at the the $k$the iteration if $j_{(1)}$ is within distance k away from $i_{(1)}$. **We use the fraction of visited nodes after each iteration as a measure to model the increase of the contribution threshold**. We now define the contribution threshold formally:

Definition 4 (**Contribution Threshold**). *Given G_1 with size n_1, let $\mathbf{T1}$ be the $n_1 \times t_{max}$ matrix for which $\mathbf{T1}_{i,k}$ is the fraction of nodes that $i_{(1)}$ has visited after the kth iteration. Then the **contribution threshold** of $i_{(1)}$, denoted by $\mathbf{c1}_i$, after the kth iteration is defined as*

$$\mathbf{c1}_i^{(k)} = \mathbf{b1}_i^{(k)} \times \mathbf{T1}_{i,k} \tag{3.3}$$

$\mathbf{T1}_{i,0} = 1/n_1$ for all $i_{(1)} \in V_1$. As k approaches t_{max}, $\mathbf{T1}_{i,k}$ approaches 1 and $\mathbf{c}_i^{(k)}$ approaches $\mathbf{b}_i^{(k)}$. $\mathbf{c2}$ and $\mathbf{T2}$ are defined for G_2 in the same fashion. The pseudocode for computing $\mathbf{T1}$ and $\mathbf{T2}$ is shown in Algorithm (1).

Finally, let $(j_{(1)}, v_{(2)})$ be a pair of cross-network neighbors of $(i_{(1)}, u_{(2)})$. Without loss of generality, suppose $\mathbf{S}_{j,v}^{(k)} \geq \mathbf{c1}_j^{(k)}$ but $\mathbf{S}_{j,v}^{(k)} < \mathbf{c2}_v^{(k)}$. This implies that there must exists a better alignemnt with higher similarity for $v_{(2)}$ at the kth iteration. We still want to accumulate the similarity between $j_{(1)}$ and $v_{(2)}$ to $\mathbf{S}_{i,u}^{(k)}$, at the same time, we should also consider the loss of similarity by aligning $j_{(1)}$ to $v_{(2)}$ (recall that we model similarity accumulation as a process of aligning neighbors). We introduce a simple measure called *net similarity*:

$$\mathbf{N}_{jv}^{(k)} = 2\mathbf{S}_{j,v}^{(k)} - \left[\frac{\mathbf{S}_{j,v}^{(k)} - \mathbf{c1}_j^{(k)}}{\mathbf{b1}_j^{(k)} - \mathbf{c1}_j^{(k)}} \cdot (\mathbf{b2}_v^{(k)} - \mathbf{c2}_v^{(k)}) + \mathbf{c2}_v^{(k)} \right] \tag{3.4}$$

Algorithm 1: Contribution Threshold

Input: $G = (V, E)$, t_{max}
Output: Contribution threshold matrix $\mathbf{T}$

$\mathbf{T} \leftarrow n \times t_{max}$ empty matrix ▷ $|V| = n$
for i ***in*** V **do**
  $\mathbf{d} \leftarrow n \times 1$ vector with all entries equal to 0
  $\mathbf{d}_i = 1$
  $num\_of\_visited\_node \leftarrow 1$
  $frontier \leftarrow$ empty list.
  $frontier$.**insert**(i)
  for $k \leftarrow 1$ ***to*** t_{max} **do**
    $new\_frontier \leftarrow$ empty list.
    for j ***in*** $frontier$ **do**
      for q ***in*** $N(j)$ **do**
        if $\boldsymbol{d}_q == 0$ **then**
          $num\_of\_visited\_node+ = 1$
          $new\_frontier$.**insert**(q)
          $\mathbf{d}_q = 1$
    $\mathbf{T}_{i,k} = \frac{num\_of\_visited\_node}{n}$
    $frontier = new\_frontier$
return $\boldsymbol{T}$

and if $\mathbf{S}_{j,v}^{(k)} \geq \mathbf{c2}_v^{(k)}$ but $\mathbf{S}_{j,v}^{(k)} < \mathbf{c1}_j^{(k)}$:

$$\mathbf{N}_{jv}^{(k)} = 2\mathbf{S}_{j,v}^{(k)} - \left[\frac{\mathbf{S}_{j,v}^{(k)} - \mathbf{c2}_v^{(k)}}{\mathbf{b2}_v^{(k)} - \mathbf{c2}_v^{(k)}} \cdot (\mathbf{b1}_j^{(k)} - \mathbf{c1}_j^{(k)}) + \mathbf{c1}_j^{(k)} \right] \tag{3.5}$$

which leads to our second rule:

Rule 2. Given a pair of cross-network vertices $(j_{(1)}, v_{(2)})$, under rule 1, the amount of similarity they can contribute to a pair of neighbors $(i_{(1)}, u_{(2)})$ is :

$$\begin{cases} \mathbf{S}_{j,v}^{(k)} & \text{if } \mathbf{S}_{j,v}^{(k)} \geq \max\{\mathbf{c1}_j^{(k)},\ \mathbf{c2}_v^{(k)}\} \\ \mathbf{N}_{j,v}^{(k)} & \text{if } \min\{\mathbf{c1}_j^{(k)},\ \mathbf{c2}_v^{(k)}\} \leq \mathbf{S}_{j,v}^{(k)} \leq \max\{\mathbf{c1}_j^{(k)},\ \mathbf{c2}_v^{(k)}\} \\ 0 & \text{otherwise} \end{cases}$$

3.1.3 Rule 3 - Prioritization

Given the first two rules, it is possible that a neighbor of $u_{(2)}$ is globally most similar to multiple neighbors of $i_{(1)}$, but with different similarities. For example, consider a sample graph shown in Figure (3.2) where $v_{(2)}$ is globally most similar to both $j_{(1)}$ and $k_{(1)}$. Dashed lines indicate the similarities between two vertices.

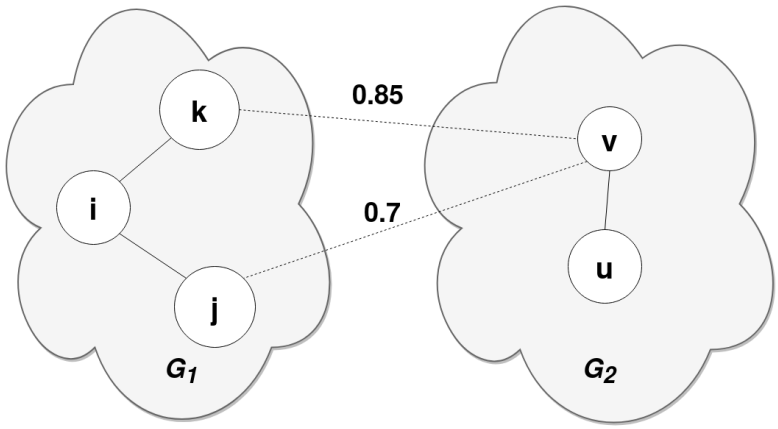

Figure 3.2: Example of rule 3

In this case, we ought to only consider the pair $(k_{(1)}, v_{(2)})$ and contribute its similarity to $\mathbf{S}_{i,u}$ which leads to the final rule:

> **Rule 3.** During the computation of the similarity between $i_{(1)}$ and $u_{(2)}$, a pair of cross-network neighbors with a **higher similarity should be given the prior consideration**.

3.1.4 The Similarity Computation Algorithm

The general idea of the `ELRUNA` is to update **S**, **b1** and **b2** iterativley based on their values in the previous iteration. **The initial similarities between each pair of cross-network vertices are uniformly distributed. We set them all equal to** 1. That is, $\mathbf{S}^{(0)}$ is a matrix of ones. Note that many other algorithms requires prior knowledge about the similarities (usually based on non-network information) between vertices, whereas

ELRUNA does not.

Algorithm 2: Similarity Computation

Input: $G_1 = (V_1, E_1)$, $G_2 = (V_2, E_2)$, t_{max}, **T1**, **T2**
Output: The similarity matrix **S**

for $k \leftarrow 1$ ***to*** t_{max} **do**
  $\mathbf{S}^{(k)} \leftarrow n_1 \times n_2$ similarity matrix
  $\mathbf{b1}^{(k)} \leftarrow n_1 \times 1$ vector with all entries equal to -1
  $\mathbf{b2}^{(k)} \leftarrow n_2 \times 1$ vector with all entries equal to -1
  Update $\mathbf{c1}^{(k-1)}$ and $\mathbf{c2}^{(k-1)}$ based on equation 3.3
  for i ***in*** V_1 **do**
    for u ***in*** V_2 **do**
      e $\leftarrow$ empty associative array
      $sum \leftarrow 0$
      for j ***in*** $N(i)$ **do**
        for v ***in*** $N(u)$ **do**
          if $\mathbf{S}_{jv}^{(k-1)} \geq \min\{\mathbf{c1}_j^{(k-1)}, \mathbf{c2}_v^{(k-1)}\}$ **then**
            $\mathbf{e}[(j, v)] \leftarrow \mathbf{S}_{jv}^{(k-1)}$
      e $\leftarrow$ sort by ***value*** in descending order
      for (j, v) ***in*** **e**.***keys*** **do**
        if j *and* v *are not selected* **then**
          Accumulate sum based on rule 2
          Mark j and v as selected
      $\mathbf{S}_{iu}^{(k)} \leftarrow \frac{sum}{\max\{\sum_{j \in N(i)} \mathbf{b1}_j^{(k-1)}, \sum_{v \in N(u)} \mathbf{b2}_v^{(k-1)}\}}$
      if $\mathbf{S}_{iu}^{(k)} > \mathbf{b1}_i^{(k)}$ **then**
        $\mathbf{b1}_i^{(k)} = \mathbf{S}_{iu}^{(k)}$
      if $\mathbf{S}_{iu}^{(k)} > \mathbf{b2}_u^{(k)}$ **then**
        $\mathbf{b2}_u^{(k)} = \mathbf{S}_{iu}^{(k)}$
return ***S***

The detailed algorithm is summarized in Algorithm (2). Overall, at the $(k-1)^{th}$ iteration, for each pair of cross-network vertices $(i_{(1)}, u_{(2)})$, we first check all pairs of their cross-network neighbors against Rule 2 to determine which pairs are qualified such that the similarity is greater than the contribution threshold of at least one node in the pair. Then we sort those pairs by similarities in descending order which is needed to follow Rule 3. After sorting, we go over each $(j_{(1)}, v_{(2)})$ in the sorted order, check $j_{(1)}$ and $v_{(2)}$ against rule

1. If none of them has contributed to $\mathbf{S}_{i,u}^{(k)}$ before, we accumulate the similarity between $(j_{(1)}, v_{(2)})$ based on Rule 2 and mark $j_{(1)}$ and $v_{(2)}$ as *selected* which indicates that they can no longer be considered. This step enforces Rule 1. Note that the *selected* neighbors will no longer be selected after we are done computing $\mathbf{S}_{i,u}^{(k)}$. At last, we update $\mathbf{S}^{(k)}$, $\mathbf{b1}^{(k)}$ and $\mathbf{b2}^{(k)}$.

After accumulating similarities from neighbors, we normalize it by:

$$\mathbf{S}_{iu}^{(k+1)} = \frac{\text{accumulated similarity}}{\max\{\sum_{j \in N(i)} \mathbf{b1}_j^{(k)}, \sum_{v \in N(u)} \mathbf{b2}_v^{(k)}\}}$$

$\mathbf{S}_{iu}^{(k+1)}$ is set to 0 if $\max\{\sum_{j \in N(i)} \mathbf{b1}_j^{(k)}, \sum_{v \in N(u)} \mathbf{b2}_v^{(k)}\} = 0$. This normalization also penalizes the degree discrepancy between $i_{(1)}$ and $u_{(2)}$ because the maximum number of pairs of cross-network neighbors that can contribute similarity to $(i_{(1)}, u_{(2)})$ is upper bounded by the smaller degree between $i_{(1)}$ and $u_{(2)}$.

The similarity computation step of `ELRUNA` consists of running Algorithm (1) and (2) which outputs the final similarity matrix $\mathbf{S}$.

3.2 Step 2 : Building Alignment of Vertices

After obtaining the final cross-network similarity matrix $\mathbf{S}$, we use two methods to extract the mappings between vertices from two networks, namely, `naive` and `seed-and-extend` alignment methdos.

3.2.1 Naive Alignment

Following the literature [34], we sort all pairs of cross-network vertices by similarities in descending order. Then iteratively align the next pair of unaligned vertices until all nodes in the smaller network are aligned. It is worth noting that this is a relatively simple alignment method, while many other algorithms [14, 21, 7, 9, 10, 2, 23] use more complicated and computationally expensive alignment methods. However, as shown in the

experimental section, using this naive alignment method, `ELRUNA` significantly outperforms other baselines.

While the `naive` alignment method produces alignments with good quality, we observe that it fails to distinguish nodes that are topologically symmetric. As an example shown in Figure (3.3) where G_1 and G_2 are isomorphic. Under this circumstance, $k_{(1)}$ is equally similar to $v_{(2)}$ and $w_{(2)}$. At the same time, $m_{(1)}$ is equally similar to $x_{(2)}$ and $y_{(2)}$. If we break ties randomly, then it is possible that $k_{(1)}$ and $m_{(1)}$ are mapped to vertices on different branches of the tree which causes the loss of the number of conserved edges.

