# Spectral Graph Correspondences

Judith Franziska Hermanns

PhD Dissertation

Department of Computer Science
Aarhus University
Denmark

# Spectral Graph Correspondences

A Dissertation
Presented to the Faculty of Natural Sciences
of Aarhus University
in Partial Fulfillment of the Requirements
for the PhD Degree

by
Judith Franziska Hermanns
August 2, 2022

# Abstract

Graph correspondence problems occur whenever we aim at establishing correspondences between nodes or parts of two or more graphs. In this thesis, we focus on two correspondence problems: graph alignment and subgraph localization. In graph alignment, nodes of a graph $G$ are aligned to their counterparts in $G'$. In subgraph localization, the problem is to localize a query graph $Q$ in a larger graph $G$. Due to their closeness to graph isomorphism problems, both tasks are deemed computationally challenging. In this thesis we consider these problems on undirected, unattributed graphs. Given that in these cases only the structure of the graph is available, we investigate how these problems can be tackled by spectral graph theory. Spectral graph theory deals with the question which properties of the graph can be inferred from the spectrum of the graph. The spectrum of the graph are the eigenvalues of the graph laplacian. The laplacian is a graph descriptor comprising all structural information of a graph.

In a first contribution, we develop the spectral graph alignment method GRASP by framing the problem of aligning nodes as a problem of mapping node-specific functions across graphs. We establish the notion of unrestricted graph alignment as graph alignment which only uses the graph structure as an input. We evaluate our approach against other unrestricted graph alignment methods and find that it outperforms other scalable methods. In a second contribution, we structure the previously unstructured field of unrestricted graph alignment algorithms by developing an evaluation framework together with benchmark datasets and we evaluate algorithms previously not compared against each other on real and synthetic data. In a third contribution, we tackle the problem of subgraph localization by aligning the spectra of the query graph $Q$ and the larger graph $G$. As subgraph localization with only structural information on large graphs has not been convincingly tackled yet, we evaluate our method against a state-of-the-art graph alignment algorithm and show that the former is superior in localization quality.

# Resumé

Graf korrespondanceproblemer forekommer, når vi sigter mod at etablere overensstemmelse mellem knuder eller dele af to eller flere grafer. I denne afhandling fokuserer vi på to korrespondanceproblemer: afstemning af grafer og lokalisering af delgrafer. I det første problem afstemmes knuder i en graf $G$ til deres modstykker i graf $G'$, og i det andet problem lokaliseres en forespurgt delgraf $Q$ i in større graf $G$. Begge opgaver anses for at være beregningsmæssigt udfordrende grundet deres lighed med graf-isomorfi-problemer. I denne afhandling undersøger vi disse to problemer på ikke-rettede og uden-attributter grafer. Da det i disse tilfælde kun er grafernes struktur, som er tilgængelig, undersøger vi, hvordan disse problemer kan løses ved hjælp af spektral grafteori. Spektral grafteori beskæftiger sig med hvilke egenskaber ved grafen, der kan udledes af grafens spektrum. Grafens spektrum er egenværdierne for grafen Laplacian. Laplacian er en grafbeskrivelse, der omfatter al strukturel information ved en graf.

I vores første bidrag udvikler vi den spektrale graf metode GRASP til afstemning af grafer ved at formulere problemet med at lokalisere knuder som et problem med at kortlægge knude-specifikke funktioner på tværs af grafer. Vi fastsætter begrebet ubegrænset graf afstemning til kun at benytte grafstrukturen som input. Vi evaluerer vores fremgangsmåde i forhold til andre ubegrænsede graf afstemnings metoder og klarer os godt inden for skalerbare metoder. I vores andet bidrag strukturerer vi det hidtil ustrukturerede felt af ubegrænsede graf afstemnings algoritmer ved at udvikle en evalueringsramme, et benchmark-datasæt, samt ved at evaluere algoritmer, der ikke tidligere er blevet sammenlignet med hinanden på både reelle og syntetiske data. I vores tredje bidrag tager vi fat på problemet med lokalisering af delgrafer ved at tilpasse spektrene for forespørgselsgrafen $Q$ og den større graf $G$. Da lokalisering af delgrafer med kun strukturelle oplysninger fra store grafer endnu ikke er blevet behandlet på overbevisende vis, evaluerer vi i forhold til en af feltets førende graf afstemnings algoritme. Vi overgår dens lokaliserings evne.

# Contents

**Abstract** **i**

**Resumé** **iii**

**Contents** **v**

**List of Figures** **vii**

**List of Tables** **x**

**I Project Overview** **1**

**1 Introduction** **3**

1.1 Challenges in solving graph correspondence problems . . . . . . . . 4

1.2 Contributions . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 5

1.3 Outline . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 7

**2 Fundamentals** **9**

2.1 Eigenvectors and Eigenvalues . . . . . . . . . . . . . . . . . . . . . 9

2.2 Graphs . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 10

2.3 Graph Laplacian . . . . . . . . . . . . . . . . . . . . . . . . . . . 12

2.4 Graph Alignment . . . . . . . . . . . . . . . . . . . . . . . . . . . 14

2.5 Subgraph Localization . . . . . . . . . . . . . . . . . . . . . . . . 16

**3 Related Work** **19**

3.1 Graph Alignment . . . . . . . . . . . . . . . . . . . . . . . . . . . 19

3.2 Shape Matching . . . . . . . . . . . . . . . . . . . . . . . . . . . . 22

3.3 Subgraph Localization . . . . . . . . . . . . . . . . . . . . . . . . 22

**II Publications** **25**

**4 Spectral Graph Alignment** **27**
4.1 Abstract . . . 27
4.2 Introduction . . . 28
4.3 Background and Problem . . . 31
4.4 Solution . . . 32
4.5 Boosting GRASP . . . 39
4.6 Experiments . . . 43
4.7 Conclusion . . . 53

**5 Evaluation of Unrestricted Graph Alignment Algorithms** **55**
5.1 Abstract . . . 55
5.2 Introduction . . . 56
5.3 Preliminaries . . . 58
5.4 Alignment Algorithms . . . 58
5.5 Excluded Algorithms . . . 65
5.6 Experimental Setup . . . 66
5.7 Results . . . 70
5.8 Conclusion . . . 87

**6 Spectral Subgraph Localization** **89**
6.1 Abstract . . . 89
6.2 Introduction . . . 89
6.3 Subgraph localization . . . 91
6.4 Spectral Subgraph localization . . . 92
6.5 Experiments . . . 99
6.6 Conclusion . . . 104

**7 Conclusion** **107**
7.1 Outcome . . . 107
7.2 Future Work . . . 108

**Bibliography** **111**

# List of Figures

4.1 With a few removed edges, REGAL [48], an alignment method based on *local* features, fails to correctly align the distorted Karate club graph to the original; GRASP identifies most of nodes (correctly aligned nodes in green). . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 29

4.2 We remove **red** edges from the **green** graph to obtain the **blue** graph. Eigenvalues interlace (a); eigenvectors $\phi_1, \phi_2, \phi_3$ for **green** and $\psi_1, \psi_2, \psi_3$ for **blue** highlight common structures. . . . . . . . . . . . . . . . . . . . . 35

4.3 An example of the voting heuristic in action; GRASP runs four times, adding one eigenvector in each iteration, from $\phi_1$ to $\phi_4$. We omit eigenvectors $\psi_1, \ldots, \psi_4$ for the sake of readability. Each set of eigenvectors returns a matching from nodes $a, b, c, d$ to $a', b', c', d'$; on node $a$ the match $a$–$a'$ is selected by 3 out of 4 eigenvector sets and chosen as the output match. 42

4.4 Accuracy on the Facebook and Arenas graphs for different numbers of corresponding functions $q$ and different numbers of eigenvectors $k$ for noise levels 5%, 10% and 15%. . . . . . . . . . . . . . . . . . . . . . . . . 45

4.5 Accuracy of CONE-Align with different embeddings on the Arenas and Facebook datasets. . . . . . . . . . . . . . . . . . . . . . . . . . . . . 45

4.6 Accuracy of GRASP (ICP, voting and JV) with PPR, PR, HK and NetMF embeddings. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 46

4.7 Accuracy of GRASP-PPR with ICP and BA. . . . . . . . . . . . . . . . . 46

4.8 Accuracy of GRASP-PPR with ICP using the SortGreedy voting procedure with different intervals of $k$ . . . . . . . . . . . . . . . . . . . . . . . 47

4.9 Accuracy with different assignment algorithms. . . . . . . . . . . . . . . 47

4.10 Accuracy of nearest neighbor and JV matching algorithms. . . . . . . . 48

4.11 Accuracy comparison under synthetic noise. . . . . . . . . . . . . . . . . 48

4.12 Accuracy compared to REGAL on three real datasets. . . . . . . . . . . 49

4.13 Accuracy comparison on bipartite graphs. . . . . . . . . . . . . . . . . . 50

4.14 Runtime vs. number of nodes on four random graphs of increasing size. 51
4.15 Time for B-GRASP and CONE with NetMF on four datasets. . . . . . 51
4.16 Alignment time *excluding* precomputation time. . . . . . . . . . . . . . 52
4.17 Alignment time *including* precomputation time. . . . . . . . . . . . . . 52
4.18 Accuracy of GRASP-PPR and CONE with JV assignment on generated power law graphs. . . . . . . . . . . . . . . . . . . . . . . . . . . 53

5.1 Different assignment methods; solid lines: Arenas real data; dashed lines: Power-Law (PL) synthetic data. . . . . . . . . . . . . . . . . . . . . . . 71
5.2 Accuracy, $S^3$, and MNC for Erdős-Rényi (ER) random graphs; noise up to 5% and different noise types. . . . . . . . . . . . . . . . . . . . . . . . 73
5.3 Accuracy, $S^3$, and MNC for Barabasi-Albert (BA) synthetic graphs; noise up to 5% and different noise types. . . . . . . . . . . . . . . . . . . . . 74
5.4 Accuracy, $S^3$, and MNC for Watts-Strogats (WS) synthetic graphs; noise up to 5% and different noise types. . . . . . . . . . . . . . . . . . . . . 75
5.5 Accuracy, $S^3$, and MNC for Newman-Watts (NW) synthetic graphs; noise up to 5% and different noise types. . . . . . . . . . . . . . . . . . . . . 76
5.6 Accuracy, $S^3$, and MNC for Powerlaw (PL) synthetic graphs; noise up to 5% and different noise types. . . . . . . . . . . . . . . . . . . . . . . . 77
5.7 Accuracy on real graphs; noise up to 5%. . . . . . . . . . . . . . . . . . 79
5.8 Accuracy on real graphs; one-way noise up to 25%. . . . . . . . . . . . . 81
5.9 Time vs. accuracy on NetScience; marks show results with One-way noise in $[0.25, 0.2, 0.15, 0.1, 0.05, 0]$. . . . . . . . . . . . . . . . . . . . . 82
5.10 Accuracy, MNC and $S^3$ results for the real graphs *HighSchool*, *Voles* and *MultiMagna*. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 83
5.11 Time vs. # nodes; configuration model graphs. . . . . . . . . . . . . . . 84
5.12 Time vs. avg degree, uniform distr.; $2^{14}$ nodes. . . . . . . . . . . . . . . 84
5.13 Memory vs. # nodes; configuration model. . . . . . . . . . . . . . . . . 85
5.14 Memory vs. avg degree, uniform distr.; $2^{14}$ nodes. . . . . . . . . . . . . 85
5.15 Accuracy for 1% One-way noise on Newman–Watts–Strogatz synthetic graphs with 2000 nodes. . . . . . . . . . . . . . . . . . . . . . . . . . 86
5.16 Accuracy for 1% One-way noise on Newman–Watts–Strogatz synthetic graphs of increasing size. . . . . . . . . . . . . . . . . . . . . . . . . . 86

6.1 An instance of *subgraph localization* (left) and its solution (right); given graphs $Q$ and $G$, we compute an indicator function $\delta(v)$ that reveals the location of $Q$ in $G$. . . . . . . . . . . . . . . . . . . . . . . . . . . 90

6.2 Alignment of the spectrum $\lambda_Q$ of $Q$ and the part of the spectrum $\lambda(\mathscr{L} - \mathbf{E} + \mathrm{diag}(\mathbf{v}))$ of $G$ corresponding to $Q$ at 1, 6, and 18 iterations. As two spectra progressively converge, especially in smaller eigenvalues, $Q$ is correctly localized in $G$. . . . . . . . . . . . . . . . . . . . . . . . . . 98
6.3 Example SSL result: blue nodes show the ground-truth subgraph $Q$; red nodes are the remaining nodes of $G$; shaded nodes are classified wrongly; solid lines show edges correctly altered (green) or unaltered (black) by $E$; dotted lines show edges incorrectly unaltered (green) or altered (black). 98
6.4 Balanced Accuracy for variants of the objective function in Equation 6.11 with: (1) data term only; (2) data term and spectral regularizer; and (3) data term, spectral regularizer. and sphere constraint (SSL objective). . 101
6.5 Plots of $\mathbf{v}$ elements sorted in ascending order, and $\mathbf{v}$ mapped to $G$ for different designs of the objective function: (1) only with data term; (2) with data term and spectral regularizer; and (3) with data term, spectral regularizer, and sphere constraint; nodes in $|V_S|$ correspond to the 20 smallest entries in $\mathbf{v}$, lighter shades indicating smaller values; applying only the data term, the method fails to localize $G_S$; applying regularization renders the separation clearer; the addition of the sphere constraint induces a perfectly clear separation. . . . . . . . . . . . . . . . . . . . . . . . . . 102
6.6 Results on localizing query graphs of 20%, 30%, 40% of the **football** (upper row) and **highschool** (lower row) graphs vs. fraction of connecting edges between the query graph and rest of the parent graph; we start with a disconnected query graph and add connecting edges, until the original parent graph is reconstructed. . . . . . . . . . . . . . . . . . . . . . . . . 103
6.7 Alignment of the spectrum $\lambda_Q$ of $Q$ and the corresponding part of the spectrum $\lambda(\mathscr{L} - \mathbf{E} + \mathrm{diag}(\mathbf{v}))$ of $G$ after convergence, ground truth $V_S$ (blue) and $V \backslash V_S$ (white), and corresponding localization by SSL; while the spectra are perfectly aligned, the detected subgraph is not the ground truth. The depicted graph is a protein-protein interaction network from the D&D dataset [31]. . . . . . . . . . . . . . . . . . . . . . . . . . . 104
6.8 Alignment of the spectrum $\lambda_Q$ of $Q$ and the corresponding part of the spectrum $\lambda(\mathscr{L} - \mathbf{E} + \mathrm{diag}(\mathbf{v}))$ of $G$ after convergence, ground truth $V_S$ (blue) and $V \backslash V_S$ (white), and corresponding localization by SSL; while the spectra are well aligned, the detected subgraph is not the ground truth; the detected subgraph coincides with nodes that have degree 0 or 1. The depicted graph is **arenas** [73]. . . . . . . . . . . . . . . . . . . . . . . 104

# List of Tables

3.1 Related work in terms of present (✓) and absent (✗) properties: *supervised* methods require aligned nodes as input; *spectral* methods use the spectra of alignment matrices; IsoRank does not work on *plain* graph structures as it requires auxiliary node similarity input; FINAL requires node or edge attribute information to perform reasonably well [48]; *flexible* methods accommodate different alignment algorithms (e.g. bipartite matching, nearest neighbors). Among unrestricted methods (rows 2–7), LREA does not benefit from *offline precomputation*. GRASP naturally captures *multiscale* properties thanks to its spectral basis, while S-GWL coarsens the graph to progressively match structures at different scales. REGAL, CONE and GRASP are *modular*, hence allow boosting various parts of the algorithm. FINAL, LREA, REGAL, and GRASP can *scale*, running on graphs of more than 10 000 nodes in less than 1 hour. GRASP treats the problem as an alignment among *functions*, embracing a variety of node functions, including embeddings. . . . . . . . . . . . . . . . . 20

4.1 Datasets used in our evaluation, $|V|$ number of nodes, $|E|$ number of vertices, GT ground truth. . . . . . . . . . . . . . . . . . . . . . . . . . 43

4.2 Estimated power law exponents. . . . . . . . . . . . . . . . . . . . . . . 52

5.1 Algorithms considered in the experiments. . . . . . . . . . . . . . . . . 59

5.2 Information about the real graphs, number of nodes $n$, number of edges $m$, number of nodes left out of the largest connected component $\ell$, and network type. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 68

5.3 Summary results vs. graph model (first and second best method marked with 🏆), graph size and density. . . . . . . . . . . . . . . . . . . . . . 69

6.1 SSL hyperparameters and default values. . . . . . . . . . . . . . . . . 97

6.2 Graphs used in our evaluation: number of nodes $|V|$, number of vertices $|E|$, and graph type. . . . . . . . . . . . . . . . . . . . . . . . . . . 99

# List of Notation

| | |
|---|---|
| $b$ | A scalar |
| $\mathbf{b}$ | A vector |
| $\mathbf{B}$ | A matrix |
| $\mathbf{A}$ | Adjacency matrix |
| $\mathbf{D}$ | Degree matrix |
| $G = (V, E)$ | A graph with nodes $V$ and edges $E$ |
| $v$ | A node |
| $(v_i, v_j)$ | An edge between $v_i$ and $v_j$ |
| $\mathscr{L}$ | Laplacian |
| $\lambda$ | Eigenvalue |
| $\phi, \psi$ | Eigenvectors |

# Part I

# Project Overview

# Chapter 1

# Introduction

When multiple objects are set in relation to each other the resulting structure can often be modeled as a graph. A graph consists of a set of nodes and a set of edges connecting these nodes. Graph structures exist in all areas of science, technology, nature, art and social life. Scientists and artists form collaboration networks [93, 131], humans form social relationships and friendship networks [139], atoms connect as chemical compounds [119] and in smart homes, devices form a network [120].

Due to their flexibility and expressiveness, graphs can model these and many more networks. These graphs can contain valuable information, often hidden in the connectivity structure of a graph. The broad area of **Graph Mining** or **Graph Analysis** [3] focuses on the problem of extracting this knowledge from graphs.

Over the years, many methods concerning graph analysis have been developed, dealing with different tasks for knowledge extraction from graphs based on their characteristic structural properties [25]. In community detection [38, 149] and graph clustering [101, 125], a graph is partitioned into several subgraphs. While also node attributes are considered here, the connectivity structure plays an important role, as nodes that form communities i.e., are connected amongst each other in patterns that separate them from the rest of the graph are natural candidates for a cluster or a community. Another relevant field is the retrieval or detection of interesting subgraphs, for example clique detection [124] or frequent subgraph mining [56]. Again we have to consider the graph structure: Is a subgraph a clique? Is a specific structure frequent? The machine learning task of data classification also appears in graph analysis as node classification [115] or graph classification [154]. Naturally, the community structure of a graph has to be considered in these cases, as only considering node attributes would make it unnecessary to model data as a graph. In a collection of graphs instead it is important to understand whether two graphs are similar and, if possible, recover correspondences among nodes. This is of paramount importance in the analysis of proteins [67], the analysis of social networks [83] and in re-identification tasks in computer vision [153].

In recent years, spectral graph theory[23] has found its way into the field of graph analysis. Spectral graph theory investigates how knowledge about a graph can be gained from analyzing the spectrum of the Laplacian of a graph. The Laplacian of a graph is a graph descriptor, which is calculated from the adjacency matrix of a graph and its degree matrix. The powerful property of the spectrum of this Laplacian is the fact that structural properties can be read from it, for example the number of connected components [23] or the overall community structure of a graph [135]. This property makes it an ideal tool for graph analysis as a compact and expressive descriptor of the graph structure. There are approaches which already apply spectral graph theory to problems for example in graph similarity applications [135], graph clustering [125], and graph neural networks [29].

In this PhD thesis, we connect spectral graph theory and graph analysis further. The focus will be graph correspondences. Graph correspondence problems occur whenever we want to establish correspondences between nodes of two or more graphs or parts of two or more graphs. This can be a standalone problem, for example if we want to align the users of one social network to the corresponding users of a second social network [155]. Solving for graph correspondences can also be a preprocessing step. In cases where we aim at applying a graph analysis task to several related graphs, pre-aligning these graphs might make some recomputations unnecessary [128], because a known alignment results makes it possible to simply transfer results across graphs. However, while being an important task in graph analysis, establishing correspondences between graphs is also a challenging computational problem, as the most general problem falls into graph isomorphism or, in the case of graphs of different size, subgraph isomorphism [25].

## 1.1 Challenges in solving graph correspondence problems

Graph correspondence problems are closely related to the fundamental problem of graph isomorphism. Two graphs $G$ and $G'$ are isomorphic if there exists a bijective mapping between the nodes of $G$ and $G'$ such that edges are preserved, i.e. two nodes are neighbors in $G$ if and only if their counterparts in $G'$ are also neighbors [89]. Checking for graph isomorphism is in NP, while subgraph isomorphism is NP-hard [65].

This a challenge when trying to solve instances of this problem in graph analysis, where scalable solutions for large graphs and graph databases are required.

A further complication occurs with the fact that in graph analysis we frequently come across correspondence problems for non-isomorphic graphs. The graphs for which we want to establish correspondences might have an overall similar community structure, but differ in local structures. For example, two social networks might have the same users, but slightly different connections between them. However, we still want to establish correspondence between the users. In this case, how can we find these correspondences? In some cases, node attributes are available, e.g. usernames, but they can be noisy or wrong, so in this PhD thesis we will instead consider the base problem without attributes.

All in all, the central question of this PhD thesis is: How can we establish correspondences between graphs, solely based on their structure? In the next section, we outline our contributions in working on this question.

## 1.2 Contributions

We restrict this PhD thesis to two specific instances of graph correspondence problems: graph alignment and subgraph localization. In graph alignment, we assign each node of one graph to at least one node of a second graph [50]. In subgraph localization, we aim at assigning a smaller graph to a corresponding subgraph in a bigger graph.

As established in the previous section, we work with the purest version of the problem, i.e., the one in which we do not assume any additional information besides nodes and edges. This is an ideal application case for the previously discussed spectral graph theory, which deals exactly with the question of how to gain knowledge about a graph from its structure, distilled in its spectrum.

This thesis comprises three manuscripts, the contributions of which we will now discuss.

**Spectral Graph Alignment**. Graph alignment is a well-researched field, as graph alignment tasks appear in a broad variety of fields, from the alignment of social networks [155] to the alignment of biological networks [81] and to computer vision applications [55]. However, this diversity in application areas leads to the fact that many algorithms are extremely specialized and require domain-specific knowledge, like protein structures in protein-protein-interaction networks [62]. Others expect a set of ground truth alignments, as often seen in social network alignment [60]. We call graph alignment algorithms which do not require such additional information *unrestricted*. There are some of those [48, 72, 126], but their alignment quality is not convincing and none of them has connected graph alignment and spectral graph theory with a focus on the community structure of the graphs.