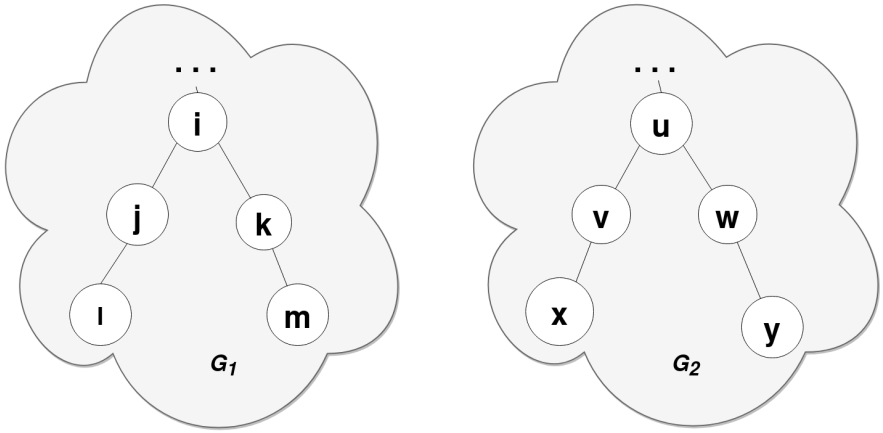

Figure 3.3: Naive alignment fails to distinguish symmetric nodes

3.2.2 Seed-and-extend Alignment

During the alignment process, nodes that have been aligned can serve as guidance for aligning other nodes [21]. Referring back to the Figure (3.3), aligning $k_{(1)}$ to $w_{(2)}$ should imply that $m_{(1)}$ ought to be aligned with $y_{(2)}$ rather than $x_{(2)}$. To address the limiation of the `naive` alignment methods, we iteratively find the pair of **unaligned** nodes with the highest similarity. Then align them and increase similarities between every pair of their **unaligned** cross-network neighbors by some small constant. Iterations proceed until all nodes in the smaller network are aligned. For efficiency, we use red-black tree to store pairs

of nodes.

3.3 Time complexity of ELRUNA

Without loss of generality, we assume that two networks has comparable number of vertices and edges. Let n and m denote the number of vertices and edges, respectively. Let t_{max} denote the total number of iterations. It is easy to see that Algorithm (1) runs in $O(n^2 + mn)$ time and the naive alignment method runs in $O(n^2 \log n)$ time.

Lemma 1. *The time complexity of Algorithm (2) is* $O(t_{max} m^2 \log n)$

Proof. Let d_i denote the degree of vertex $i_{(1)}$. Operations on lines 12 to 13 and lines 19 to 23 takes constant time thus the nested for loops on lines 10 to 13 take $\Theta(d_i d_u)$ time for every pair of $i_{(1)}$ and $u_{(2)}$. Sorting on line 14 takes $\Theta(d_i d_u \log d_i d_u)$ time and the for loop on lines 15 to 18 takes $\Theta(d_i d_u)$ time. We observe that $O(\log d_i d_u) = O(\log n)$, therefore, the outer nested for loop on lines 6 to 23 takes:

$$\begin{aligned} O(\sum_{i \in V_1} \sum_{u \in V_2} d_i d_u log(d_i, d_u)) &= O(\log n \sum_{i \in V_1} d_i \sum_{u \in V_2} d_u) \\ &= O(m^2 \log n) \end{aligned} \tag{3.6}$$

Finally, the time complexity of the Algorithm (2) is $O(t_{max} m^2 \log n)$. □

Lemma 2. *The time complexity of the* `seed-and-extend` *alignment method is* $O((n^2 + mn) \log(n^2 + mn))$.

Proof. At the beginning of the algorithm, adding all n^2 pairs of nodes into the red-black tree takes $O(n^2 \log n)$ time. Whenever we align a pair of vertices $(i_{(1)}, u_{(2)})$, we increases the similarities between all pairs of their unaligned cross-network neighbors. To update the corresponding similarities in the red-black tree, we add those pairs with new similarities into the tree. The total number of insertions of such pairs is

$$\sum_{i_{(1)} \in V_1} d_i d_{f(i)}$$

where $f(i) \in V_2$ is the node that $i_{(1)}$ is aligned to. It is easy to see that the function above is upper bounded by $O(mn)$ which implies that the total number of elements in the tree is $O(n^2+mn)$. We perform at most n^2+mn number of *find_max*, *deletion* and *insertion* operations, therefore, the overall running time of the algorithm is $O((n^2 + mn)\log(n^2 + mn))$. □

t_{max} equals to the diameter of the larger network. We assume $O(m) = O(n \log n)$ which is a fair assumption for the sparse networks that are used in the experiments. Therefore, the overall running time of the proposed `ELRUNA` is $O(t_{max} n^2 \log^3 n)$. Note that for all the networks we are testing, t_{max} is at most 20.

We acknowledge that the combination of rule 1 and rule 3 is equivalent to solving the maximum matching problem and there are faster algorithms to perform such tasks [6]. Also, there exists room for improvements of `ELRUNA` such that we do not need to sort $\mathbf{S}$ at neighborhood scale. We will explore these approaches in the future works.

Chapter 4

RAWSEM : Random-walk Based Selection Method

In this section, we introduce the proposed selection rule RAWSEM for local search procedure. We first discuss the baseline which is used as the comparison method to RAWSEM. We then present the mechanism of RAWSEM.

4.1 The Baseline

Following the literature [28], given the permutation matrix produced by any alignment algorithm, the baseline algorithm constructs the search space for local search by selecting a subset of vertices from the smaller network and generate all permutations of their alignments while fixing the alignment of all other vertices that are not in the subset [28].

Given $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$, we transform the alignment matrix $\mathbf{P}$ to the $n_1 \times 1$ alignment vector $\widetilde{\Pi}$ for which $\widetilde{\Pi}_i = u_{(2)}$ implies vertex $i_{(1)}$ is aligned to vertex $u_{(2)}$.

For each iteration, we randomly select a subset of vertices $V_1' \subset V_1$ with a fixed cardinality. Let $V_2' = \{\widetilde{\Pi}_i : i \in V_1'\}$. Let $\mathbf{A}$ and $\mathbf{B}$ be the adjacency matrices of G_1 and G_2,

respectively. The baseline local search explores all feasible solutions in the search space and attempts to find a new alignment vector Π with a lower objective:

$$\min_{\Pi}\{-\sum_{i,j\in V_1'}(\mathbf{A}_{ij}\mathbf{B}_{\Pi_i\Pi_j}+\sum_{k\in N(i)|k\notin V_1'}\mathbf{A}_{ik}\mathbf{B}_{\Pi_i\widetilde{\Pi}_k}+ \sum_{p\in N(j)|p\notin V_1'}\mathbf{A}_{jp}\mathbf{B}_{\Pi_j\widetilde{\Pi}_p})\} \quad (4.1)$$
$$\text{s.t. } \Pi_i \in V_2' \ \ \forall i \in V_1'$$

The baseline local search proceeds until an optimal has been reached, that is, the objective has not been improved in over a number of iterations.

4.2 RAWSEM Algorithm

Depending on the quality of the initial solution, it is possible that only a small fraction of vertices are not mapped optimally. Therefore, the baseline approach which constructs the subset V_1' with random selections from the entire vertex set is not efficient. Ideally, we want to locate vertices that are not mapped optimally and only permute the alignments between them.

For simplicity, suppose nodes in V_1 are label with consecutive integers starting from 1, and nodes in V_2 are label with consecutive integers starting from $|V_1|+1$. To quantify the level of mismatching of each vertex, we first define the concept of **violation**:

Definition 5 (**Violation**). *Let* $\mathbf{o}'$ *be a* $n_1 \times n_2$ *by* 1 *vector. The* ***violation*** *value of a vertex* $i_{(1)} \in G_1$ *is defined as*

$$\mathbf{o}_i' = \sum_{j\in N(i)}(1-\mathbf{B}_{\widetilde{\Pi}_i,\widetilde{\Pi}_j}) \quad (4.2)$$
$$= |N(i)| - \sum_{j\in N(i)}\mathbf{B}_{\widetilde{\Pi}_i,\widetilde{\Pi}_j}$$

Informally, $\mathbf{o}'_i$ *is the number of the neighbors of* $i_{(1)}$ *that are not conserved by* $i_{(1)}$*.*

Let $\bar{V}_2 = \{u_{(2)} \in V_2 \ : \ \widetilde{\Pi}_i = u_{(2)} \ \exists \ i_{(1)} \in V_1\}$ *be the subset of* V_2 *consisting all aligned vertices in* G_2*. Let* $\widetilde{\Pi}^{-1} : \bar{V}_2 \to V_1$ *be the inverse mapping, then the violation value of vertex* $u \in V_2$ *is defined in the same fashion:*

$$\begin{aligned} \mathbf{o}'_u &= \sum_{v \in N(u)} (1 - \mathbf{A}_{\widetilde{\Pi}_u^{-1}, \widetilde{\Pi}_v^{-1}}) \\ &= |N(u)| - \sum_{v \in N(u)} \mathbf{A}_{\widetilde{\Pi}_u^{-1}, \widetilde{\Pi}_v^{-1}} \end{aligned} \tag{4.3}$$

Finally, we normalize violations of vertices by their degrees. Let $\mathbf{o}$ be vector which encodes the normalized violation value of each vertex, we have:

$$\mathbf{o} = \mathbf{D}^{-1}\mathbf{o}' \tag{4.4}$$

where $\mathbf{D}$ is the diagonal degree matrix such that $\mathbf{D}_{i,i}$ is the degree of vertex i. $\mathbf{o}$ provides the *initial level of mismatching* of each vertex. We further normalize $\mathbf{o}$ such that its $L1$ norm equals to 1.

From a high level, `RAWSEM` is a two-step procedure:

(1) **Ranking vertices**: Starting from the initial level of mismatching of vertices, `RAWSEM` iteratively update $\mathbf{o}$ in a `PageRank` [22] fashion. Then, rank vertices by the the converged levels of mismatching.

(2) **Local search**: Construct the search space based on the ranking and perform the local search.

4.2.1 Step 1 - Ranking Vertices

For two real world networks where isomorphism does not exist, we expect many vertices to have nonzero initial violations. Clearly, a nonzero violation does not imply a non-optimal mapping. It it worth noting that a zero violation does not always imply an

optimal mapping. As shown in the Figure (4.1) where dashed lines indicate mappings, $i_{(1)}$ has zero violation. This suggests the insufficiency of initial violation values.

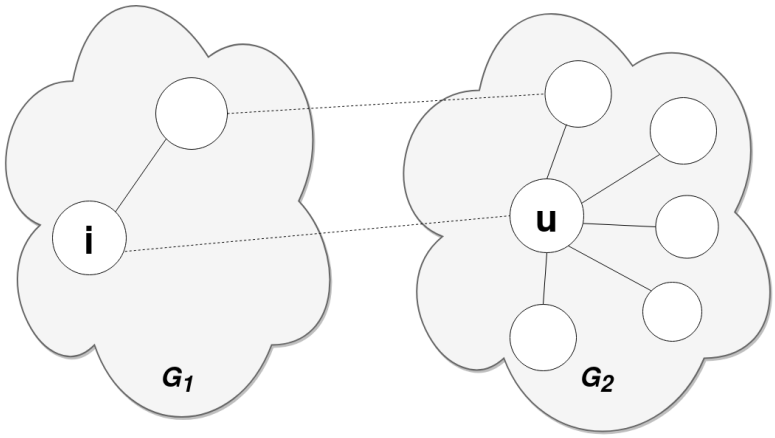

Figure 4.1: Zero violation of vertex i

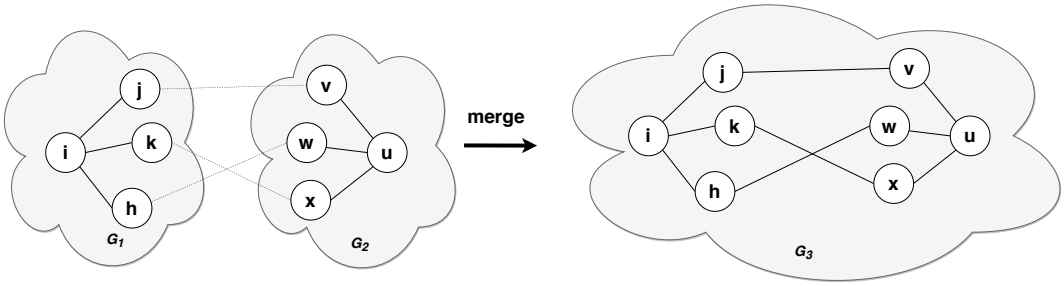

Figure 4.2: Example of the merge operaiton

To adapt the network information into our model, we use an iterative approach based on the intuition that *a vertex is more mismatched if its neighbors are more mismatched*. One immediate method is to propagate violation value via edges throughout the networks. Let $G_2' = (\bar{V}_2, \bar{E}_2)$ be the subgraph induced by $\bar{V}_2$ (as stated above, $\bar{V}_2$ is the set of aligned vertices in G_2). To start with, we merge G_1 and G_2' by adding edges to connect aligned cross-network vertices. Figure (4.2) illustrates a example of the merge operation. Denote the newly constructed undirected graph $G_3 = (V_3, E_3)$ where $V_3 = V_1 \cup \bar{V}_2$ and $E_3 = E_1 \cup \bar{E}_2 \cup \{(i_{(1)}, u_{(2)}) : i_{(1)} \in V_1, u_{(2)} = \widetilde{\Pi}_i\}$. Let $\mathbf{C}$ denote the adjacency matrix of G_3. Let

$\mathbf{R}$ be the vector which encode the propagated levels of mismatching for each vertex. Let $\mathbf{D}$ denote the diagonal degree matrix such that $\mathbf{D}_{i,i} = |N(i)|$. We propagate violations in a `PageRank` [22] fashion:

$$\mathbf{R}_i^{(k)} = \alpha \sum_j \frac{\mathbf{C}_{ij}}{\mathbf{D}_{j,j}} \mathbf{R}_j^{(k-1)} + (1-\alpha)\mathbf{o}_i. \tag{4.5}$$

In matrix notation:

$$\mathbf{R}^{(k)} = \alpha \mathbf{C}\mathbf{D}^{-1}\mathbf{R}^{(k-1)} + (1-\alpha)\mathbf{o} \tag{4.6}$$

By initializing $\mathbf{R}^{(0)}$ as a probability vector, we can rewrite equation (4.6):

$$\mathbf{R}^{(k)} = \left[\alpha \mathbf{C}\mathbf{D}^{-1} + (1-\alpha)\mathbf{o}\mathbf{1}^T\right] \mathbf{R}^{(k-1)} \tag{4.7}$$

where $\mathbf{1}$ is the vector with all entries equal to 1.