In our first project we deal with this unrestricted alignment case. We develop an approach which uses spectral graph theory in order to establish a mapping which enables the transfer of arbitrary functions across graphs [51]. We then use this mapping in order to map node-specific functions across graphs for the purpose of obtaining a final alignment. The latter step returns node embeddings for both graphs, which can be aligned to each other via simple linear assignment algorithms. Furthermore, we set our approach into context with other state-of-the-art embedding-based graph alignment algorithms, identifying a modular alignment framework which all of these algorithms follow. We develop further boosting techniques, which can be used to improve our own method GRASP, but can also be applied in any other modular graph alignment method.

**Evaluation of Algorithms for Unrestricted Graph Alignment**. In the course of our work on GRASP, we noticed the lack of a consistent evaluation methodology and benchmark datasets for unrestricted graph alignment. Furthermore, as the existing unrestricted graph alignment algorithms emerged from different fields, they were never all evaluated against each other. Thus we evaluate nine state-of-the-art unrestricted graph alignment methods [19, 48, 66, 72, 95, 126, 145, 146] thoroughly on both real and synthetic graphs, providing insight into advantages and disadvantages of each algorithm and providing a benchmark for future work in this area [127].

**Spectral Subgraph Localization.** In a third work, we tackle the problem of subgraph localization. Subgraph localization is a task that plays a role as a subtask in many graph correspondence problems, for example in subgraph queries [59], subgraph discovery [74] and subgraph isomorphism problems [34]. However, it is rarely tackled as a task on its own. Existing work can only deal with very small subgraphs [34], so there is a potential to contribute a scalable subgraph localization approach. Again we approach this problem from a spectral point of view. Here our approach is to align the spectra of graph and subgraph and in this way, obtain a localization of the subgraph in the larger graph [52].

The work conducted in this thesis has lead to five manuscripts. Two short papers are published at WAIM-APWeb 2021 [50] and CIKM 2021 [76]. We combined and extended them to a paper about GRASP, which is under review at TKDD [51]. Our work on evaluating graph alignment algorithms is accepted for publication at EDBT 2023 [127] and our work on subgraph alignment is under review for ICDE 2023 [52].

## 1.3 Outline

This thesis is organized as follows. In Chapter 2, we introduce the preliminaries necessary for understanding the subsequent chapters. Chapter 3 reviews existing work on graph correspondences and spectral methods. In Chapter 4, we present our work on spectral graph alignment and modular graph alignment algorithms, Chapter 5 is devoted to our work on structuring the unrestricted graph alignment landscape. In Chapter 6, we present our work on spectral subgraph localization. The thesis concludes with a summary of our findings and an outlook to future work in Chapter 7.

# Chapter 2

# Fundamentals

In this chapter, we present the preliminaries necessary for understanding our contributions.

## 2.1 Eigenvectors and Eigenvalues

A central element of this PhD thesis are *eigenvectors* and *eigenvalues*. In this section, we describe the mathematical foundations behind them which are necessary for the understanding of the remainder of this PhD thesis. The concepts presented here can be found in standard works on introduction to linear algebra, e.g. in [142].

An eigenvector $\mathbf{v} \in \mathbb{R}^n$ of a square matrix $\mathbf{A} \in \mathbb{R}^{n \times n}$ is a vector which changes only in length but not in direction when transformed by $\mathbf{A}$. The factor $\lambda \in \mathbb{R}$ by which this eigenvector is scaled is the eigenvalue associated with the eigenvector $\mathbf{v}$.

The following eigenvalue equation describes the interaction of eigenvector and eigenvalue:

$$\mathbf{A}\mathbf{v} = \lambda \mathbf{v} \tag{2.1}$$

The matrices of which we use eigenvalues and eigenvectors throughout this thesis have a special form: they are symmetric and positive semidefinite.

**Definition 1** (Symmetric matrix)**.** *A matrix $\mathbf{A} \in \mathbb{R}^{n \times n}$ is symmetric if $\mathbf{A}_{ij} = \mathbf{A}_{ji}$ for $i, j \in \{1, \ldots, n\}$.*

**Definition 2** (Positive semidefinite matrix)**.** *A matrix $\mathbf{A} \in \mathbb{R}^{n \times n}$ is positive-semidefinite iff $\mathbf{X}\mathbf{A}^\top \geq 0 \, \forall \, \mathbf{X} \in \mathbb{R}^n$.*

The $n$ eigenvalues of a positive semidefinite matrix are real and positive. The corresponding eigenvectors of a positive semidefinite matrix form an orthogonal basis. To understand this, we first define orthogonality between two vectors:

**Definition 3.** *Two vectors* $\mathbf{x}, \mathbf{y} \in \mathbb{R}^n$ *are orthogonal if for the dot product* $\mathbf{x} \cdot \mathbf{y} = \sum_{i=0}^{n} x_i y_i$ *it holds that* $\mathbf{x} \cdot \mathbf{y} = 0$.

A matrix is orthogonal if all of its columns are orthogonal to each other. An orthogonal basis of the vector space $\mathbb{R}^n$ is a set of vectors that can be linearly combined to any vector in $\mathbb{R}^n$. Formally, this is defined as follows:

**Definition 4** (Orthogonal basis)**.** *An orthogonal basis of* $\mathbb{R}^n$ *is a set of vectors* $\{\mathbf{v}_1, \dots, \mathbf{v}_n\}$ *with* $\mathbf{v}_i \in \mathbb{R}^n$ *such that for all* $x \in \mathbb{R}^n$ *there exist* $a_1, \dots, a_n \in \mathbb{R}$ *such that* $x = \sum a_i \mathbf{v}_i$.

**Eigendecomposition.** Let us go back to eigenvalues and eigenvectors. Every real symmetric matrix $\mathbf{A} \in \mathbb{R}^{n \times n}$ can be decomposed into a matrix $\Phi \in \mathbb{R}^{n \times n}$, which contains the eigenvectors of $\mathbf{A}$ as columns, and a matrix $\Lambda \in \mathbb{R}^{n \times n}$, a diagonal matrix, which contains the corresponding eigenvalues $\lambda_1, \dots, \lambda_n$ of $\mathbf{A}$ on its diagonal. $\mathbf{A}$ can then be written as:

$$\mathbf{A} = \Phi \Lambda \Phi^T \tag{2.2}$$

This is called an *eigendecomposition* of $\mathbf{A}$. As the matrices that are eligible for eigendecomposition in this thesis are real and positive semidefinite, it holds that $\lambda_1, \dots, \lambda_n \in \mathbb{R}_{\geq 0}$. The eigenvectors in $\Phi$ form an orthogonal basis.

We now have discussed how eigenvalues and eigenvectors are defined and know some properties important for this PhD thesis. In the next section, we introduce a second main element of this thesis: graphs.

## 2.2 Graphs

In this section, we define graphs and introduce basic notations as well as concepts that will become important in the rest of this thesis. The concepts presented here can be found in [4, 25].

### 2.2.1 Basic Definitions

An undirected graph $G = (V,E)$ is defined as a tuple of nodes $V$ and edges $E$ with $E \subseteq V \times V$. We call $n = |V|$ the size of the graph. Important concepts in graphs are neighbors and neighborhoods.

**Definition 5** (Neighborhood)**.** *In a graph $G = (V,E)$, a node $v_i$ is a neighbor or adjacent to $v_j$ if there is an edge $(v_i, v_j) \in E$. The neighborhood $N(v_i)$ of a node $v_i$ is the set of all neighboring nodes of $v_i$, thus $N(v_i) = \{v_j|(v_i, v_j) \in E\}$.*

The number of neighbors of a node $v_i$ is called the degree of $v_i$.

**Definition 6** (Degree)**.** *The degree $d(v_i)$ of a node $v_i, v_i \in E$ in a graph $G = (V,E)$ is defined as $|N(v_i)|$.*

A graph can be represented in matrix form by its adjacency matrix $\mathbf{A} \in \mathbb{R}^{n\times n}$. This is a binary matrix constructed as follows:

$$\mathbf{A}_{ij} = \begin{cases} 1 & \text{if } (v_i, v_j) \in E \\ 0 & \text{else} \end{cases} \tag{2.3}$$

From this matrix, we can calculate a degree matrix $\mathbf{D} \in \mathbb{R}^{n\times}$, which is a diagonal matrix containing on its diagonal the degrees of the nodes of $G$.

$$\mathbf{D}_{ii} = \sum_j \mathbf{A}_{ij} \tag{2.4}$$

After introducing these basic graph concepts, we now discuss PageRank [104], a method to determine the importance of a node in a graph.

#### PageRank (PR)

PageRank [104] was developed to make it possible for a search engine to evaluate the importance of web pages. The principle behind this method is very simple: web pages are considered important if other important web pages link to them. The scenario can be represented as a (directed) graph, where nodes are web pages and edges are links between web pages. The rank $r(v_i)$ of a node $v_i$ is then computed as:

$$r(v_i) = \sum_{(v_i,v_j)\in E} \frac{r(v_j)}{d(v_i)} \tag{2.5}$$

PageRank is based on the idea of a random surfer who randomly clicks from page to page. Since the web page graph is directed, a random surfer may get stuck in nodes with degree 1. For this reason, a damping factor $\alpha$ is introduced to model the probability of a random surfer either going to another linked page or jumping to a random page. The formula for the rank $r(v_i)$ changes as follows:

$$r(v_i) = (1-\alpha) \sum_{(v_i,v_j)\in E} \frac{r(v_j)}{d(v_i)} + \alpha \frac{1}{n} \tag{2.6}$$

This can be solved iteratively by initializing $r_0(v_i)$ as $r_0(v_i) = \frac{1}{n}$ and then updating $r(v_i)$ as

$$r_{t+1}(v_i) = (1-\alpha) \sum_{(v_i,v_j)\in E} \frac{r_t(v_j)}{d(v_i)} + \alpha \frac{1}{n} \tag{2.7}$$

until convergence.

**Personalized PageRank (PPR)**. PageRank can be personalized by weighting nodes differently instead of a random jump, i.e., a probability of $\frac{1}{n}$ for all nodes.

After this overview of the graph concepts needed in the rest of this dissertation, we now combine our knowledge of eigenvalues and eigenvectors from Section 2.1 and our knowledge of graphs from this section.

## 2.3 Graph Laplacian

In this section, we discuss the graph Laplacian as well as a related operator on shapes.

Adjacency matrix and degree matrix contain all information about the structure of a graph. There are different ways to combine them now in the Laplacian. In this thesis, we use the *normalized* and the *combinatorial* Laplacian. We can perform an eigendecomposition on both Laplacians such that $\mathscr{L} = \Phi\Lambda\Phi^T$. $\Lambda$ is a diagonal matrix containing the eigenvalues of the graph Laplacian on its diagonal. These eigenvalues $\lambda_1, \dots, \lambda_n$ are called spectrum of the graph. $\Phi$ contains the eigenvectors of $\mathscr{L}$ such that $\Phi = [\phi_1, \dots, \phi_n]$.

We now present the two Laplacians we work with in this thesis as introduced in [23].

**Combinatorial Laplacian.** This is the most simple case of a Laplacian, computed as $\mathscr{L} = \mathbf{D} - \mathbf{A}$. We use this Laplacian in our work on subgraph localization [52] presented in Chapter 6. In this work, an optimization problem is solved which involves editing the edges of the considered graph. Editing an edge changes fewer elements in the combinatorial Laplacian than in the normalized Laplacian, which is favorable for the optimization problem tackled in 6.

**Normalized Laplacian.** This Laplacian is caluclated as $\mathscr{L} = \mathbf{I} - \mathbf{D}^{-\frac{1}{2}}\mathbf{A}\mathbf{D}^{-\frac{1}{2}}$. For the spectrum of this Laplacian $\lambda_1, \ldots, \lambda_n$, it holds that $0 \leq \lambda_i \leq 2$ for $1 \leq i \leq n$. For our work on graph alignment [51] presented in Chapter 4, we use this Laplacian.

Several structural properties of a graph are encoded in the spectrum of a graph, which makes the spectrum an expressive and at the same time compact descriptor. Various statements about the community structure of a graph can be obtained from observing the spectrum. For example, the number of eigenvalues equal to zero corresponds to the number of connected components in a graph [23]. The spectrum and the associated eigenvectors can also be used to infer possible partitions in spectral clustering [101].

### 2.3.1 The Connection between Shapes and Graphs

Inspiration behind the graph correspondence methods developed in this thesis are spectral methods applied to related problems on geometric shapes. Spectral methods have been applied to shape matching [71, 82, 103] and recently also to partial shape localization [112]. In this section, we explore the relation and differences between shapes and graphs and between spectral methods on shapes and graphs.

**Laplace-Beltrami operator.** Graphs and shapes are closely related. In fact, for the algorithmic processing of shapes, they are often discretized in graph form by sampling $n$ points and connecting them as a triangular mesh. These graphs differ from the graphs we consider in this thesis by their spatial component: the positions of nodes relative to each other matter. The graph Laplacian also has an equivalent for shapes, the Laplace-Beltrami operator [103]. Similar to graphs, an eigendecomposition of the Laplace-Beltrami operator is possible, which decomposes the operator into eigenvalues and eigenfunctions. The relationship between the Laplace-Beltrami operator and the Laplacian of a graph is investigated in [152]. This work proves that the eigenvectors of a point cloud graph converge to the eigenfunctions of the Laplace-Beltrami operator of the Riemannian manifold related to that point cloud graph.

For a shape discretized as a graph, also the Laplace-Beltrami operator can be discretized into matrix form. A common discretization scheme for the Laplace-Beltrami operator is the cotangent scheme [90]. Opposite to the Laplacian on graphs, it takes angles between edges and areas of triangles into account. The discretized Laplace-Beltrami operator is calculated as

$$\mathscr{L} = \mathbf{D}\mathbf{W}^{-1}. \tag{2.8}$$

Here, **D** denotes a diagonal matrix with $d_{ii}$ denoting the area of all triangles that share the node $v_i$. Matrix **W** is calculated according to Equation 2.9.

$$w_{ij} = \begin{cases} \frac{cot(\alpha_{ij})+cot(\beta_{ij})}{2} & \text{if } i \neq j \\ \sum_{k \neq i} w_{ik} & \text{if } i = j \end{cases} \tag{2.9}$$

The angles $\alpha$ and $\beta$ denote angles in the two triangles an edge between $v_i$ and $v_j$ participates in. These angles are measured between the two edges of the two triangles which are not $e_{ij}$.

By including angles and areas in the computation of this discretized Laplace-Beltrami operator, it gets a spatial component. This goes beyond the structural information we obtain from the graphs we work with in this thesis. When transferring spectral methods developed for shapes to graphs, this has to be considered.

After having discussed the fundamentals of the Laplacian, in the next section, we turn to the fundamentals of the first graph correspondence task in this thesis: graph alignment.

## 2.4 Graph Alignment

In this section, we discuss different possibilities to approach the graph aligninment task, depending on the specific problem instance as well as the information that is available about the graph. We start by discussing four ways of assigning nodes to each other.

1. **One-to-one alignment**: Here, exactly one node from $G$ is assigned to one node in $G'$. Our method GRASP [51] is applying one-to-one alignment. This approach has the advantage that an unambiguous assignment is made.

2. **One-to-many alignment**: In this case, we assign one node from $G$ to one or more nodes from $G'$. This is for example the case when performing a top-k query for a node to obtain the $k$ most likely matches, e.g. performed by [48]. This results in a *soft alignment*.

3. **Many-to-one alignment**: A node from $G'$ can have multiple nodes from $G$ assigned to it. This can happen, for example, if we assign nodes from $G$ to nodes from $G'$ via nearest-neighbor-search, since in this case, multiple nodes can have the same nearest neighbor.

4. **Many-to-many alignment**: Nodes from $G$ as well as nodes from $G'$ can be assigned to multiple nodes.

These four cases classify graph alignment based on the alignments of individual nodes. If we consider the graph as a whole, there are two possibilities for both graphs: Either we have to find a corresponding match for all nodes or we can ignore single nodes. GRASP [51] aligns every node of the first graph to exactly one node of the second graph. This results in a bijective alignment. Of course, in this case, both graphs must have the same size. This is the most unambiguous alignment. All other possibilities can either be forced, if the graphs are of different sizes, or offer the possibility to treat single nodes as outliers or noise and to ignore them during the alignment process.

If graphs are of different sizes, we call it *partial graph alignment*. If two graphs of the same sizes are to be aligned, we call it *full graph alignment*.

We now discuss the different information that can be used in a graph alignment algorithm. We present a classification scheme for graph alignment algorithms which we developed in [51] and start with the base form of the problem, the unrestricted graph alignment.

### 2.4.1 Unrestricted Graph Alignment

In unrestricted graph alignment, only the binary, symmetric adjacency matrix of the graph is available. All alignments can only be calculated based on the graph structure.

### 2.4.2 Restricted Graph Alignment

In restricted graph alignment, there exists information beyond the structure of the graph. We distinguish between supervised methods, where ground truth alignment knowledge is assumed, and assisted methods, where further information is available in the graph itself.

**Supervised methods**. Here there is a set of seed nodes [60] available, which represent ground truth alignments that are already known before the alignment process, e.g. users of different networks, who have linked the user profile of the second network in the user profile of the first network. Based on these seed nodes, supervised methods infer the alignment.

**Assisted Methods**. Assisted methods do not require ground truth, but other knowledge contained in the graph, such as attributes. These offer additional guidance for the alignment process if e.g. the protein structure is an attribute of a node in a PPI network and thus similarity between nodes can be determined without the structure of the graph. But of course, these attributes can always be noisy or even missing, such that one would have to resort to unrestricted methods again.

After this short introduction to graph alignment, we now turn to the second correspondence problem of this thesis: Subgraph localization

## 2.5 Subgraph Localization

Subgraph localization is a generalization of the graph alignment problem. Instead of a node in graph $G$ being aligned to a node in graph $G'$, we align a graph $Q$ to a subgraph of $G$. Thus we are given a graph $G = (V,E)$ and a query graph $Q = (V_Q,E_Q)$ with $|V_Q| < |V|$. We aim at localizing $Q$ in $G$. To achieve this, we first define subgraphs [4].

**Definition 7** (Subgraph)**.** *Given is a graph $G = (V,E)$. A subgraph $G_s$ of $G$ is a graph $G_s = (V_s,E_s)$ such that $V_s \subseteq V$ and for all edges $e = (v_i,v_j)$ with $e \in E_s$ it holds: $e \in E$ and $v_i,v_j \in V_s$.*

Thus, a subgraph $G_s$ is a graph containing part of the nodes of $G$, and containing only edges that are connecting these nodes also in the original graph. A subgraph $G_s$ that contains all edges that also exist between the corresponding nodes in $G$ is called *induced subgraph* [4].

**Definition 8** (Induced Subgraph)**.** *Given is a graph $G = (V,E)$. An induced subgraph $G_s$ of $G$ is a graph $G_s = (V_s,E_s)$ such that $V_s \subseteq V$ and for all nodes $v_i,v_j \in V_s$ it holds: if $(v_i,v_j) \in E$, then $(v_i,v_j) \in E_S$.*

In our work, we want to localize an induced subgraph, i.e., there exists an induced subgraph $G_s$ of $G$ such that $G_s$ and $Q$ are isomorphic.

**Definition 9** (Subgraph Localization)**.** *Given a graph $G$ and a query graph $Q$, we call an induced subgraph $G_s$ of $G$ a localized subgraph to $Q$, if $Q$ and $G_s$ are isomorphic.*

Of course, this definition can be relaxed, similar to how we went beyond isomorphism in graph alignment. But for our work, we work with this isomorphism idea.

Now, how do we express a localization we have found? A straightforward way is an indicator function, which for each node in $G$ indicates whether it is part of the localized subgraph, as formulated in Equation 2.10

$$\delta(v) = \begin{cases} 1 & \text{if } v \in V_Q \\ 0 & \text{otherwise} \end{cases} \tag{2.10}$$

With this section about subgraph localization, we conclude this chapter and move on to a discussion of related work in the next chapter.

# Chapter 3

# Related Work

In this chapter, we review related work on graph alignment and subgraph localization. Sections 3.1 and 3.2.1 are extracted from [51], Section 3.3 is extracted from [52]. We applied minor corrections and adapted formatting and notation. After the publication of the articles there has been further development in terms of accuracy [49, 107].We have noted a quite active growth of semi-supervised or supervised methods [108, 148, 156] and self-supervision exploiting for instance GNNs or embeddings [122]. In addition, we have identified another recent line of work that focuses on improving given alignments rather than computing them from scratch [30, 49, 58]. These lines of work, although interesting are considered posterior to our publications and left as references for future work.

## 3.1 Graph Alignment

We distinguish graph alignment methods introduced in related work in two main categories: (i) **restricted alignment** methods, which require ground-truth mapping or other additional information; and (ii) **unrestricted alignment** methods, which require neither supervision nor additional information. Table 3.1 overviews the mean features of related works.

### 3.1.1 Restricted Alignment

*Restricted* methods incorporate non-structural information. We further subdivide restricted methods into *supervised* or *assisted* ones.

**Supervised methods** exploit pre-aligned pairs of seed nodes to construct a first alignment. *Percolation graph matching* (PGM) [60, 150] propagates ground-truth align-

| **Method** | Unrestr. | Spectral | Plain | Flexible | Precomp. | Multisc. | Modular | Scalable | Func. |
|---|---|---|---|---|---|---|---|---|---|
| BigAlign [70] | ✘ | ✘ | ✔ | ✘ | ✘ | ✘ | ✘ | ✘ | ✘ |
| FINAL [155] | ✘ | ✘ | ✔ | ✘ | ✘ | ✘ | ✘ | ✔ | ✘ |
| IsoRank [126] | ✘ | ✘ | ✘ | ✔ | ✔ | ✘ | ✘ | ✘ | ✘ |
| GRAMPA [36] | ✔ | ✔ | ✔ | ✔ | ✘ | ✘ | ✘ | ✔ | ✘ |
| LaplMatch [64] | ✔ | ✔ | ✔ | ✘ | ✔ | ✘ | ✘ | ✘ | ✘ |
| LREA [95] | ✔ | ✔ | ✔ | ✘ | ✘ | ✘ | ✘ | ✔ | ✘ |
| REGAL [48] | ✔ | ✘ | ✔ | ✔ | ✔ | ✘ | ✔ | ✔ | ✘ |
| CONE [19] | ✔ | ✘ | ✔ | ✔ | ✔ | ✘ | ✔ | ✘ | ✘ |
| S-GWL [145] | ✔ | ✘ | ✔ | ✔ | ✔ | ✔ | ✘ | ✘ | ✘ |
| **GRASP** | ✔ | ✔ | ✔ | ✔ | ✔ | ✔ | ✔ | ✔ | ✔ |

Table 3.1: Related work in terms of present (✔) and absent (✘) properties: *supervised* methods require aligned nodes as input; *spectral* methods use the spectra of alignment matrices; IsoRank does not work on *plain* graph structures as it requires auxiliary node similarity input; FINAL requires node or edge attribute information to perform reasonably well [48]; *flexible* methods accommodate different alignment algorithms (e.g. bipartite matching, nearest neighbors). Among unrestricted methods (rows 2–7), LREA does not benefit from *offline precomputation*. GRASP naturally captures *multiscale* properties thanks to its spectral basis, while S-GWL coarsens the graph to progressively match structures at different scales. REGAL, CONE and GRASP are *modular*, hence allow boosting various parts of the algorithm. FINAL, LREA, REGAL, and GRASP can *scale*, running on graphs of more than 10 000 nodes in less than 1 hour. GRASP treats the problem as an alignment among *functions*, embracing a variety of node functions, including embeddings.

ments across the network. *Representation learning* approaches, such as IONE [83], PALE [87], and DeepLink [157], learn a low-dimensional embedding of the graph nodes and map the node embeddings of one graph to another. A similar method aligns multiple networks at once [22]. Other recent embedding-based alignment methods, such as cM$^2$NE [144], SSPDG [159] and NEXTALIGN [156] require pre-aligned seed nodes. BRIGHT [148] uses embeddings and *neighborhood consistency*, by which an alignment should preserve the neighborhoods of aligned nodes. *Active network alignment* [86] and, recently, ATTENT [158] apply active learning to elicit expert guidance on alignments. DANA [30], an adversarial-based learning method, also requires ground-truth alignments in the training phase. Overall, such supervised methods rely on prior knowledge, which may not be available.