The equation (4.7) encodes an eigenvalue problem which can be approximated by power iteration. Let $\mathbf{E} = \alpha \mathbf{C}\mathbf{D}^{-1} + (1-\alpha)\mathbf{o}\mathbf{1}^T$. By the undirected nature of G_3, the transition matrix $\mathbf{C}\mathbf{D}^{-1}$ is *irreducible*, therefore $\mathbf{E}$ is a left-stochastic matrix with leading eigenvalue equal to 1 and the solution of equation (4.7) is the principle eigenvector of $\mathbf{E}$. On top of that, $\mathbf{E}$ is also primitive therefor the leading eigenvalue of $\mathbf{E}$ is unique and the corresponding principle eigenvector can be chosen to be strictly positive. As a result, the power iteration converges to its principle eigenvector.

Violation vector $\mathbf{o}$ plays the role of teleportation distribution which encodes external influences on the importance of vertices. The converged $\mathbf{R}$ gives the levels of mismatching of vertices and a vertex with a higher value is more mismatched. Finally, we rank vertices in G_1 based on $\mathbf{R}$ in descending order.

4.2.2 Step 2 - Local Search

Based on the ranking produced in the previous step, `RAWSEM` uses a sliding window over sorted vertices to narrow down the search space. Let m be the size of the window with the tail lying at the highest-ranked vertices. For each iteration, we construct V_1' by randomly selecting vertices within the window and perform the local search. If the objective has not been improved for s iterations, we move the window l nodes forward then continues the local search procedure. The local search process terminates when the objective has not been improved for s_{max} number of iterations. In our experiments, we set $|V_1| = 6$.

Chapter 5

Experimental Results

In this section, we first present the experimental setup and performance of the proposed ELRUNA in comparison with 8 baseline methods over 3 alignment scenarios. Then, we study the time-quality trade-off and the scalability of ELRUNA. *We emphasize, that ELRUNA is not a local search method and should not be compared to other local search methods as ELRUNA can serve as a preprocessing step to all local searches.* Finally, we demonstrate that RAWSEM could further improve the alignment quality with significant less number of iterations than the naive local search method.

Reproducibility: Our source code, documentation, and data sets are available at https://tinyurl.com/uwn35an.

Baselines for ELRUNA. We compare ELRUNA against 8 state-of-the-art network alignment algorithms: IsoRank [29], Klau [14], NetAlign [2], REGAL [12], EigenAlign [7], C-GRAAL [20], NETAL [21], and HubAlign [10]. These algorithms published in year 2008 - 2019 have found to be superior to many other methods, so we decide to choose them. Other methods that perform in significantly longer running time have not been considered. The baseline methods are described in the related work section. C-GRAAL, Klau and Netalign require prior similarities between cross-network vertices as input. Following [34, 12], we use the degree similarities as the prior similarities. For Klau and Netalign, as suggested by [12], we construct the prior alignment matrix by choose the highest $k \times \log_2 n$ vertices where

$k = 5$. Note that ELRUNA **does not requires prior knowledge about the similarities between cross-network vertices**.

We do not compare ELRUNA to FINAL [34] because FINAL solves a different problem, namely, the attributed network alignment problem. ModuleAlign [9] is also not chosen as a baseline method because it is the same as HubAlign [10], except that ModuleAlign uses a different method to optimize the biological similarities between vertices. Additionally, we do not compare with GHOST [23] because its signature extraction step took hours even for small networks.

Experimental Setup for ELRUNA. All experiments are performed on an Intel Xeon E5-2670 machine with 64GB of RAM. For the sake of iterative progress comparison, we set the maximum number of iterations t_{max} to the larger diameter of the two networks.

Our experiment consists of 3 scenarios: (1) ***Self-alignment without and under the noise***. (2) ***Alignment between homogeneous networks***. (3) ***Alignment between heterogeneous networks***. Detailed descriptions of each category are presented in the later section. In general, the first test case *self-alignment without and under the noise* consists of 12 networks from various domains. For each network, we generate up to 14 noisy copies with increasing noise levels (defined later) up to 25%. In total, this gives us a total of 164 pairs of network to align. The second test scenario *Alignment between homogeneous networks* has 3 pairs of networks for which each pair consists of two subnetworks of a larger network. In our third test case *Alignment between heterogeneous networks*, we align 5 pairs of networks where each pair consists of two networks from different domains. It is worth noting that the third comparison scenario is used by attributed network alignment algorithms. Therefore, it is not exactly what we solve with our formulation since ELRUNA only relies on the toplogy of the networks. However we still demonstrate the results because it is a practically important task. The experimental results of the third test case is shown in the Appendix section. To the best of our knowledge, our experimental setup is the most comprehensive in terms of the combination of the number of baselines, the number of networks, the categories of testing cases, and the levels of noise applied.

Evaluation Metric. To quantify the alignment quality, we use two well-known metrics: the *edge correctness* (EC) [21] and the *symmetric substructure score* (S^3) [27]. Let $f(V_1) = \{u \in V_2 : \mathbf{P}_{i,u} = 1, \exists i \in V_1\}$, and $f(E_1) = |\{(f(i_{(1)}), f(j_{(1)})) \in E_2 \ : \ (i_{(1)}, j_{(1)}) \in E_1\}|$. That is, $f(V_1)$ is the set of vertices in G_2 that are aligned (note that since we assume $|V_1| \leq |V_2|$, some vertices in G_2 are left unaligned), and $f(E_1)$ is the set of edges in G_2 such that for each edge, the alignment of its incident vertices are adjacent in G_1. Then, we have

$$EC = \frac{|f(E_1)|}{|E_1|} \tag{5.1}$$

and

$$S^3 = \frac{|f(E_1)|}{|E_1| + |E(G_2[f(V_1)])| - |f(E_1)|} \tag{5.2}$$

where $|E(G_2[f(V_1)])|$ is the number of edges in the subgraph of G_2 induced by $f(V_1)$.

5.1 Self-alignment without and under the noise

In this experiment, we analyze how `ELRUNA` performs under structure noise being added to the original network. Given a target network G_1, simply aligning G_1 with its random permutations is not a challenge task for most of the existing state-of-the-art network alignment algorithms. A more interesting test cases arises when we try to align the original networks G_1 with its **noisy** permutations G_2 such that G_2 is a copy of G_1 with additional edges being added [21, 15, 34, 33, 11]. This scenario is more challenging when the number of noisy edges is large with respect to the number of edges in G_1. In addition, this perturbation approach also reflects many real-life network alignment task scenarios [2, 11, 12].

Given the network $G_1 = (V_1, E_1)$, a noisy permutation of G_1 with noise level p, denoted by $G_2^{(p)} = (V_2, E_2)$ is created with two steps:

1. Permute G_1 with some randomly generate permutation matrix.

2. Add $p|E_1|$ edges to G_1 uniformly at random by randomly connecting nonadjacent

pairs of vertices.

Properties of each network G_1 is given in Table 5.1. Note that under this model, the highest EC any algorithm can achieve by aligning G_1 and G_2 is always 1. ***In this experiment, we demonstrate the results WITHOUT applying local search, i.e., we demonstrate how ELRUNA outperforms the state-of-the-art methods.***

Table 5.1: Networks for test case: *self-align without and under noise*

Doamin	n	m	**label**
Barabasi random network	400	2,751	`barabasi`
Homle random network	400	2,732	`homel`
Coauthorships	379	914	`co-auth_1`
Gene functional association	993	1,300	`bio_1`
Economy	1,258	7,513	`econ`
Router	2,113	6,632	`router`
Protein-protein interaction	2,831	4,562	`bio_2`
Twitter	4,171	7,059	`retweet_1`
Erods Collaboration	5,019	7,536	`erdos`
Twitter	7,252	8,061	`retweet_2`
Social Interaction	10,680	24,316	`social`
Google+	23,628	39,242	`google+`

To demonstrate the effectiveness of `ELRUNA`, we first solve the alignment problems on random networks generated by Baràbasi-Albert preferential attachment (BA model) [1] and Holme-Kim model (HK model) [13], see Table 5.1. The HK model reinforces BA model with an additional probability q of creating a triangle after connecting a new node to an existing node. In our experiment we set $q = 0.4$. For each of the random network, we generate 12 noisy permutations with increasing noise level p from 0 to 0.21. We then align G_1 with each of its noisy permutations using the `ELRUNA` and baseline algorithms. `ELRUNA` has two versions: `ELRUNA_Naive` and `ELRUNA_Seed`, that differ by the alignment methods

we use. The results are summarized in the Figure 5.1 and 5.2.

Next, we solve the network alignment problem on 10 real-world networks [25, 17, 3] from various domains. The details of selected networks are shown in Table 5.1. For each network, we use the same model to generate 14 noisy permutations with increasing noise level p from 0 up to 0.25. That is, $G_2^{(0.25)}$ has added additional 25% noizy edges added to G_1. We then align G_1 with each of its permutation using the `ELRUNA` and baseline algorithms. The results are summarized from Figure 5.3 to 5.12.

Results. It is clear that `ELRUNA` significantly outperforms all baselines on all networks. In particular, both versions of `ELRUNA` (`ELRUNA_Naive` and `ELRUNA_Seed`) outperform 6 existing methods (`REGAL`, `EigenAlign`, `Klau`, `IsoRank`, `C-GRAAL` and `NetAlign`) by an order of magnitude under high noise levels. For the other two baselines (`NETAL` and `HubAlgin`), `ELRUNA` also produces much better results than them, with improvement up to 60% under high noise levels. At the same time, both versions of `ELRUNA` are robust to noise such that they output high-quality alignment even when p reaches 0.25. Noisy edges change the degrees of vertices by making them more uniformly distributed, i.e., performs a process that can be viewed as network anonymization. This experiment demonstrates `Elruan`'s superiority in identifying the hidden isomorphism between two networks.

We observe that the alignment quality of `Klau` and `NetAlign` are very similar. Such behaviors were also exibited in other literature [12, 33]. We also note that the trend of EC and S^3 are almost the same. `EigenAlign` crashed on `bio_1` (for $p > 0.17$), `bio_2`, `econ` (for $p > 0.05$), `router`, and `erdos` networks, and it took over 23 hours to even run on one instance of `social`, `google+`, and `retweet_2` networks. `C-GRAAL` crashed on `econ` (only ran successfully on $p = 0.3$), `google+` and `social` networks. `HubAlign` crashed on `google+` networks.