**Assisted methods** utilize auxiliary information or structural constraints. BigAlign [70] focuses on bipartite graphs; however, most graphs are not bipartite. FINAL [155] aligns nodes based on similarity of topology and attributes. IsoRank [126] aligns multiple protein-protein interaction networks aiming to maximize quality across all input networks; it constructs an eigenvalue problem for every pair of input networks and extracts a global alignment across the input set by a $k$-partite matching; it relies on structural properties (PageRank), but also on a similarity measure between nodes which in a biology-specific case builds on the similarity of the proteins. It is improved by a greedy approach in [67] and in IsoRankN [81], which performs spectral clustering on the induced graph of pairwise alignment scores, claiming error-tolerance and computationally efficiency. Both FINAL and IsoRank also present an unrestricted variant, the former variant being a scaled version of the latter. Yet, as they are deliberately designed for, and perform well in, the case where additional information is available, we classify both methods as restricted. GSANA [151] lets pairwise distances to seed nodes guide the matching. Another variant matches *weigthed* matrices using their spectra [136]; that is inapplicable to the unweighted case. Karakasis et al. [58] propose a refinement method for precomputed alignments, which qualifies as an assisted method rather than a stand-alone alignment algorithm. WAlign [42] aligns attributed graphs using Graph Convolutional Networks and a Wasserstein discriminator. Overall, such restricted methods cannot handle cases where the only given information is graph structure.

### 3.1.2 Unrestricted Alignment

*Unrestricted* methods require neither prior knowledge of ground-truth pairs nor other information on the input graph. We categorize them as follows.
**Integer-programming methods.** Klau [62] presents a Lagrangian relaxation for the integer programming problem posed by network alignment; though the resulting algorithm is polynomial, it is still impracticable for large networks.

**Embedding-based methods.** REGAL [48] constructs node embeddings based on the connectivity structure and node attributes, and uses the similarity between these features for node alignment; we classify REGAL as an unrestricted method since it can work without attributes. CONE [19] realigns node representations, without prejudice to the representation used. GWL [146] jointly optimizes for node embeddings and alignment by minimizing the Gromov-Wasserstein distance among the embedding distances. Contrariwise to CONE and REGAL, it computes the embeddings and the final alignment at the same time. S-GWL [145], a scalable version of GWL, achieves scalability by aligning progressively smaller graph partitions.

**Matrix decomposition methods.** EigenAlign [37] formulates the problem as a Quadratic Assignment Problem that considers both matches and mismatches and solves it by *spectral decomposition* of matrices. Building thereupon, Low-Rank EigenAlign (LREA) [95] solves a maximum weight bipartite matching problem on a *low-rank* version of a node-similarity matrix, hence requires memory linear in the size of the graphs. However, EigenAlign variants use the first eigenvector of a joint adjacency matrix between the two graphs to be aligned, rather than the eigenvectors of graph Laplacians, which provides richer information.

**Belief propagation methods.** Alternatively, NetAlign [8] solves a specific *sparse* variant of the graph alignment problem by a message-passing algorithm.

## 3.2 Shape Matching

Our work is inspired by shape matching methods that employ spectral properties [71, 82, 103]. Functional maps [103] generalize the matching of points to the matching of *corresponding functions* among shapes, by revealing a common decomposition of such functions using the eigenvectors of the Laplace-Beltrami operator; the graph equivalent of that operator is a graph's Laplacian matrix. Extensions of this methods match non-isometric shapes by aligning their Laplace-Beltrami operators' eigenbases [71], and match a part of a shape to another full shape in the spectral domain [82] without requiring spatially modeling the part of a shape.

### 3.2.1 Spectral Methods

Graph spectra [23] facilitate problem-solving in graph analysis, image partitioning, graph search, and machine learning [125]. NetLSD [135] uses Laplacian spectral signatures to detect graph similarity, *but not to align graphs*, in a multi-scale fashion. LaplMatch [64] derives a permutation matrix for shape matching from Laplacian eigenvectors, without considering multiscale properties. While calculating a graph's spectrum is computationally challenging, recent work proposes an approximation via spectral moments estimated through random walks [24].

## 3.3 Subgraph Localization

We review related work on five problems related to subgraph localization, namely subgraph isomorphism, subgraph discovery, subgraph querying, subgraph matching, and subgraph localization.

### 3.3.1 Subgraph Isomorphism

The *subgraph isomorphism* problem is to decide whether a source graph contains a target subgraph and return that exact subgraph in the source. In graph analytics, this problem is mainly solved for very small target subgraphs ($\leq 10$ nodes) and aims at exact matches. Several methods speed up this process by exploiting query specifics, such as patterns in multiple subgraph queries [34]. By contrast, our method aims at bigger target subgraphs.

### 3.3.2 Subgraph Discovery

In *subgraph discovery*, a target subgraph is not given as input, yet the problem is to identify interesting components of a source graph according to some criteria, as, e.g., those that appear frequently [74, 75], achieve a density threshold [77, 110], or form cliques [13].

### 3.3.3 Subgraph Querying

In *subgraph querying*, the goal is to identify all source graphs among a collection that contain a query target subgraph, without necessarily indicating the position of that subgraph within the returned graphs [59, 129, 130]. A closely related topic is subgraph retrieval, where the goal is to retrieve the most relevant graphs from a graph database, with relevance being measured by some score. In [117] node embeddings are learned in order to produce a subgraph matching for the computation of the relevance score. In [80], nodes are matched in order to produce a graph similarity score without producing node embeddings as an intermediate step. In both cases, the queries are significantly smaller than ours.

### 3.3.4 Subgraph Matching

The goal of *Subgraph matching* is to match the nodes of a smaller graph to those of a subgraph in a bigger graph via minimizing some error criteria, possibly in the presence of available attribute information. Many methods for graph matching effectively solve a subgraph isomorphism problem, even though they are not specifically designed for this purpose [155]. Recent work [80, 84] employs deep neural models to learn *node embeddings* that are subsequently used for matching.

### 3.3.5 Subgraph Localization

The problem of *subgraph localization* calls to detect the best fit of a target subgraph within a bigger source graph, without aiming for full isomorphism. This problem has been scarcely studied. A recent application in computer vision [147] uses subgraph localization to detect temporal actions, where a graph models actions and the temporal relations between them. However, this model uses edges for temporal aspects and inter-scene relations, and hence does not generalize to arbitrary graphs. An existing spectral solution [17] is limited to special families of graphs, such as cliques.

# Part II

# Publications

# Chapter 4

# Spectral Graph Alignment

The work presented in this paper was prepared in collaboration with Konstantinos Skitsas, Anton Tsitsulin, Marina Munkhoeva, Alexander Frederiksen Kyster, Simon Daugaard Nielsen, Alex Bronstein, Davide Mottin and Panagiotis Karras. It combines and extends two papers published at APWeb-WAIM 2021 [50] and CIKM 2021 [76]. It is currently under review for TKDD [51]. We applied minor corrections and adapted formatting and notation and removed the section on related work, as it is already presented in Chapter 3.

## 4.1 Abstract

What is the best way to match the nodes of two graphs? This *graph alignment* problem generalizes graph isomorphism and arises in applications from social network analysis to bioinformatics. Some solutions assume that auxiliary information on known matches or node or edge attributes is available, or utilize arbitrary graph features. Such methods fare poorly in the pure form of the problem, in which only graph structures are given. Other proposals translate the problem to one of aligning node embeddings, yet, by doing so, provide only a single-scale view of the graph.

In this paper, we transfer the shape-analysis concept of functional maps from the continuous to the discrete case, and treat the graph alignment problem as a special case of the problem of finding a mapping between functions on graphs. We present GRASP, a method that first establishes a correspondence between functions derived from Laplacian matrix eigenvectors, which capture multiscale structural characteristics, and then exploits this correspondence to align nodes. We enhance the basic form of GRASP by altering two of its components, namely the embedding method and

the assignment procedure it employs, leveraging its modular, hence adaptable design. Our experimental study, featuring noise levels higher than anything used in previous studies, shows that the enhanced form of GRASP outperforms scalable state-of-the-art methods for graph alignment across noise levels and graph types, and performs competitively with respect to the best non-scalable ones. We include in our study another modular graph alignment algorithm, CONE, which is also adaptable thanks to its modular nature, and show it can manage graphs with skewed power-law degree distributions.

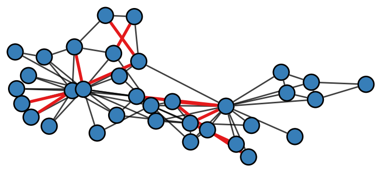
## 4.2 Introduction

Graphs model relationships between entities in several domains, e.g., social networks, protein interaction networks, email communication or chemical molecules. The structure of such graphs captures information on, e.g., people's connections, molecule functions, and protein interactions.

At the same time, the expressive nature of graphs also implies complexity, which renders some fundamental problems hard. For instance, the *graph isomorphism* problem, which is to determine whether two graphs share the same structure is neither known to be polynomially solvable nor **NP**-complete, and has been used to define the **GI** complexity class [65]. Problems that generalize graph isomorphism occur frequently in the field of graph analytics. One of those is the **NP**-complete *subgraph isomorphism* problem; another is *graph alignment*, which aims to find the best (exact or inexact) matching among the nodes of a pair of graphs; a solution to this problem is sine qua non in tasks such as identifying users in different social networks [60], matching objects in images by establishing feature correspondences and comprehending protein response in the body [62].

In case additional background information is available, such as node and edge attributes in the graphs to be aligned or valid *seed* matches, then the problem is solvable via supervised methods [22, 83]. However, in case only graph structures are given, then the problem of aligning two graphs by matching structures, is at least as hard as graph isomorphism even in its approximate version [1].

Existing approaches to graph alignment are oriented toward using a few *heuristic graph features*, such as landmarks, in order to detect a good alignment [48], *exploiting additional information* such as node attributes [155] or bipartite networks [70], or optimizing objectives based only on *local connections* among nodes [37, 81, 95]. On the other hand, the spectra of *Laplacian matrices* have been successfully employed to


(a) Karate club; Red edges removed.

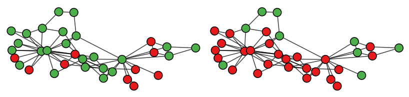

(b) Alignment by GRASP (left) and REGAL (right).

Figure 4.1: With a few removed edges, REGAL [48], an alignment method based on *local* features, fails to correctly align the distorted Karate club graph to the original; GRASP identifies most of nodes (correctly aligned nodes in green).

devise a similarity measure among graphs [135]. Laplacian spectra capture important *multiscale* properties, such as local-scale ego-nets and global-scale communities. Previous approaches rooted in spectral characteristics decompose large matrices expressing all alignments among edges in two graphs [37, 81, 95] and formulate the solution as finding the leading eigenvector of such matrices. These approaches disregard most eigenvectors and consider only local edge variations. To our knowledge, the spectral properties of *Laplacian matrices* have *not yet* been utilized to any significant extent for an end-to-end graph alignment method.

In this paper, we propose GRASP, short for **GR**aph **A**lignment through **SP**ectral Signatures, a principled method to detect a good alignment among graphs based on their spectral characteristics, i.e., eigenvalues and eigenvectors of their Laplacian matrices [23]. We transfer the methodology of matching among shapes based on *corresponding functions* [103] to the domain of graphs: we first extract a mapping of node-evaluated functions, based on, for example, the graph's heat kernel or PageRank measures, and then apply this mapping to the matching on nodes. Figure 4.1 shows an example alignment of the Karate club with a deteriorated version obtained by removing some edges; GRASP correctly aligns most of the nodes, while REGAL [48] based on local descriptors fails to do so.

In addition, we interpret GRASP as a representative of a family of *modular* graph alignment algorithms; GRASP is modular by nature, i.e., made of adjustable components. In contrast, other graph alignment methods are *monolithic*, i.e., they are made of components such as matrix factorization [81, 95] or integer programming [62] that were hard to adapt, are designed for a specific graph type, e.g., biological [5] or bipartite networks [70], and fare poorly on other types [48]. Embedding-based methods [148, 159] strongly rely on an appropriate embedding model tailored for each graph type. On the other hand, the *modular nature* also encapsulates two other graph alignment methods, REGAL [48] and CONE [19], that leverage advances in node embeddings [109, 113]; it is outlined as follows:

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

We propose enhancements in each part of this modular framework in GRASP and CONE, interchanging, enhancing, and adding to framework components. In a targeted experimental study, we evaluate GRASP with a set of different components and show that these enhancements improve upon the effectiveness in recovering real-graph alignments with high accuracy and nearly no impact on efficiency. This work completes, consolidates, and extends two precedent short publications, [50] and [76]; it expands and completes the experimental study in [50] by applying to GRASP the enhancements introduced in [76] and including a wider array of data sets and an exhaustive set of competing methods. In particular, we contribute the following material in addition to previously published work:

1. We extensively discuss the modular graph alignment framework and possible enhancements to GRASP (Section 4.5).
2. We include the recently proposed graph alignment methods S-GWL [145] and CONE [19] in our experimental comparison (Section 6.5).
3. We examine and provide additional experimental insights on the effects of parameter choice in GRASP, including the parameter $k$, i.e., the number of eigenvectors used for computations and $q$, i.e., the number of corresponding functions used (Section 4.6.1).
4. We conduct new experiments on real-world data, namely two temporal proximity networks and different variants of a PPI-network (Table 4.1, Section 4.6.4).
5. We provide additional experiments examining scalability in the number of nodes in a graph (Section 4.6.6).

## 4.3 Background and Problem

**Graph Alignment.** Consider two undirected graphs, $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$, where $V_*$ are node sets, $E_* \subseteq V_* \times V_*$ are edges, and[1] $|V_1| = |V_2| = n$. A graph's *adjacency matrix* $\mathbf{A} \in \{0,1\}^{n \times n}$ is a binary matrix where $\mathbf{A}_{ij} = 1$ if there is an edge between nodes $i$ and $j$ and $\mathbf{A}_{ij} = 0$ otherwise.

**Definition 10.** *Given two graphs $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$, a graph alignment $R : V_1 \to V_2$ is an* injective *function that maps nodes of $G_1$ to nodes of $G_2$.*

The graph alignment problem is to find such a function, which, expressed as a permutation matrix $\mathbf{P}$, minimizes the difference $||\mathbf{P}\mathbf{A}_1\mathbf{P}^\top - \mathbf{A}_2||^2$. In case of isomorphic graphs, there exists a $\mathbf{P}$ such that $\mathbf{P}\mathbf{A}_1\mathbf{P}^\top = \mathbf{A}_2$, i.e., aligns the two graphs exactly. We are interested in the general, unrestricted problem case, in which there are no additional constraints on node attributes or matches known in advance. The problem is hard and not known to be in **NP**.

We may express graph alignment in terms of a ground truth function $\tau : V_1 \to V_2$ that returns the correct alignment between the nodes $V_1$ in $G_1$ and the nodes $V_2$ in $G_2$. In the case of isomorphic graphs, this ground truth function $\tau$ is a bijection that admits an inverse mapping $\tau^{-1} : V_2 \to V_1$. The composition of the indicator function $\delta_i : V_1 \to \{0,1\}$ with $\tau^{-1}$, $\delta_i \circ \tau^{-1} : V_2 \to \{0,1\}$ expresses the complete isomorphism among the two graphs, returning 1 if node $u \in V_2$ maps to node $i \in V_1$, 0 otherwise. By generalization, the composition $g_i = f_i \circ \tau^{-1}$ maps functions in $G_2$ to functions in $G_1$ for any family of real-valued functions $f_1, ..., f_q, f_i : V_1 \to \mathbb{R}$ and $g_1, ..., g_q, g_i : V_2 \to \mathbb{R}$ that associate a real value to each node in $G_1$ and $G_2$. This transformation among functions is called a *functional representation* of the mapping $\tau$. In effect, finding an alignment among the nodes of two graphs corresponds to finding an alignment among functions on those nodes. We use such *functional alignments* as a shortcut to *node alignments*. To get there, we extend the concept of a functional map [103] from the continuous to the discrete case.

**Functional maps.** The operator $T_{\mathscr{F}} : (V_1 \times \mathbb{R}) \to (V_2 \times \mathbb{R})$ maps functions $f$ on the nodes in $G_1$ to functions $g$ on the nodes in $G_2$, i.e. $T_{\mathscr{F}}(f) = f \circ \tau^{-1} = g$. This operator is linear in the function space, i.e., $T_{\mathscr{F}}(c_1 f_1 + c_2 f_2) = (c_1 f_1 + c_2 f_2) \circ \tau^{-1} = c_1 f_1 \circ \tau^{-1} + c_2 f_2 \circ \tau^{-1} = c_1 T_{\mathscr{F}}(f_1) + c_2 T_{\mathscr{F}}(f_2)$. In addition, let $\phi_1, ..., \phi_n$ and $\psi_1, ..., \psi_n$ denote orthogonal bases for the space of functions on $G_1$'s nodes, $V_1 \times \mathbb{R}$, and that on $G_2$'s

[1]Solutions to the problem of aligning graphs with unequal numbers of nodes can rest on solutions to this basic problem form; we may append zero-entries to the eigenvectors of the smaller graph to obtain the eigenvector size of the larger graph.

nodes, $V_2 \times \mathbb{R}$, respectively. Since those functions produce $n$-dimensional vectors, we can represent them as linear combinations of their basis vectors, $f = \sum_{i=1}^{n} a_i \phi_i$ and $g = \sum_{j=1}^{n} b_j \psi_j$. Then, by the linearity of $T_{\mathscr{F}}$,

$$T_{\mathscr{F}}(f) = T_{\mathscr{F}}\left(\sum_{i=1}^{n} a_i \phi_i\right) = \sum_{i=1}^{n} a_i T_{\mathscr{F}}(\phi_i) = \sum_{i=1}^{n} a_i \sum_{j=1}^{n} c_{ij} \psi_j = \sum_{j=1}^{n} b_j \psi_j$$

where $T_{\mathscr{F}}(\phi_i) = \sum_{j=1}^{n} c_{ij} \psi_j$. It follows that each coefficient $b_j$ is the dot-product $\sum_{i=1}^{n} a_i c_{ij}$ between the coefficients $(a_1, ...., a_n)$ of functions in $G_1$ and the coefficients $(c_{1j}, ...., c_{nj})$ of the operator $T_{\mathscr{F}}$. In conclusion, in order to align real-valued functions on the nodes of two graphs, we need to find a *mapping matrix* $\mathbf{C} \in \mathbb{R}^{n \times n}$ of coefficients among those functions; such a mapping matrix $\mathbf{C}$ maps functions from $G_1$ to $G_2$, even when the ground-truth mapping $\tau$ is unknown. In a nutshell, GRASP obtains such a mapping matrix $\mathbf{C}$ for a well-chosen function and applies that $\mathbf{C}$ to mapping the indicator function $\delta$ from $G_1$ to $G_2$, thereby constructing a node alignment. The main question we need to answer is what orthogonal basis and functions we should use to construct our mapping matrix $C$. The next section answers this question and builds on that answer to devise a solution.

## 4.4 Solution

Here, we choose an orthonormal basis and a function, which are, in our judgement, appropriate for node alignment purposes, and define the complete pipeline of our solution. First, in Section 4.4.1 we choose a basis for the functions. In Section 4.4.2, we choose functions that capture important graph properties. Then, in Section 4.4.3, we define a mapping matrix to map functions across graphs. In Section 4.4.4, we compute node-to-node alignments from such a map, and Section 4.4.5 describes a method to refine the mapping matrix.

### 4.4.1 Choice of basis: Normalized Laplacian

As a basis for representing functions as linear combinations of base functions, we use the eigenvectors of the graph's *normalized Laplacian*, i.e., the matrix $\mathscr{L} = I - \mathbf{D}^{-\frac{1}{2}} \mathbf{A} \mathbf{D}^{-\frac{1}{2}}$, where $\mathbf{D}$ a diagonal degree matrix of node degrees $\mathbf{D}_{ii} = \sum_{j=1}^{n} \mathbf{A}_{ij}$ and $\mathbf{A}$ is the graph adjacency matrix; its eigendecomposition is $\mathscr{L} = \Phi \Lambda \Phi^{\top}$, where $\Lambda$ is a diagonal matrix of eigenvalues, $\{\lambda_1, \ldots, \lambda_n\}$ ordered by non-decreasing value, i.e., the graph's *spectrum*, which encodes information about communities, degree distribution, and diameter, and $\Phi$ is a matrix of corresponding eigenvectors, $\Phi_{\mathscr{L}} = [\phi_1 \phi_2 \ldots \phi_n]$.

The eigenvectors form an orthogonal basis, which we use a standard basis. We use $\phi$ ($\psi$) to indicate the eigenvectors of the Laplacian of graph $G_1$ ($G_2$).

We consider this basis to be suitable, since the eigenvectors of the normalized Laplacian converge to the eigenfunctions of the Laplace-Berltrami operator [9], which measures the smoothness of continuous surfaces.

### 4.4.2 Choice of function: Heat Kernel

The choice of functions $f_i : V_1 \to \mathbb{R}$, $g_i : V_2 \to \mathbb{R}$, is critical for our method. A poor choice would be detrimental. A good choice should have the following properties:

**Expressiveness.** The function should express the graph's structure. For instance, a constant function returning the same value for all nodes would not yield a meaningful alignment.

**Permutation-invariance.** The function should not depend on the node index $i$; the indicator function lacks this property.

**Multiscale robustness.** The function should robustly capture both local and global structures (e.g., edges and communities), insensitively to small perturbations.