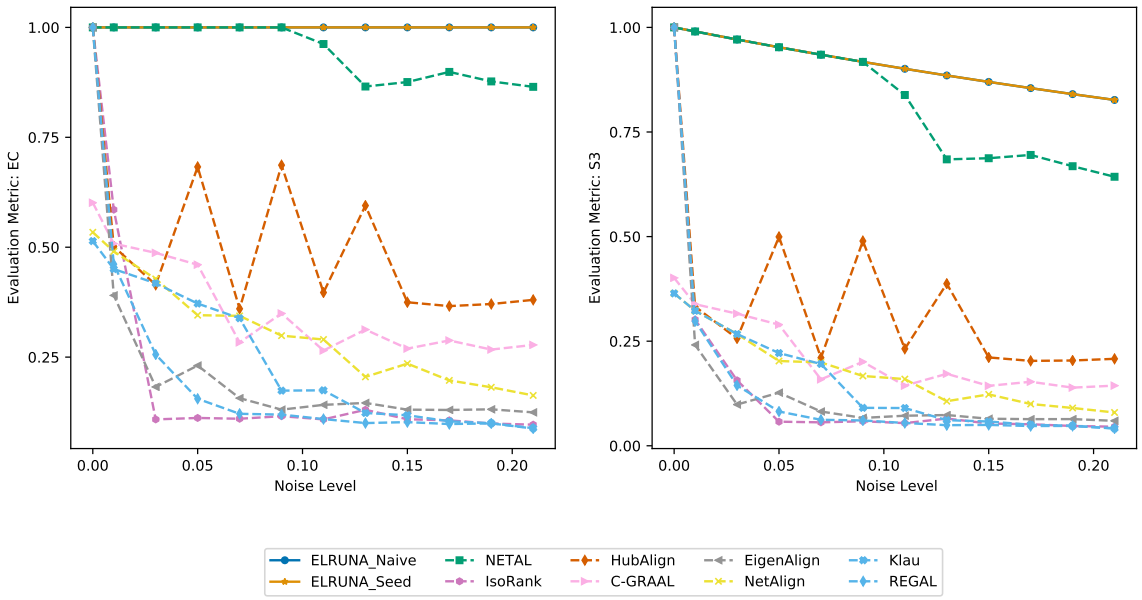

Figure 5.1: Alignment quality comparison on `barabasi` network

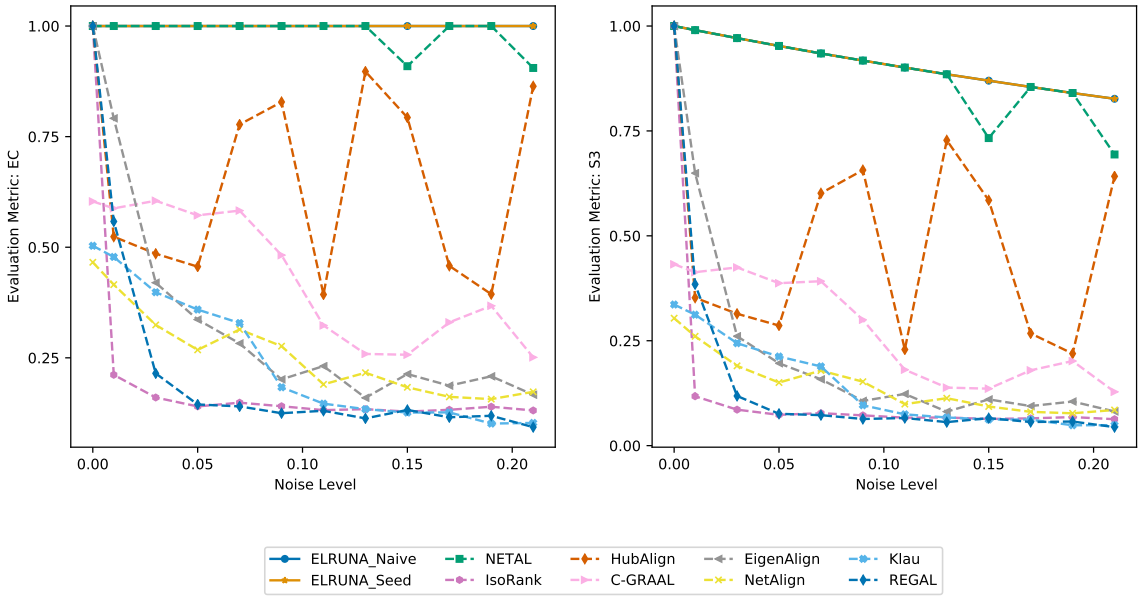

Figure 5.2: Alignment quality comparison on `homle` network

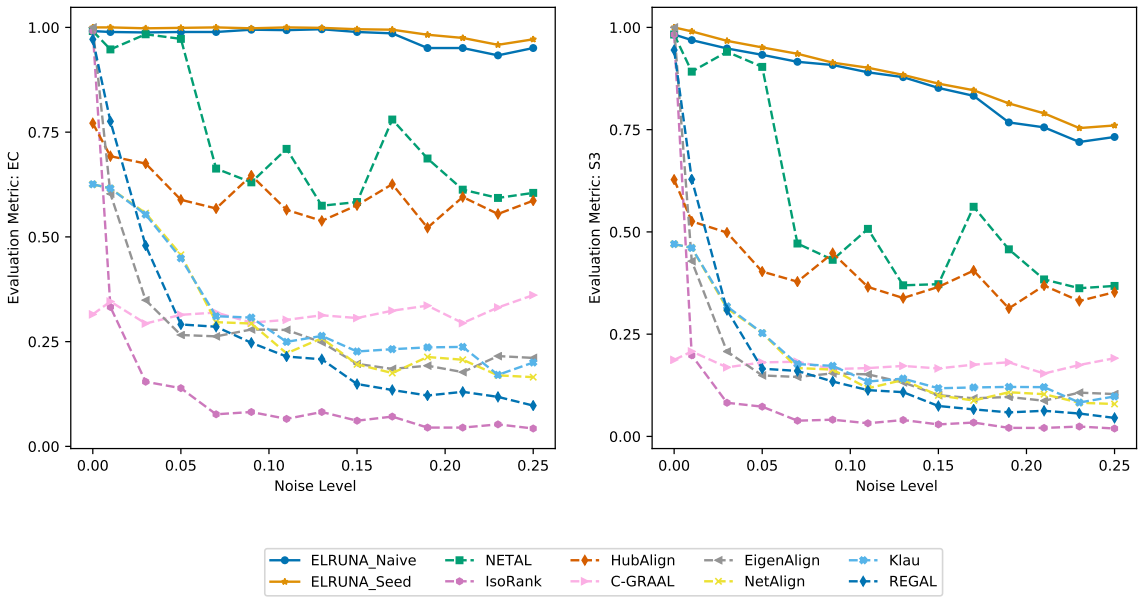

Figure 5.3: Alignment quality comparison on `co-auth_1` network

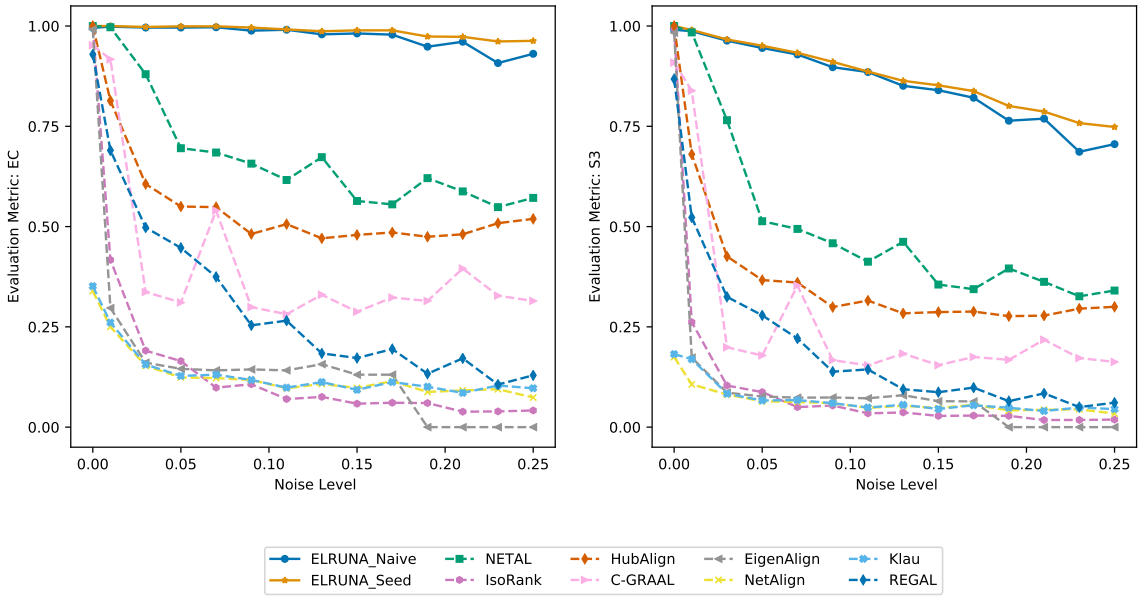

Figure 5.4: Alignment quality comparison on `bio_1` network

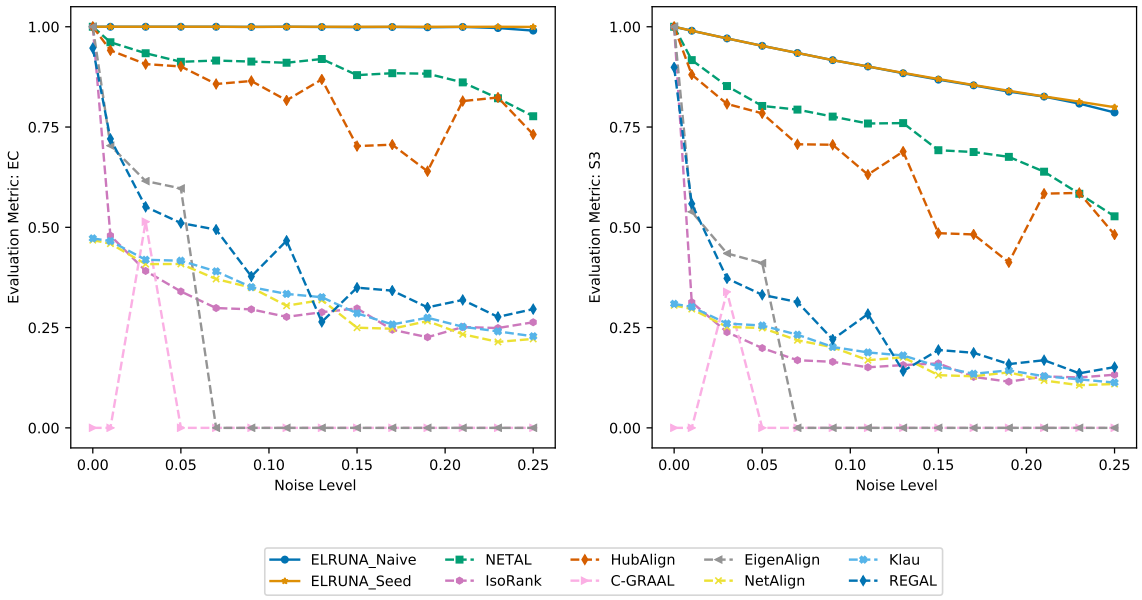

Figure 5.5: Alignment quality comparison on `econ` network

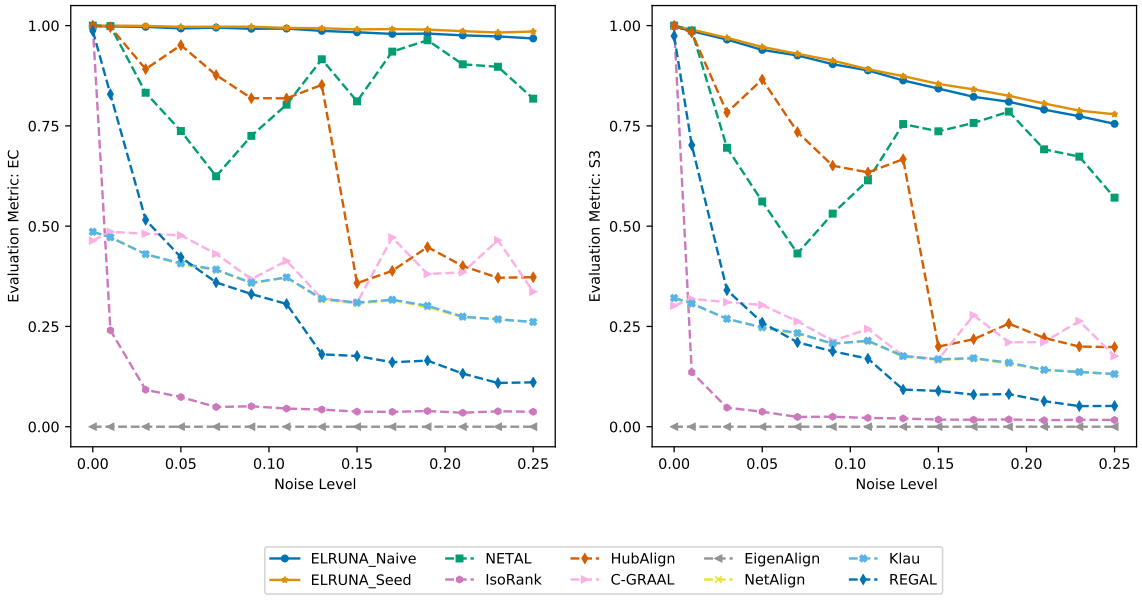

Figure 5.6: Alignment quality comparison on `router` network

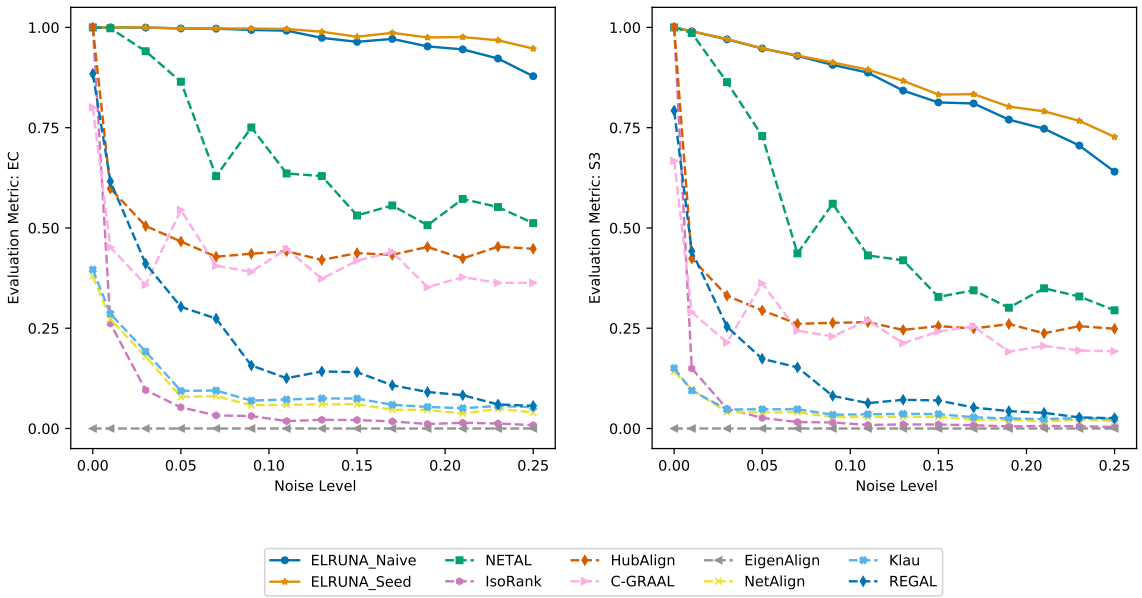

Figure 5.7: Alignment quality comparison on `bio_2` network

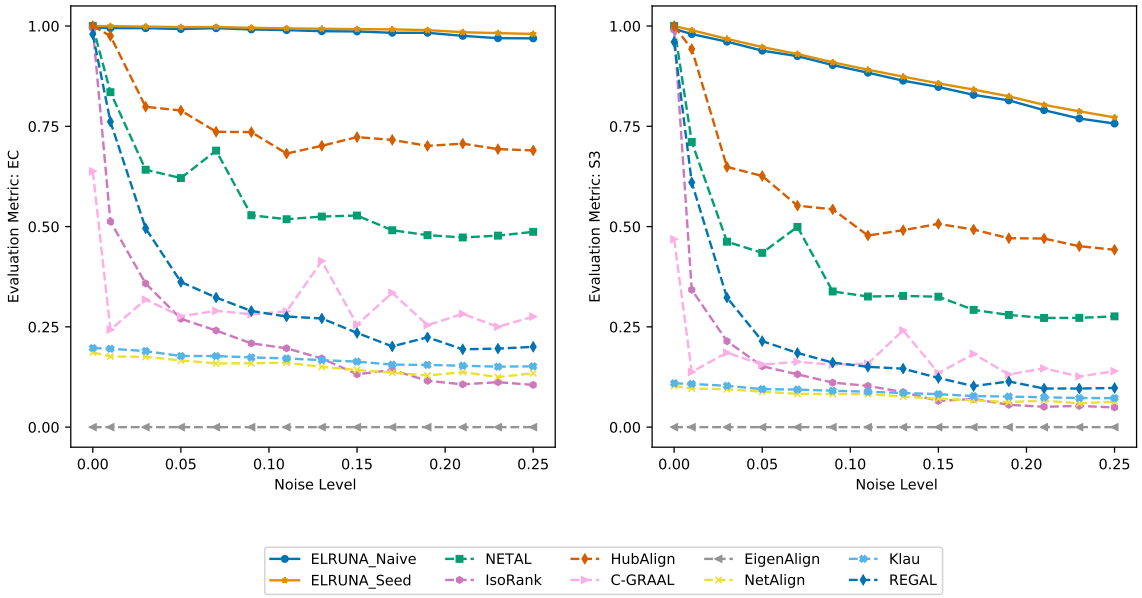

Figure 5.8: Alignment quality comparison on `retweet_1` network

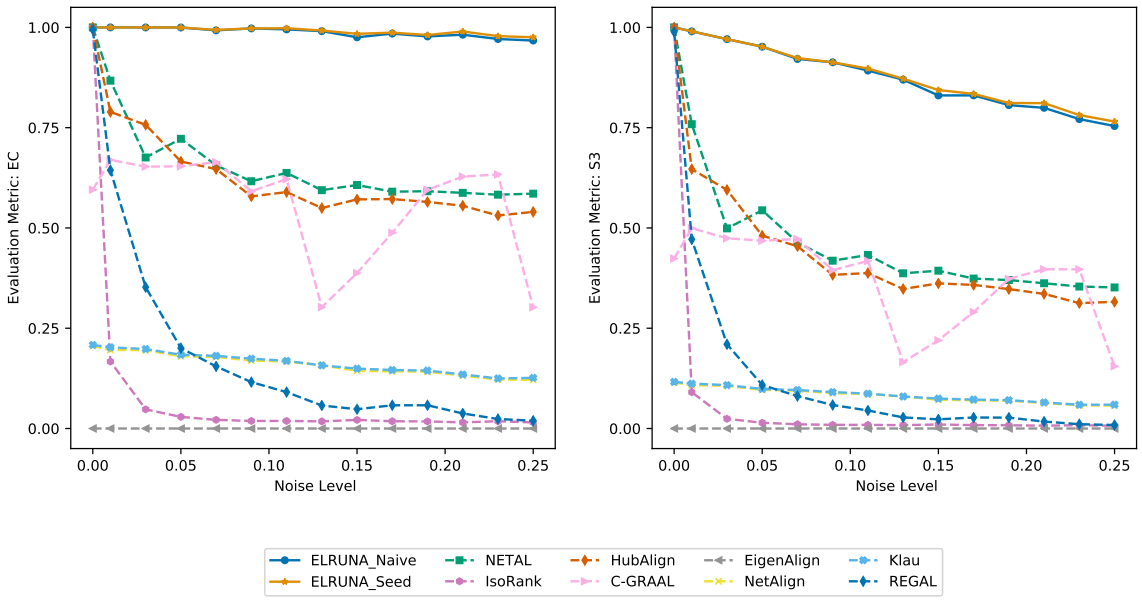

Figure 5.9: Alignment quality comparison on `erdos` network

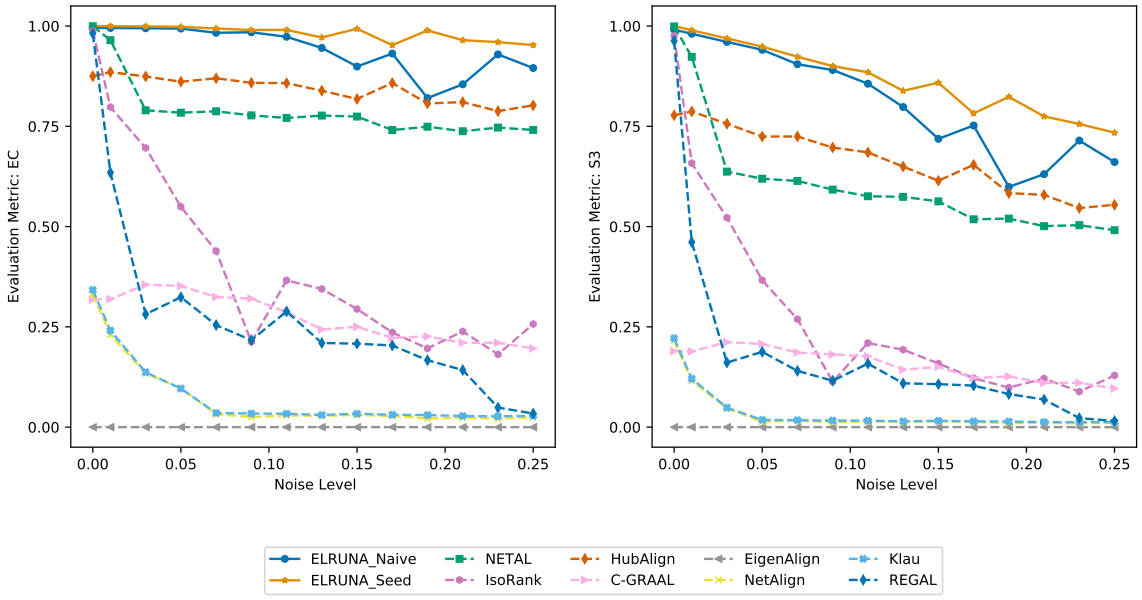

Figure 5.10: Alignment quality comparison on `retweet_2` network

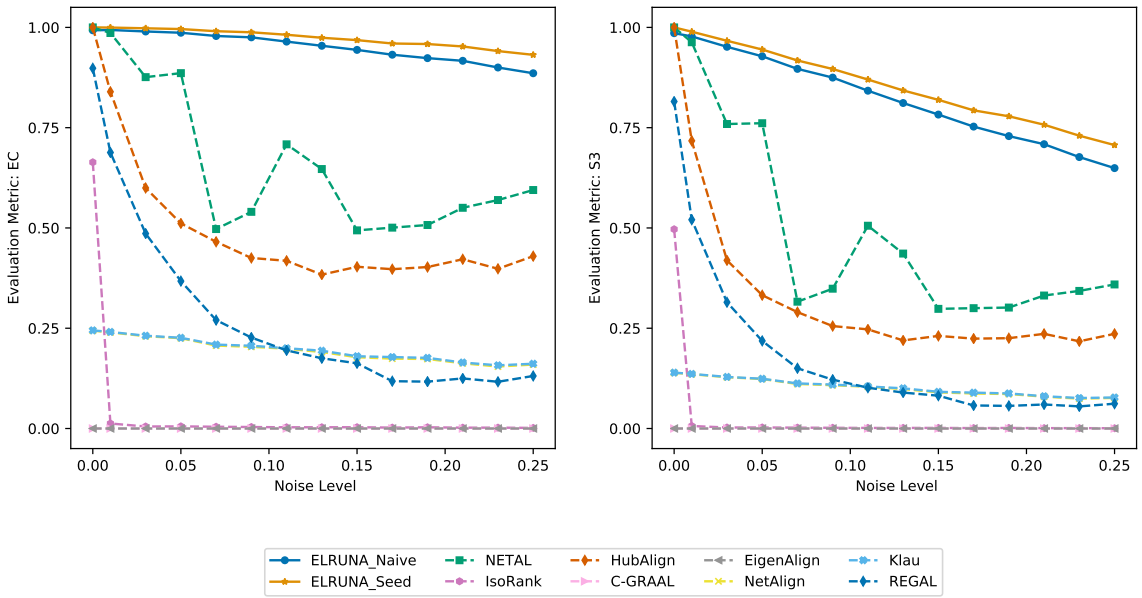

Figure 5.11: Alignment quality comparison on `social` network

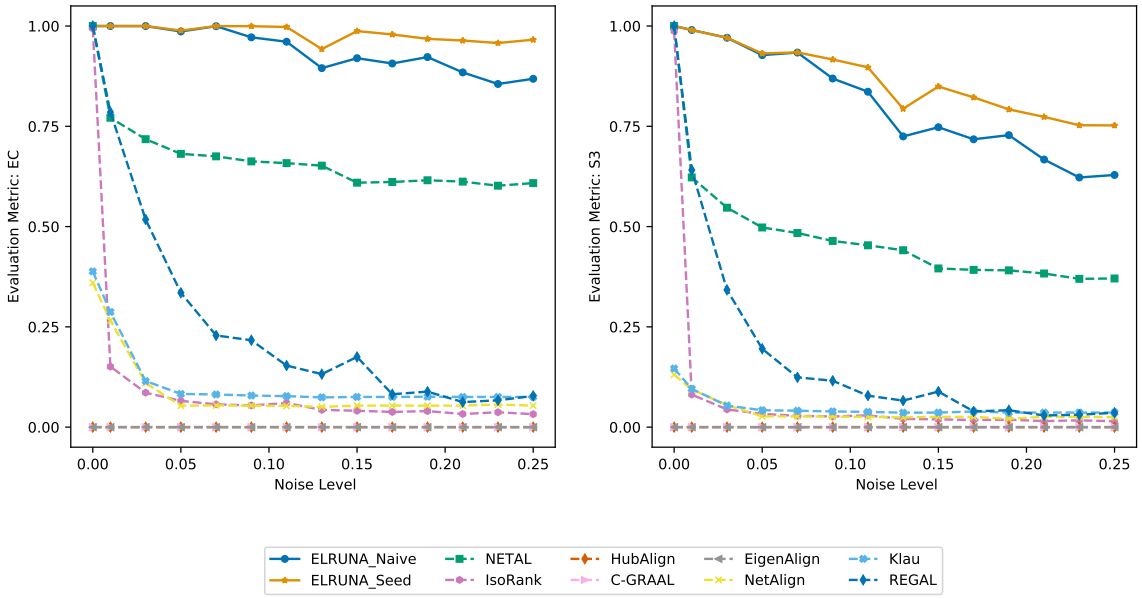

Figure 5.12: Alignment quality comparison on `google+` network

5.2 Alignment Between Homogeneous networks