A function fulfilling these requirements is the time-parameterized *heat kernel* [135]:

$$\mathbf{H}_t = \Phi e^{-t\Lambda} \Phi^\top = \sum_{j=1}^{n} e^{-t\lambda_j} \phi_j \phi_j^\top \tag{4.1}$$

where $\mathbf{H}_{t[ij]}$ measures the flow of heat from node $i$ to node $j$ at time $t$, as it diffuses from each node's neighborhood to the whole graph. We build our model functions over a sequence of time steps $t$ using the *diagonal* of the heat kernel, which measures the heat flowing back to each node at time $t$.

The heat kernel expresses multiscale graph structure in a permutation-invariant manner and is robust to small changes. In the beginning of the diffusion, Equation (4.1) emphasises large $\lambda$, which correspond to *local* edge and ego-net properties. As time progresses, smaller eigenvalues get emphasized, reflecting *global* graph properties, such as communities.

We build our *corresponding functions* $f_i$, $g_i$, from the heat kernel at different time steps $t$, as linear combinations of the graph's Laplacian orthogonal eigenvectors. Let $\mathbf{F} \in \mathbb{R}^{n \times q}$, $\mathbf{F} = [\mathbf{f}_1, \dots, \mathbf{f}_q]$ be the matrix containing the diagonals of the heat kernel of $G_1$, $\mathbf{H}_t^{G_1}$, over $q$ time[2] steps, $\mathbf{f}_i = \sum_{j=1}^{n} e^{-t_i \lambda_j} \phi_j \odot \phi_j$, where $\odot$ denotes the element-wise vector product. Likewise, the matrix $\mathbf{G} \in \mathbb{R}^{n \times q}, \mathbf{G} = [\mathbf{g}_1, \dots, \mathbf{g}_q]$ contains the

[2]In our experiments we select $q = 100$ values evenly spaced on the linear scale in the range [0.1, 50].

diagonals of $\mathbf{H}_t^{G_2}$, the heat kernel of $G_2$. While the *q columns* of **F** and **G** contain the same time-dependent heat-kernel-diagonal functions on the nodes of two graphs, their *n rows* (i.e., nodes) are not aligned. We need to obtain such a node alignment.

### 4.4.3 Mapping matrix

We approximate each function $f_i$ using only the first $k$ eigenvectors, as done, by analogy, on shapes analysis [9], and thereby calculate the corresponding function matrices $F$ and $G$. **F** and **G** can be thought as coefficient matrices used to produce linear combinations, $\mathbf{F}^\top\Phi$ and $\mathbf{G}^\top\Psi$, of the Laplacian eigenvectors of $G_1$ and $G_2$, respectively. With a slight abuse of notation, we denote with $\Phi$ and $\Psi$ the first $k$ eigenvectors, hence $\mathbf{F}^\top\Phi$ and $\mathbf{G}^\top\Psi$ are in $\mathbb{R}^{q\times k}$. In the projection of the functions on the first $k$ eigenvectors, we would like the corresponding functions to be equal up to a coefficient matrix $\mathbf{C}\in\mathbb{R}^{k\times k}$. In the case of isomorphic graphs, it holds that $\mathbf{F}^\top\Phi = \mathbf{G}^\top\Psi\mathbf{C}$

$$\begin{bmatrix} \mathsf{diag}(\mathbf{g}_1^\top\Psi) \\ \vdots \\ \mathsf{diag}(\mathbf{g}_q^\top\Psi) \end{bmatrix} \begin{bmatrix} c_{11} \\ \vdots \\ c_{kk} \end{bmatrix} = \begin{bmatrix} \Phi^\top\mathbf{f}_1 \\ \vdots \\ \Phi^\top\mathbf{f}_q \end{bmatrix} \tag{4.2}$$

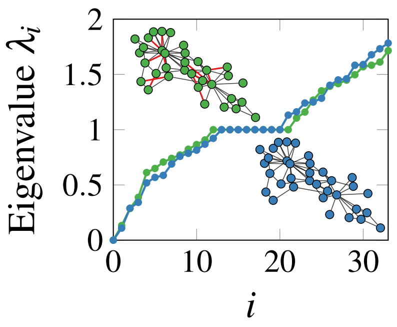
Matrix **C** is diagonal in the case of isomorphic graphs and deviates from a diagonal form as graphs diverge from isomorphism; for simplicity, we assume a diagonal **C**, and obtain the diagonal entries that minimize the $L_2$-norm difference $\|\cdot\|_2^2$ between the left and rights side of Eq. (4.2) using the ordinary least squares method, as in [71]. In Section 4.4.5 we delve into the case of non-isomorphic graphs.

### 4.4.4 Node-to-node correspondence

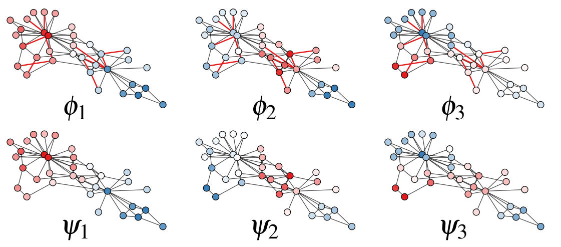
We consider the delta function $\delta_i(\cdot)$ as corresponding function; these functions yield an $n\times n$ identity matrix. We express such a function as a vector of coefficients, since the vector of $\delta_i$ is the $i$th row of the heat kernel at $t=0$:

$$\delta_i = \mathbf{H}_{i,t=0}^{G_1} = \textstyle\sum_{j=1}^{n} \phi_{ij}\phi_j$$

The computation for delta functions on $G_2$ follows equivalently using $\Psi$ in place of $\Phi$. We may match the coefficient vectors of these corresponding indicator functions, as, ideally, for two matching nodes $v_i\in V_1$ and $v'_j\in V_2$, the coefficients of $\delta_i$ and $\delta_j$ for $\Phi$ and $\Psi$ should be identical. In particular, the coefficients expressing $\delta_i$ as a linear combination of the first $k$ eigenvectors are $\phi_{i1},\ldots,\phi_{ik}$. We set $\Phi^\top$ and $\mathbf{C}\Psi^\top$ in $\mathbb{R}^{k\times n}$



(a) Two graphs spectra



(b) Their first 3 eigenvectors

Figure 4.2: We remove **red** edges from the **green** graph to obtain the **blue** graph. Eigenvalues interlace (a); eigenvectors $\phi_1, \phi_2, \phi_3$ for **green** and $\psi_1, \psi_2, \psi_3$ for **blue** highlight common structures.

as coefficient matrices of the delta functions, aligned by $\mathbf{C}$. Rows correspond to the first $k$ Laplacian eigenvectors, while columns stand for graph nodes, rather than for time steps of heat diffusion. We need to match coefficient vectors, i.e., columns of $\Phi^\top$ and $\mathbf{C}\Psi^\top$, to each other. This problem amounts to a *linear assignment problem*; we apply an off-the-shelf algorithm therefore, such as **nearest neighbor search** or **Jonker-Volgenant (JV)** [57], to obtain a one-to-one matching between the columns of $\Phi^\top$ and $C\Psi^\top$and hence an alignment of nodes. GRASP is flexible in that we may choose any matching method.

## 4.4.5 Base Alignment

We have hitherto assumed that the graphs to be aligned are isomorphic, hence their eigenvectors correspond to each other with possible sign changes and an orthogonal diagonal mapping matrix $\mathbf{C}$ exists. Still, if the graphs are not isomorphic, then their eigenvectors diverge and the diagonal matrix $\mathbf{C}$, which we enforce, cannot capture their relationship well. Figure 4.2 highlights this issue: at a high level the eigenvectors underline common structures, but they differ at the node level. In this case, we need to *align* the two eigenvector bases before we consider aligning corresponding vectors and, eventually, nodes. We express this *base alignment* [71] in terms of an alignment matrix $\mathbf{M}$.

**Diagonalization.** We align the eigenvectors $\Psi$ by a rotation matrix $\mathbf{M}$ so as transform $\Psi$ into $\Phi$: $\hat{\Psi} = \Psi\mathbf{M}$. Since $\mathscr{L}\Psi = \Psi\Lambda$, finding $\Psi$ is equivalent to the solution of the quadratic minimization problem $\min_\Psi \mathrm{off}(\Psi^\top \mathscr{L}_2 \Psi)$ s.t. $\Psi^\top\Psi = \mathbf{I}$ which penalizes the sum of elements $\mathsf{off}(\cdot)$ outside of the diagonal, in order to preserve orthogonality of the basis.

Since the eigenvectors are orthonormal, $\Psi^\top\Psi = \mathbf{I}$ and for $G_2$'s graph Laplacian eigenvectors $\Lambda_2$, $\Psi^\top \mathscr{L}_2 \Psi = \Psi^\top \Psi \Lambda_2 = \Lambda_2$, and $\mathbf{M}^\top \Psi^\top \mathscr{L}_2 \Psi \mathbf{M} = \mathbf{M}^\top \Lambda_2 \mathbf{M}$. Putting the above together, our diagonalizing term is:

$$\min_{\mathbf{M}} \mathsf{off}(\mathbf{M}^\top \Lambda_2 \mathbf{M}) \text{ s.t. } \mathbf{M}^\top \mathbf{M} = \mathbf{I}$$

As we are minimizing over orthogonal matrices we can equivalently express the objective above as a minimization over orthogonal matrices of size $n \times n$, $S(n,n)$:

$$\min_{\mathbf{M} \in S(n,n)} \mathsf{off}(\mathbf{M}^\top \Lambda_2 \mathbf{M})$$

**Coupling.** In addition, the correspondence $\tau : G_1 \to G_2$ so that $\phi_i \approx \tau \circ \psi$ translates to

$$\min_{\Phi} \|\mathbf{F}^\top \Phi - \mathbf{G}^\top \Psi \mathbf{M}\|_F^2$$

where $\mathbf{F}$ and $\mathbf{G}$ contain each graphs's corresponding functions. We combine the minimization terms for diagonalization and coupling, to get the following minimization problem, with regularization factor $\mu$[3]:

$$\min_{\mathbf{M} \in S(n,n)} \mathsf{off}(\mathbf{M}^\top \Lambda_2 \mathbf{M}) + \mu \|\mathbf{F}^\top \Phi - \mathbf{G}^\top \Psi \mathbf{M}\|_F^2 \tag{4.3}$$

Given that the eigenvectors of isomorphic graphs match with sign changes, we initialize $\mathbf{M}$ as a diagonal matrix with $\mathbf{M}_{ii} = 1$ if $\|\mathbf{F}^\top \phi_i - \mathbf{G}^\top \psi_i\| \leq \|\mathbf{F}^\top \phi_i + \mathbf{G}^\top \psi_i\|$, $\mathbf{M}_{ii} = -1$ otherwise. Eq. (4.3) leads to a manifold optimization problem, which we solve by trust-region methods [2].

**Scalability.** We avoid computing all eigenvectors $n \times n$, exploiting the fact that we only need the first $k$ eigenvectors for calculating $\mathbf{C}$ (see Section 4.4.3). So we only align the first $k$ eigenvectors of $\Psi$ to the first $k$ eigenvectors of $\Phi$, i.e $\bar{\Phi} = \hat{\Psi} = \bar{\Psi}\mathbf{M}$ with $\bar{\Phi} = [\phi_1, \ldots, \phi_k]$ and $\bar{\Psi} = [\psi_1, \ldots, \psi_k]$. Let $\bar{\Lambda}_2 = \mathsf{diag}(\lambda_1, \ldots, \lambda_k)$, the problem in Eq. (4.3) becomes

$$\min_{\mathbf{M} \in S(k,k)} \mathsf{off}(\mathbf{M}^\top \bar{\Lambda}_2 \mathbf{M}) + \mu \|\mathbf{F}^\top \bar{\Phi} - \mathbf{G}^\top \bar{\Psi} \mathbf{M}\|_F^2 \tag{4.4}$$

[3] $\mu = 0.132$ in our experiments

After obtaining $\mathbf{M}$, we use the eigenvectors in $\bar{\Phi}$ and the aligned eigenvectors $\hat{\Psi} = \bar{\Psi}\mathbf{M}$ in the next step for the final alignment of nodes.