In this experiment, we study how ELRUNA performs when aligning two networks that were subgraphs of a larger network. Given a network G, we extract two *induced* subnetworks G_1 and G_2 of G for which G_1 and G_2 share a common set of vertices. We compare ELRUNA_Naive and ELRUNA_Seed against the state-of-the-art methods by aligning 3 pairs of G_1 and G_2 [25, 4, 31] from 3 domains, respectively. The properties of networks are listed in Table 5.2. Note that the implementation of REGAL does not support alignment between networks with different sizes, therefore, it is not included in this testing case. In addition, the benchmark of EigenAlign is not included for the facebook networks because its running time was over 23 hours.

The first pair consists of two DBLP subnetworks with 2,455 overlapping nodes. The second pair of Digg social networks have 5,104 overlapping nodes, and lastly, the Facebook Friendship networks has 8,130 nodes in common. Note that under this setting, the highest EC any algorithm can achieve is lower than 1. In fact, the optimal EC value is not known.

Table 5.2: Datasets for test case: *Alignment between homogeneous networks*

Doamin	n	m	**label**
DBLP	3,134 vs 3,875	7,829 vs 10,594	dblp
Digg Social Network	6,634 vs 7,058	12,177 vs 14,896	digg
Facebook Friendship	9,932 vs 10,380	26,156 vs 31,280	facebook

In this experiment, we demonstrate the results WITHOUT local search, i.e., we demonstrate how ELRUNA outperforms the state-of-the-art methods. The results are shown in Figure 5.13, 5.14 and 5.15.

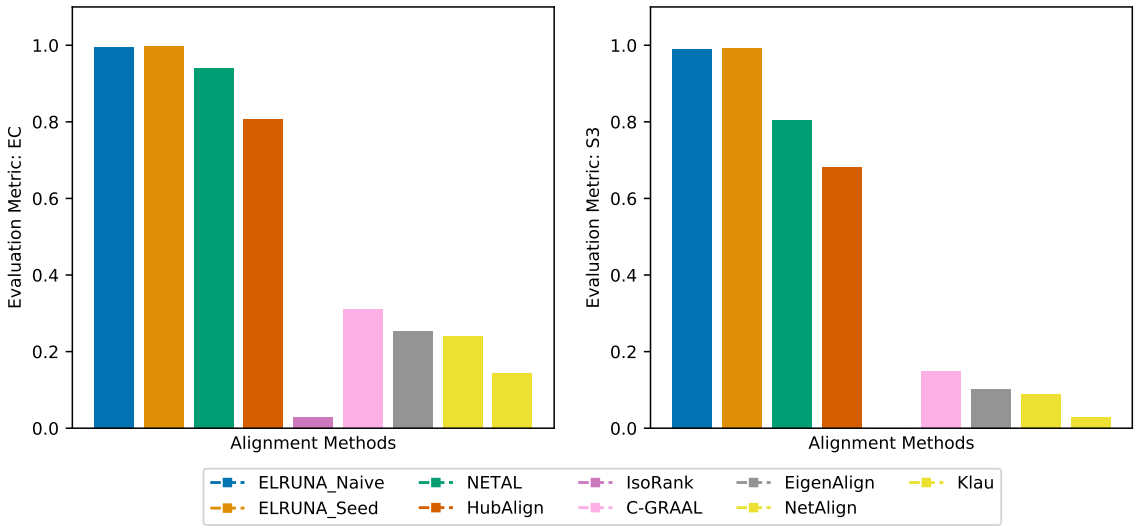

Figure 5.13: Alignment quality comparison on `dblp` networks

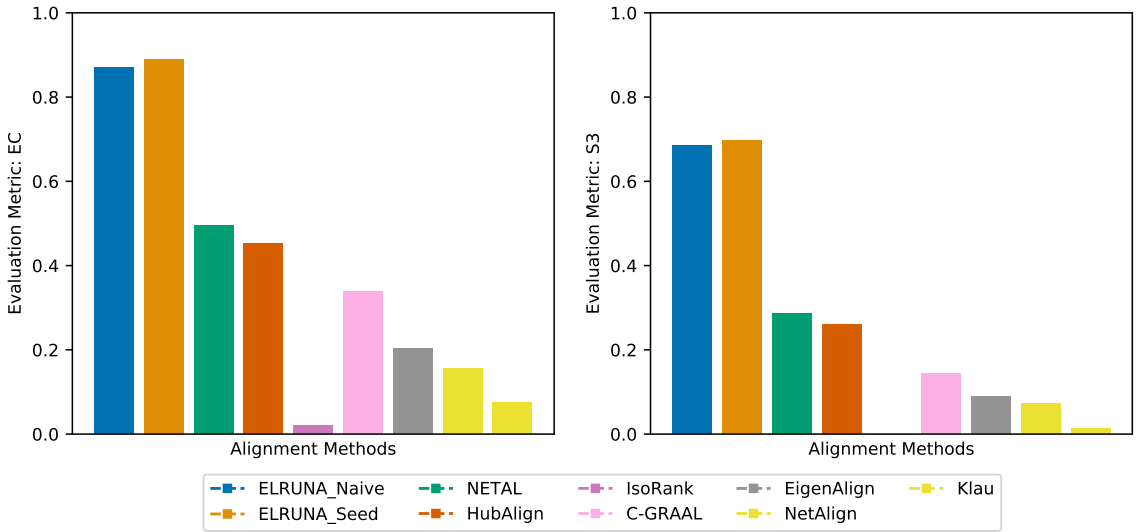

Figure 5.14: Alignment quality comparison on `digg` networks

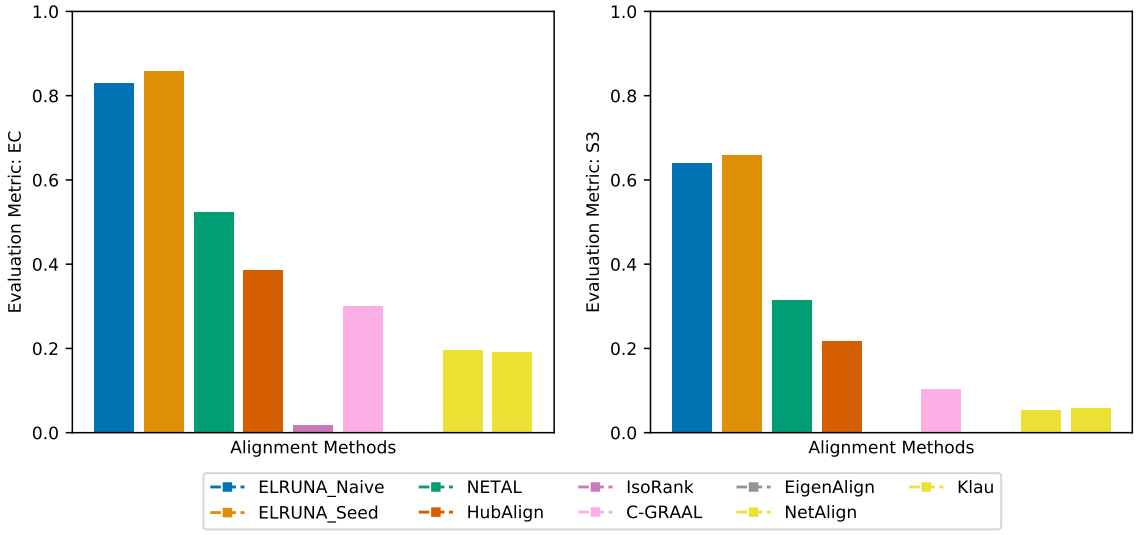

Figure 5.15: Alignment quality comparison on `facebook` networks

Results. `ELRUNA` outperforms all other baselines on `DBLP`, `digg` and `facebook` networks. For the `dblp` networks, even thought the EC value of `NETAL` is close to `ELRUNA` (differ by 5.7734%), the difference between their S^3 score is more significant, with `ELRUNA` surpassing `NETAL` by 18.7728%. For the `digg` and `facebook` networks, `ELRUNA_Naive` achieves a 75.88% and a 58.45 % improvement of the EC score over `NETAL`, respectively. As of S^3, `ELRUNA_Naive` achieves and a 139.65% and a 103.8% increase over `NETAL`. At the same time, the EC produced by `ELRUNA_Naive` are 2 to 11 times higher than other baselines. `ELRUNA_Seed` outperforms `ELRUNA_Naive` (and therefore all baselines) with improvement of EC up to 2.76% and improvement of S^3 up to 2.1% over `ELRUNA_Naive`.

This experiment further demonstrates the superiority of `ELRUNA` in identifying the underlying similar topology of networks and discovering correspondences of nodes.

5.3 Quality-speed trade-off and Scalability

As shown in the previous section, `ELRUNA` significantly outperforms other methods in terms of alignment quality. In this section, we study the quality-speed trade-off of `ELRUNA` against baselines. Then, we evaluate the scalability of `ELRUNA`.

5.3.1 Quality-speed trade-off

We first evaluate the trade off by running each algorithm on `bio_1`, `bio_2`, `erdos`, `retweet` and `social` networks under the highest noise levels ($p = 0.25$). That is, we align each G_1 with its corresponding $G_2^{(0.25)}$, then record the running time and alignment quality of each algorithm. In addition, we perform the same experiment on two pairs of networks from the homogeneous testing case: `digg` and `facebook` networks.

We measure the running time (in seconds) and alignment quality (EC and S^3) of `ELRUNA_Naive` and `ELRUNA_Seed` with incremental number of iterations. That it, we run the algorithm several times until convergence, each time with one additional iteration. For all other methods, we do not perform this incremental-iteration approach because they either cannot specify the number of iterations or the alignment quality are significantly lower than `ELRUNA`. The results are shown in Figure 5.16 to 5.22. For clarification, each dot on the `ELRUNA_Naive` and `ELRUNA_Seed` lines is one measurement after the termination of the algorithm under a particular number of iterations. Two adjacent dots are two measurements that differ by one additional iteration.

Note that running time of `C-GRALL` and `EigenAlign` are not included because they either have crashed (as described in the previous section) or both ran several hours which are not comparable to other methods. In additionally, the running time of `REGAL` are not included for `digg` and `facebook` networks because the implementation of `REGAL` does not support alignment between networks with different sizes.

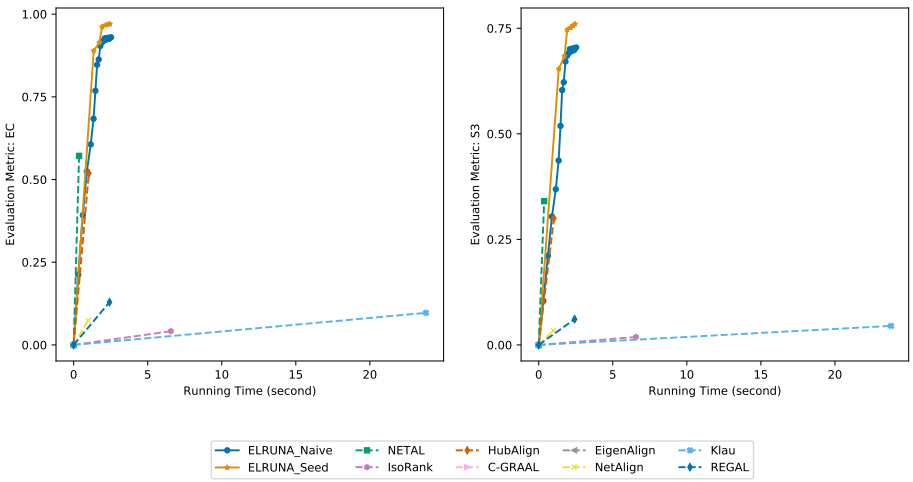

Figure 5.16: Quality-time comparison on `bio_1` networks. The last marker indicates the termination of the algorithm at which the algorithm is not improving any more. Each intermediate point indicates the termination of the algorithm at a particular iteration before convergence.

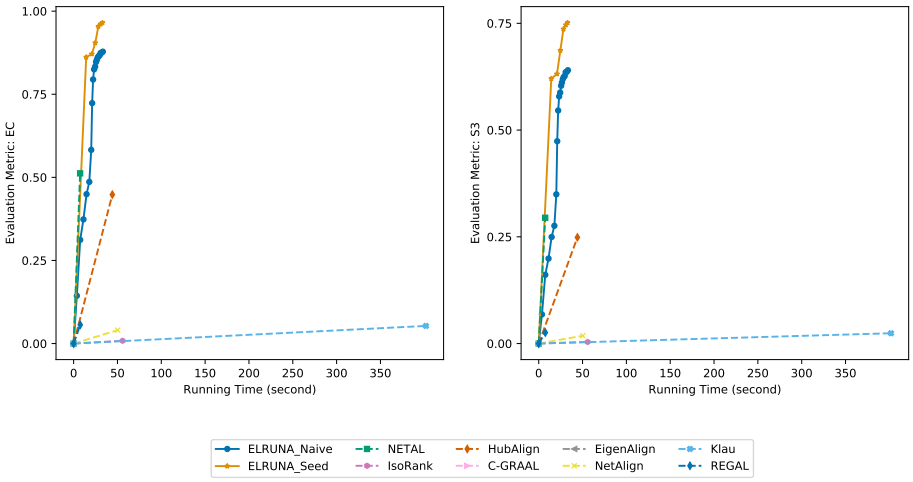

Figure 5.17: Quality-time comparison on `bio_2` networks. The last marker indicates the termination of the algorithm at which the algorithm is not improving any more. Each intermediate point indicates the termination of the algorithm at a particular iteration before convergence.

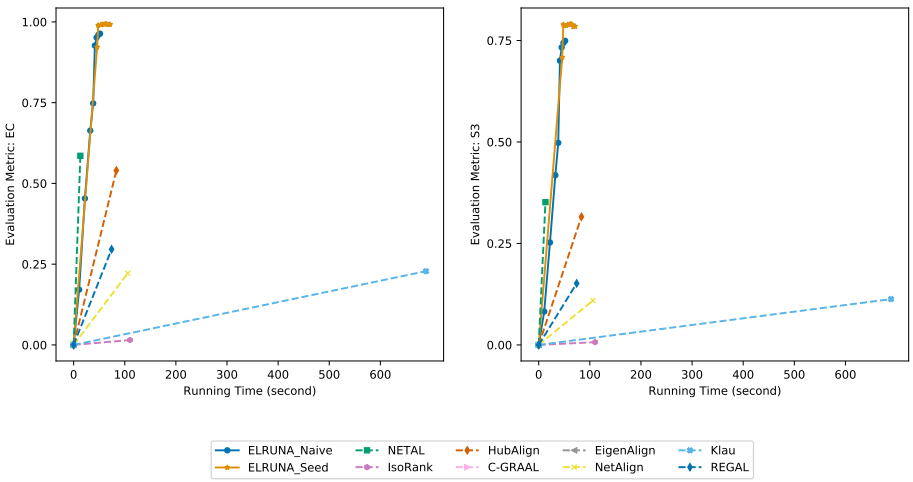

Figure 5.18: Quality-time comparison on `erdos` networks. The last marker indicates the termination of the algorithm at which the algorithm is not improving any more. Each intermediate point indicates the termination of the algorithm at a particular iteration before convergence.

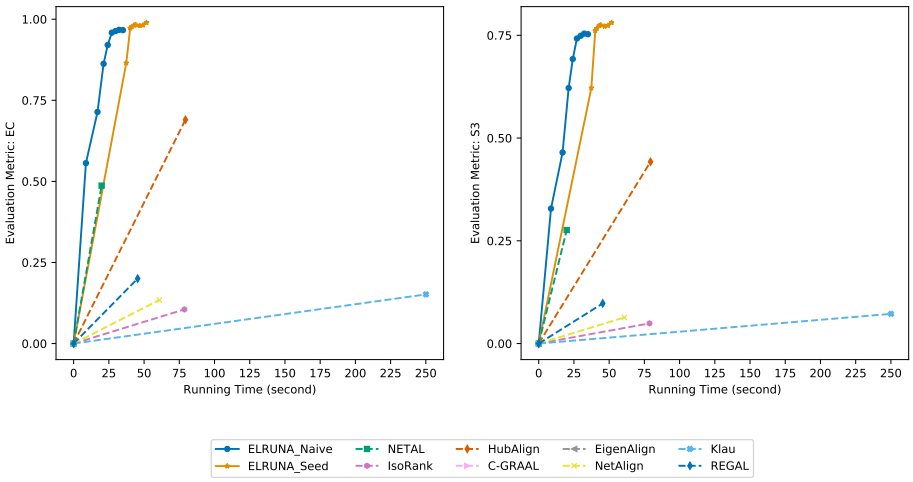

Figure 5.19: Quality-time comparison on `retweet_1` networks. The last marker indicates the termination of the algorithm at which the algorithm is not improving any more. Each intermediate point indicates the termination of the algorithm at a particular iteration before convergence.

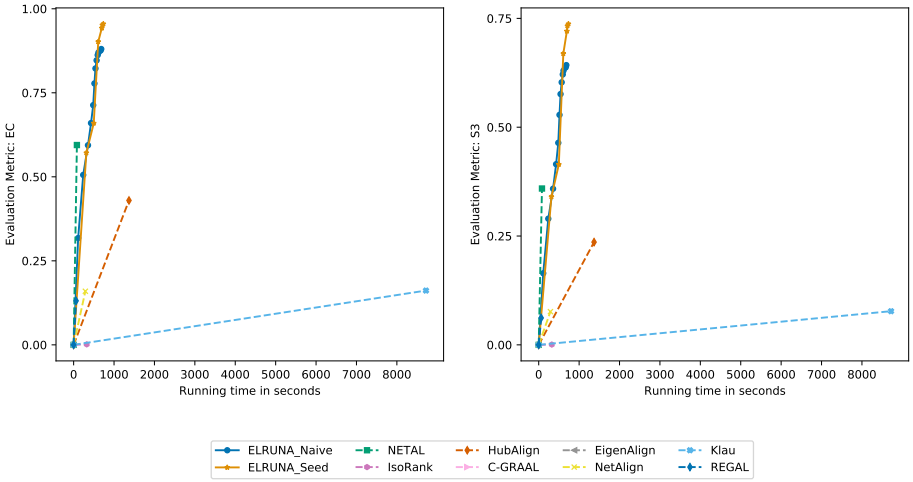

Figure 5.20: Quality-time comparison on `social` networks. The last marker indicates the termination of the algorithm at which the algorithm is not improving any more. Each intermediate point indicates the termination of the algorithm at a particular iteration before convergence.

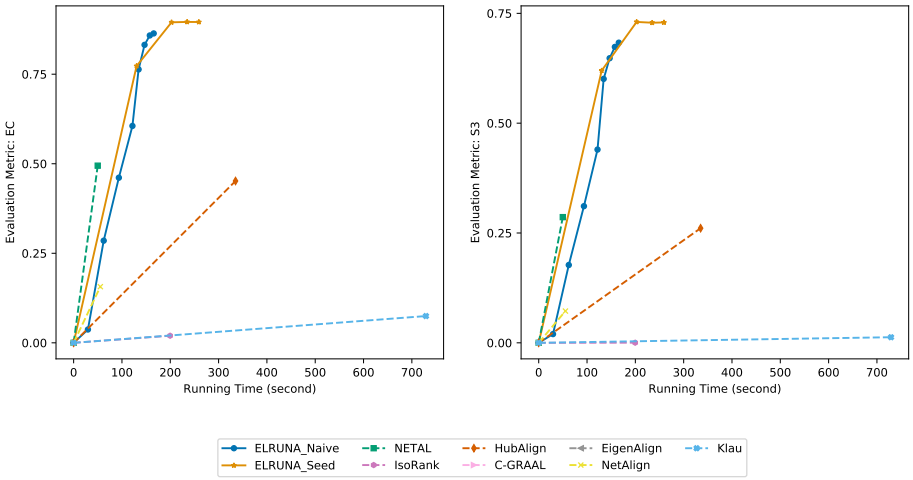

Figure 5.21: Quality-time comparison on `digg` networks. The last marker indicates the termination of the algorithm at which the algorithm is not improving any more. Each intermediate point indicates the termination of the algorithm at a particular iteration before convergence.

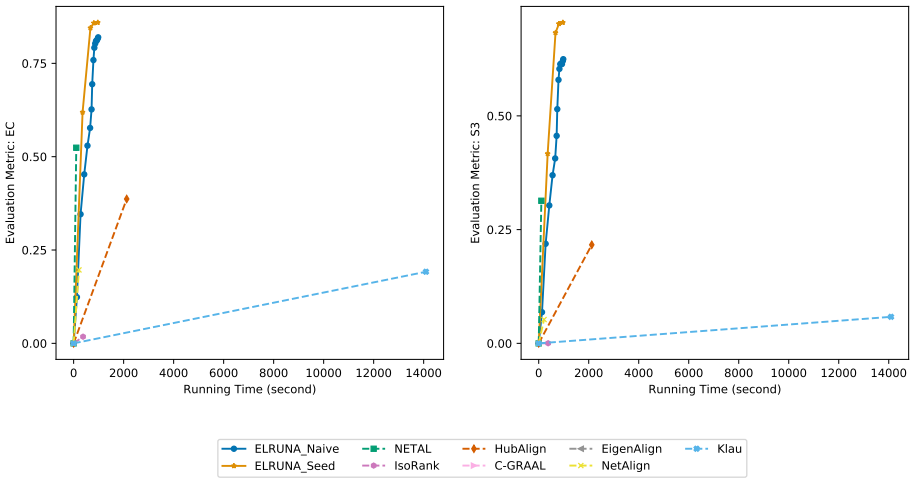

Figure 5.22: Quality-time comparison on `facebook` networks. The last marker indicates the termination of the algorithm at which the algorithm is not improving any more. Each intermediate point indicates the termination of the algorithm at a particular iteration before convergence.

Result. As we have observed, in comparison with `Klau` and `HubAlign`, both versions of `ELRUNA` achieve significantly better alignment results with lower running time. Moreover, `ELRUNA_Naive` and `ELRUNA_Seed` always have intermediate states (at some kth iteration) which have the similar or lower running time than `Netalign`, `REGAL` and `IsorRank`, but produce much better results.

As of `NETAL`, we observe that `ELRUNA_Naive` always has an intermediate state with similar alignment quality and slightly higher running time. Also, `ELRUNA_Seed` always has an intermediate state with better alignment quality and slightly higher running time than `NETAL`. However, both proposed methods can further improve the alignment quality greatly beyond the intermediate state, whereas `NETAL` and other baseline cannot. In addition, as the two proposed algorithms proceed, each succeeding iteration always takes lower time than the previous because the similarities are more defined after each iteration.

We observe that `ELRUNA_Seed` usually takes less iterations and longer running time to converge than `ELRUNA_Naive`. The is expected because the `seed-and-extend` alignment method is more computationally expensive than the `naive` alignment method.

5.3.2 Scalability

We evaluate the scalability of `ELRUNA_Naive` and `ELRUNA_Seed` by running them on networks from the *self-alignment without and under noise* testing case with no noisy edges. That is, we align G_1 with $G_2^{(0)}$. The result is shown in Figure 5.23.

Results. We observe that the running time of both versions of `ELRUNA` is quadratic with respect to the number of nodes in networks. The observation is also consistent with the running time analysis shown in the section 3.3.

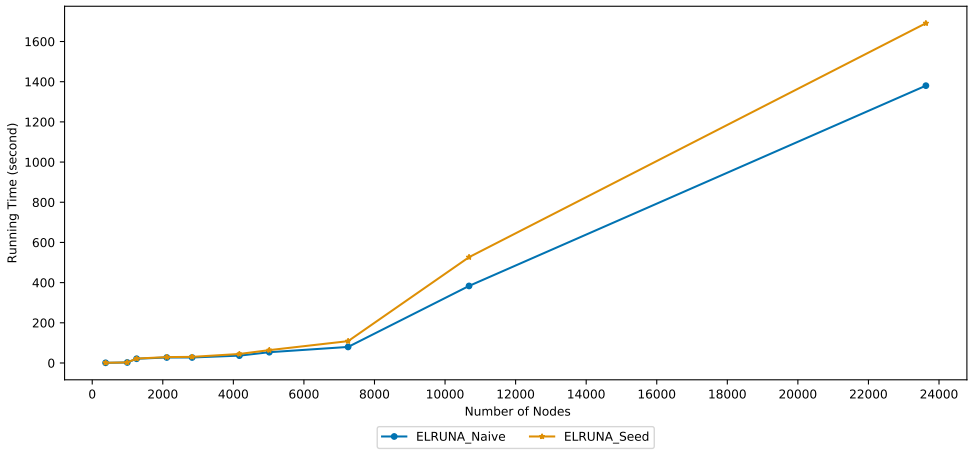

Figure 5.23: Scalability of `ELRUNA`

5.3.3 Evaluation of `RAWSEM`

In this section, we compare the performance of the proposed `RAWSEM` against the `baseline local search`. To evaluate their efficiency and effectiveness, we apply both methods as the post-processing steps for `ELRUNA_Naive`. We align real-world networks from the *self-alignment without and under noise* testing case under the highest noise level. That is, G_1 is aligned with $G_2^{(0.25)}$. For each method, we run it 20 times on each pair of networks and measure its average running time, the average number of iterations to reach a optima, and the average improved alignment quality. The results are summarized in Table 5.3 and 5.4.

Table 5.3: RAWSEM vs Baseline on the first 8 networks

	co-autho_1		bio_1	
	RAWSEM	Baseline	RAWSEM	Baseline
# of iterations	1,301	28,091	1,817	32,274
Time (in seconds)	0.093	2.052	0.095	2.327
Improved EC	2.3%	0.982%	3.231%	1.073%
Improved S^3	4.91%	1.175%	5.193%	1.91%
	econ		router	
	RAWSEM	Baseline	RAWSEM	Baseline
# of iterations	1,580	50,000	798	42,367
Time (in seconds)	0.11	3.324	0.077	3.049
Improved EC	0.506%	0%	1.406%	0.441%
Improved S^3	0.724%	0%	2.991%	0.892%
	bio_2		retweet_1	
	RAWSEM	Baseline	RAWSEM	Baseline
# of iterations	3,069	61,290	4,415	60,272
Time (in seconds)	0.191	4.48	0.217	4.411
Improved EC	5.477%	0.61%	1.241%	0.392%
Improved S^3	7.017%	1.326%	2.019%	0.673%
	erods		retweet_2	
	RAWSEM	Baseline	RAWSEM	Baseline
# of iterations	4,701	88,221	3,320	72,392
Time (in seconds)	0.323	6.25	0.204	5.19
Improved EC	2.91%	0.31%	5.72%	0.437%
Improved S^3	4.801%	0.781%	10.08%	0.901%

Table 5.4: RAWSEM vs Baseline on the last 2 networks

	social		google+	
	RAWSEM	Baseline	RAWSEM	Baseline
# of iterations	7,701	100,297	11,928	152,116
Time (in seconds)	1.953	7.855	3.04	10.238
Improved EC	4.903%	0.631%	3.29%	0.723%
Improved S^3	9.29%	1.01%	7.81%	1.59%

Results. Clearly, RAWSEM outperforms the Baseline local search in terms of the efficiency and effectiveness. In particular, RAWSEM achieves an up to 13 times and 11 times increase of EC and S^3, respectively, over the Baseline local search. Additionally, the number of iterations of Baseline local search are orders of magnitude lager than that of RAWSEM. This experiment shows that RAWSEM can boost the alignment quality within seconds which makes it a great candidate for a post-processing step of network alignment algorithms.

We observe that RAWSEM does not always raise the objective to the global optima. This suggest us to combine our selection rule with different local search methods, such as simulated annealing, to further enhance its performance. This is a further direction.

Chapter 6

Conclusion and future work

In my thesis, we propose ELRUNA, an iterative network alignment algorithm based on the elimination rules. We also introduce RAWSEM, a novel random-walk based selection rule for local search scheme which decreases the number of iterations it takes to reach local/global optimum. We conducted extensive experimental and demonstrate the superiority of ELRUNA and RAWSEM over eight state-of-the-art baselines. For the future works, we aim to (1) Imporve the running time of ELRUNA; (2) improve the performance of ELRUNA on aligning regular graphs; (3) Extend ELRUNA on aligning dynamic and attributed networks; (4) Develop advanced local search schemes to further reduce the number of iterations; (5) Develop hybrid quantum-classical network alignment algorithms.

Appendices

Appendix A Alignment Between Heterogeneous Networks

We compare ELRUNA to baselines on 5 pairs of networks where each pair consists of two networks from different domains. The first 3 pairs are social networks collected by [34], and the other two pairs are biological networks [27]. The details of networks are listed in Table 1.

Table 1: Datasets for test case: *Alignment between heterogeneous networks*

Doamin	n	m
Offline vs Online	1,118 vs 3,906	1,511 vs 8,164
Flickr vs Lastfm	12,974 vs 15,436	16,149 vs 16,319
Flickr vs Myspace	6,714 vs 10,693	7,333 vs 10,686
Syne vs Yeast	1,837 vs 1,994	3,062 vs 15,819
Ecoli vs Yeast	1,274 vs 1,994	3,124 vs 15,819

In this experiment, we demonstrate the results without local search, i.e., we demonstrate how ELRUNA outperforms the state-of-the-art methods. Results are shown in Figure 1 to 3. Because each pair of networks do not have underlying isomorphic subgraphs, the optimal objective is not known and the highest EC is not 1.

It is worth noting that each pair of networks in this testing case do not have structural similar underlying subgraphs. In fact, their topology could be very distinct from each other, given their different domains. Also, as we stated in the beginning of the experiment section, this comparison scenario is used by attributed network alignment algorithms. Therefore, **it is not exactly what we solve with our formulation**. However, we still show the results because of the importance of this task.

Results. We observe that ELRUAN_Naive did not perform well in this testing case which is due to the naive alignment method that it uses. As of ELRUAN_Seed, it achieve a 7.85% and a 11.41% increase in EC over and HubAlign and NETAL, respectively, for the pairs offline and online networks. As of other pairs of networks, the ELRUAN_Seed's improvements of EC HubAlign and NETAL are statistically insignificant. The results suggest

us to extend ELRUNA to an attributed version to perform better under this testing case. This will be a futuer direction.

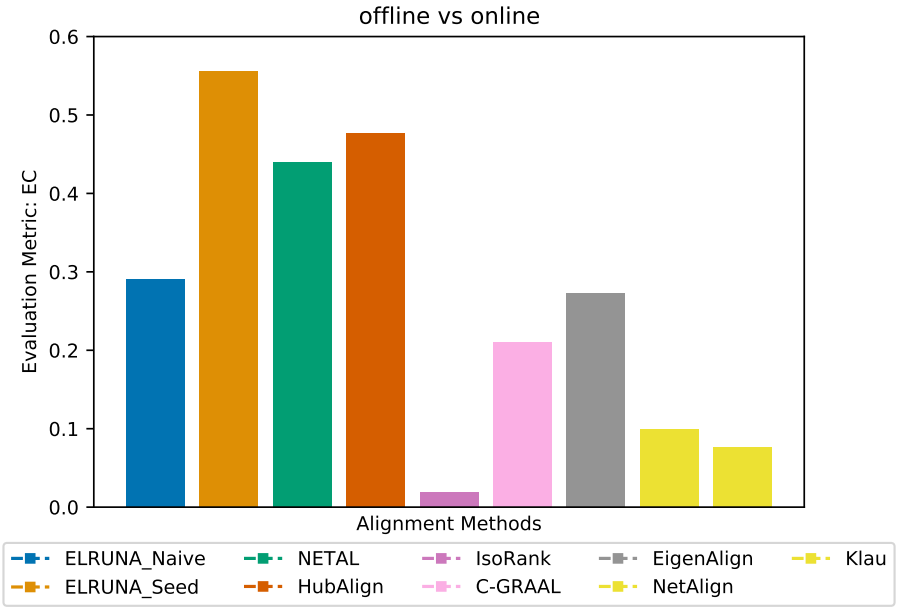

Figure 1: offline vs online networks

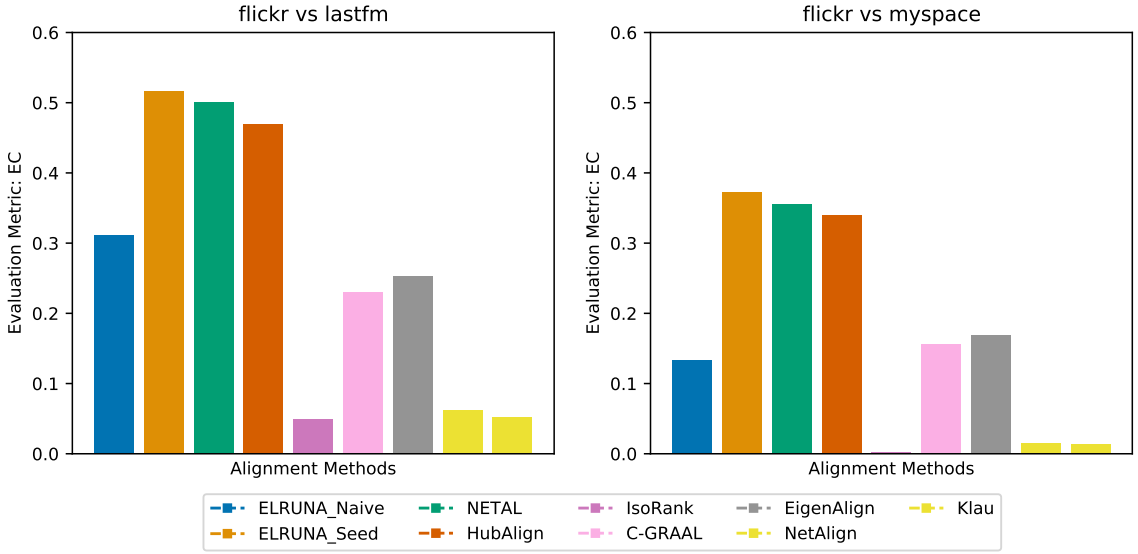

Figure 2: Flickr vs Lastfm and Flickr vs Myspace networks

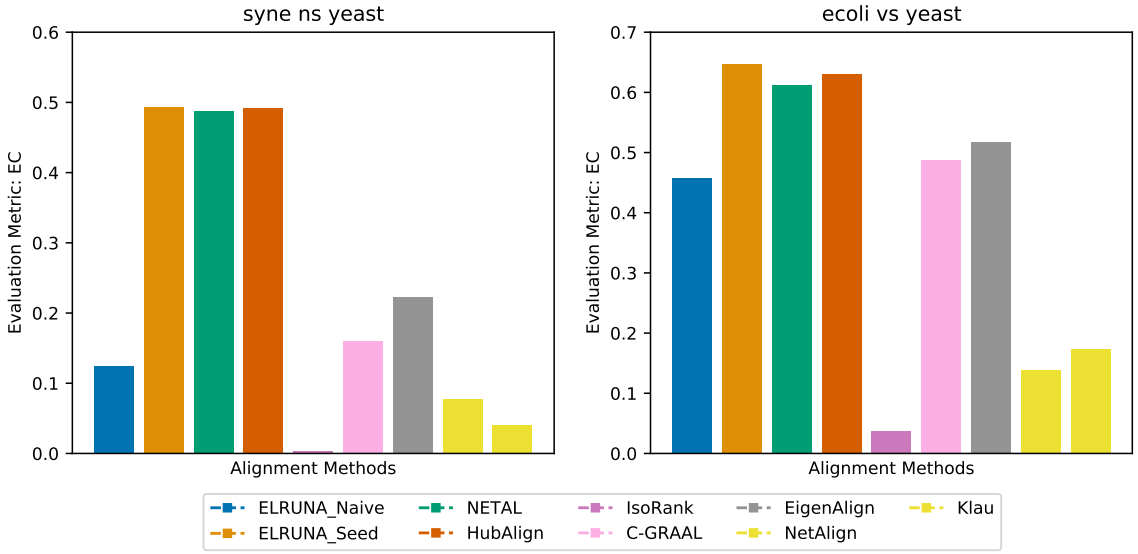

Figure 3: `Syne` vs `Yeast` and `Ecoli` vs `Yeast` networks

Bibliography

[1] A. Barabási and R. Albert. Emergence of scaling in random networks. *science*, 286(5439):509–512, 1999.

[2] M. Bayati, M. Gerritsen, D. Gleich, A. Saberi, and Y. Wang. Algorithms for large, sparse network alignment problems. In *2009 Ninth IEEE International Conference on Data Mining*, pages 705–710, 2009.

[3] Marián Boguná, Romualdo Pastor-Satorras, Albert Díaz-Guilera, and Alex Arenas. Models of social networks based on social distance attachment. *Physical review E*, 70(5):056122, 2004.

[4] Munmun De Choudhury, Hari Sundaram, Ajita John, and Dorée Duncan Seligmann. Social synchrony: Predicting mimicry of user actions in online social media. In *2009 International conference on computational science and engineering*, volume 4, pages 151–158. IEEE, 2009.

[5] Boxin Du, Si Zhang, Nan Cao, and Hanghang Tong. First: Fast interactive attributed subgraph matching. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1447–1456, 2017.

[6] Ran Duan and Seth Pettie. Linear-time approximation for maximum weight matching. *Journal of the ACM (JACM)*, 61(1):1–23, 2014.

[7] Soheil Feizi, Gerald Quon, Mariana Mendoza, Muriel Medard, Manolis Kellis, and Ali Jadbabaie. Spectral alignment of graphs. *IEEE Transactions on Network Science and Engineering*, 2019.

[8] P. Hiram Guzzi and T. Milenković. Survey of local and global biological network alignment: the need to reconcile the two sides of the same coin. *Briefings in bioinformatics*, 19(3):472–481, 2017.

[9] S. Hashemifar, J. Ma, H. Naveed, S. Canzar, and J. Xu. Modulealign: module-based global alignment of protein–protein interaction networks. *Bioinformatics*, 32(17):i658–i664, 2016.

[10] S. Hashemifar and J. Xu. Hubalign: an accurate and efficient method for global alignment of protein–protein interaction networks. *Bioinformatics*, 30(17):i438–i444, 2014.

[11] Mark Heimann, Wei Lee, Shengjie Pan, Kuan-Yu Chen, and Danai Koutra. Hashalign: Hash-based alignment of multiple graphs. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pages 726–739. Springer, 2018.

[12] Mark Heimann, Haoming Shen, Tara Safavi, and Danai Koutra. Regal: Representation learning-based graph alignment. In *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*, pages 117–126, 2018.

[13] P. Holme and B. Kim. Growing scale-free networks with tunable clustering. *Physical review E*, 65(2):026107, 2002.

[14] G W. Klau. A new graph-based method for pairwise global network alignment. *BMC bioinformatics*, 10(1):S59, 2009.

[15] D. Koutra, H. Tong, and D. Lubensky. Big-align: Fast bipartite graph alignment. In *2013 IEEE 13th ICDM*, pages 389–398, 2013.

[16] Danai Koutra and Christos Faloutsos. Individual and collective graph mining: principles, algorithms, and applications. *Synthesis Lectures on Data Mining and Knowledge Discovery*, 9(2):1–206, 2017.

[17] Jure Leskovec and Julian J Mcauley. Learning to discover social circles in ego networks. In *Advances in neural information processing systems*, pages 539–547, 2012.

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

[19] Yangwei Liu, Hu Ding, Danyang Chen, and Jinhui Xu. Novel geometric approach for global alignment of ppi networks. In *Thirty-First AAAI Conference on Artificial Intelligence*, 2017.

[20] Vesna Memišević and Nataša Pržulj. C-graal: C ommon-neighbors-based global gra ph al ignment of biological networks. *Integrative Biology*, 4(7):734–743, 2012.

[21] B. Neyshabur, A. Khadem, S. Hashemifar, and S. Arab. Netal: a new graph-based method for global alignment of protein–protein interaction networks. *Bioinformatics*, 29(13):1654–1662, 2013.

[22] Lawrence Page, Sergey Brin, Rajeev Motwani, and Terry Winograd. The pagerank citation ranking: Bringing order to the web. Technical report, Stanford InfoLab, 1999.

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

[24] Dorit Ron, Ilya Safro, and Achi Brandt. Relaxation-based coarsening and multiscale graph organization. *Multiscale Modeling & Simulation*, 9(1):407–423, 2011.

[25] Ryan A. Rossi and Nesreen K. Ahmed. An interactive data repository with visual analytics. *SIGKDD Explor.*, 17(2):37–41, 2016.

[26] Ilya Safro, Peter Sanders, and Christian Schulz. Advanced coarsening schemes for graph partitioning. *Journal of Experimental Algorithmics (JEA)*, 19:1–24, 2015.

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

[28] R. Shaydulin, H. Ushijima-Mwesigwa, I. Safro, S. Mniszewski, and Y. Alexeev. Community detection across emerging quantum architectures. *arXiv preprint arXiv:1810.07765*, 2018.

[29] R. Singh, J. Xu, and B. Berger. Global alignment of multiple protein interaction networks with application to functional orthology detection. *Proceedings of the National Academy of Sciences*, 105(35):12763–12768, 2008.

[30] Vipin Vijayan, Dominic Critchlow, and Tijana Milenković. Alignment of dynamic networks. *Bioinformatics*, 33(14):i180–i189, 2017.

[31] Bimal Viswanath, Alan Mislove, Meeyoung Cha, and Krishna P Gummadi. On the evolution of user interaction in facebook. In *Proceedings of the 2nd ACM workshop on Online social networks*, pages 37–42, 2009.

[32] J. Vogelstein, J. Conroy, V. Lyzinski, L. Podrazik, S. Kratzer, E. Harley, D. Fishkind, R. Vogelstein, and C. Priebe. Fast approximate quadratic programming for graph matching. *PLOS one*, 10(4):e0121002, 2015.

[33] Abdurrahman Yasar and Ümit V Çatalyürek. An iterative global structure-assisted labeled network aligner. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 2614–2623, 2018.

[34] S. Zhang and H. Tong. Final: Fast attributed network alignment. In *Proceedings of the 22nd ACM SIGKDD*, pages 1345–1354. ACM, 2016.

[35] S. Zhang, H. Tong, R. Maciejewski, and T. Eliassi-Rad. Multilevel network alignment. In *WWW*, pages 2344–2354. ACM, 2019.