Our approach effectively trades off graph alignment with a proxy problem of manifold optimization, which we solve with reasonable accuracy and scalability.

**Algorithm 1** GRASP

```
Require: Graphs $G_1 = (V_1, E_1)$, $G_2 = (V_2, E_2)$
Params: Eigenvectors $k$, corresponding functions $q$, time steps $t = [t_1, \dots, t_q]$
    // Step 1: Eigendecomposition of Laplacian
 1: $\mathscr{L}_1 \leftarrow \mathbf{I} - \mathbf{D}_1^{-\frac{1}{2}} \mathbf{A}_1 \mathbf{D}_1^{-\frac{1}{2}}$, $\mathscr{L}_2 \leftarrow \mathbf{I} - \mathbf{D}_2^{-\frac{1}{2}} \mathbf{A}_2 \mathbf{D}_2^{-\frac{1}{2}}$
 2: $\Phi, \Lambda_1 \leftarrow \mathsf{eig}(\mathscr{L}_1)$
 3: $\Psi, \Lambda_2 \leftarrow \mathsf{eig}(\mathscr{L}_2)$
    // Step 2: Compute corresponding functions
 4: for all $t_i$ in $t$ do
 5:     $\mathbf{F}_i \leftarrow \sum_{j=1}^{n} e^{-t_i \lambda_j} \phi_j \odot \phi_j$
 6:     $\mathbf{G}_i \leftarrow \sum_{j=1}^{n} e^{-t_i \lambda_j} \psi_j \odot \psi_j$
 7: $\mathbf{F} \leftarrow [\mathbf{f}_1, \dots, \mathbf{f}_q]$
 8: $\mathbf{G} \leftarrow [\mathbf{g}_1, \dots, \mathbf{g}_q]$
    // Step 3: Base alignment
 9: $\bar{\Phi} = [\phi_1, \dots, \phi_k]$
10: $\bar{\Psi} = [\psi_1, \dots, \psi_k]$
11: $\mathbf{M} \leftarrow \min_{\mathbf{M} \in S(k,k)} \mathsf{off}(\mathbf{M}^\top \bar{\Lambda}_2 \mathbf{M}) + \mu \|\mathbf{F}^\top \bar{\Phi} - \mathbf{G} \bar{\Psi} \mathbf{M}\|_F^2$
12: $\hat{\Psi} = \bar{\Psi}\mathbf{M}$
    // Step 4: Calculate mapping matrix
13: $$\min_{[c_{11},\dots,c_{kk}]^\top} \left\| \begin{bmatrix} \mathsf{diag}(\mathbf{g}_1^\top \hat{\Psi}) \\ \vdots \\ \mathsf{diag}(\mathbf{g}_q^\top \hat{\Psi}) \end{bmatrix} \begin{bmatrix} c_{11} \\ \vdots \\ c_{kk} \end{bmatrix} - \begin{bmatrix} \bar{\Phi}^\top \mathbf{f}_1 \\ \vdots \\ \bar{\Phi}^\top \mathbf{f}_q \end{bmatrix} \right\|_2^2$$
14: $\mathbf{C} \leftarrow \mathsf{diag}([c_{11}, \dots, c_{kk}]^\top)$
    // Step 5: Matching by linear assignment
15: $\mathbf{N} \leftarrow$ Matching of columns of $\bar{\Phi}^\top$ to those of $\mathbf{C}\hat{\Psi}^\top$
16: return $\mathbf{N}$
```

### 4.4.6 Our algorithm: GRASP

Putting it all together, GRASP consists of five steps, as Algorithm 1 shows in pseudocode.

**Steps 1: Compute eigenvectors.** In the first step, calculate the Laplacians $\mathscr{L}_1, \mathscr{L}_2$ of the two graphs $G_1$ and $G_2$. Then compute the eigenvectors $\Phi, \Psi$ and eigenvalues $\Lambda_1, \Lambda_2$ by the eigendecomposition $\mathscr{L}_1 = \Phi\Lambda_1\Phi^\top$ and $\mathscr{L}_2 = \Psi\Lambda_2\Psi^\top$.

**Step 2: Compute corresponding functions.** In the second step, calculate the matrices of corresponding functions $\mathbf{F} = [\mathbf{f}_1, \dots, \mathbf{f}_q]$ and $\mathbf{G} = [\mathbf{g}_1, \dots, \mathbf{g}_q]$ as diagonals of the heat kernel at time steps $[t_1, \dots, t_q]$ with $\mathbf{f}_i = \sum_{j=1}^{n} e^{-t_i\lambda_j}\phi_j \odot \phi_j$ and $\mathbf{g}_i$ equivalently using $\Psi$.

**Step 3: Base alignment.** After the corresponding functions are calculated, obtain the base alignment matrix $\mathbf{M}$ by minimizing Eq. 4.3. Then align the first $k$ columns of $\Psi$, denoted by $\bar{\Psi}$ to the corresponding first $k$ columns $\bar{\Phi}$ of $\Phi$ as $\hat{\Psi} = \bar{\Psi}\mathbf{M}$.

**Step 4: Calculate mapping matrix.** Under the assumption that $\mathbf{C}$ is a diagonal matrix, calculate its diagonal elements $c_{11}, \dots, c_{kk}$ by solving the least squares problem:

$$\min_{[c_{11},\dots,c_{kk}]^\top} \left\| \begin{bmatrix} \mathsf{diag}(\mathbf{g}_1^\top\hat{\Psi}) \\ \vdots \\ \mathsf{diag}(\mathbf{g}_q^\top\hat{\Psi}) \end{bmatrix} \begin{bmatrix} c_{11} \\ \vdots \\ c_{kk} \end{bmatrix} - \begin{bmatrix} \bar{\Phi}^\top\mathbf{f}_1 \\ \vdots \\ \bar{\Phi}^\top\mathbf{f}_q \end{bmatrix} \right\|_2^2 \tag{4.5}$$

We then set $C = \mathsf{diag}(c_{11}, \dots, c_{kk})$.

**Step 5: Node alignment.** To get the final node alignment, we apply a linear assignment algorithm on the rows of $\bar{\Phi}$ and $\mathbf{C}^\top\hat{\Psi}$, which hold the indicator function coefficients.

**Complexity analysis** The computation of the first $k$ Laplacian eigenvectors takes $\mathscr{O}(k\max\{|E_1|, |E_2|\})$ by fast methods for diagonally dominant matrices [68]. Base alignment needs $\mathscr{O}(k^3)$ to solve the orthogonality constraint through trust-region methods. The least-squares method runs in $\mathscr{O}(qk)$. The matching step by JV runs in $\mathscr{O}(n^3)$. Overall, the $\mathscr{O}(n^3)$ time factor is dominant.

**Connection to Differential Geometry** Our work rests on the theory on Riemannian manifolds [41] and builds on the analogy between a graph's Laplacian and the continuous Laplace-Beltrami operator [135].

## 4.5 Boosting GRASP

Having introduced GRASP, we observe that most state-of-the-art methods for graph alignment leverage advances in graph representation learning [47], so that, rather than directly aligning nodes, these algorithms compute node representations using graph embeddings and align those representations instead. Building on this observation, we outline a *modular framework* that characterizes such methods, and suggest an alternative materialization of the building blocks of GRASP within this framework.

A naïve approach would embed the two graphs in a common space and align nodes based on their proximity. Yet, an embedding of one graph is not necessarily comparable to that of another graph, as the two embedding spaces may be rotated, shifted, or stretched with respect to each other. To obtain comparable embeddings, we have to align the two embedding spaces.

Section 4.5.1 introduces the framework of modular graph alignment algorithms to which GRASP belongs. Then, we propose extensions to GRASP by virtue of its modular structure.

### 4.5.1 Modular Graph Alignment

GRASP belongs to a family of graph alignment algorithms that expose a *modular* structure [76]. Besides GRASP, the recently proposed CONE [19] and its predecessor REGAL [48] are also modular algorithms. Such algorithms tackle the alignment problem in the following three steps.

**Step 1: EMBED** computes vector representations of nodes. A good representation encodes the connectivity structure or neighborhood [109]. For GRASP, we choose the spectral embeddings as described in Section 4.4.4. CONE uses NetMF [111] embedding and optimizes for node neighborhood consistency. REGAL proposes embeddings that take into account the node degrees of the node's neighborhood.

**Step 2: ALIGN** alters the node each graph's embeddings, so that corresponding nodes have comparable embeddings. If the embeddings are computed on each graph individually, only the relations of nodes within the graph are taken into account. This step accounts for changes in the embeddings of one graph that potentially alter the node matching across graphs. Modular methods realign the embedding spaces. GRASP, achieves embedding alignment through the base alignment process in Section 4.4.5. CONE aligns the embeddings by iteratively solving for node correspondence and embedding space correspondence. REGAL maintains relative distances to a set of anchor nodes so as to reduce the impact of comparisons based on the absolute position of the embeddings.

**Step 3:** **ASSIGN** matches node representations so as to minimize a cost function; this step corresponds to a *linear assignment* between the nodes in one graph to those nodes in the other graph. CONE and REGAL employ a nearest neighbor matching scheme, while GRASP implements a variant of the Hungarian algorithm [57].

**Monolithic Alignment Algorithms** In this context it is worth pointing out that two state-of-the-art graph alignment algorithms, S-GWL [145], and LREA [95] do not fit in the modular framework, as they tackle the problem in steps and return alignments in a *monolithic* fashion. In particular:

**S-GWL.** S-GWL jointly computes embeddings and alignments, expressed through a single objective. Therefore, it does not allow for substituting one embedding or assignment algorithm for another.

**LREA.** LREA first computes embeddings of an edge-matching matrix and then matches nodes among graphs. The alignment step and the assignment steps are intertwined by means of a low-rank approximation, and are therefore inseparable.

In the following, we take advantage of GRASP's modularity to devise improvements on it that change the operations in its modular components.

### 4.5.2 PageRank (PR)

We develop an embedding based on PageRank [104]. PR is real-valued vector that represents the importance of each node, defined as $\mathbf{r}_j = \sum_{(i,j)\in E} \alpha \frac{\mathbf{r}_i}{d_i} + (1-\alpha)\mathbf{p}$, where $\alpha \in [0,1]$ is the damping factor and $p_i = 1/n$ for each $i$. We generate corresponding functions sampling PR at different $\alpha$ values. In addition, we use *Personalized PageRank* (PPR), which defines a non-uniform restart probability vector $\mathbf{p}$. By setting the restart probability to 1 on node $i$ and 0 elsewhere, the PageRank score measures the relevance of other nodes to node $i$. We use PPR in our corresponding function to measure proximity with respect to other nodes or groups of nodes, thus enriching the embeddings. First, we partition the nodes into $q$ groups of size $t$ by their PageRank values; the first group contains the $t$ nodes with the highest PageRank, the second group contains the $t$ nodes with the second-highest PageRank, and so on. Then, we calculate PPR values for each group, with the personalization vector evenly distributed among the nodes in the group, and set a node's corresponding functions as its PPR values for each of the $q$ groups. Algorithm 2 describes this process.

**Algorithm 2** PPR-based Corresponding Function

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

### 4.5.3 Iterative Closest Point

The Iterative Closest Point (ICP) method [12] aligns two point clouds, i.e., sets of $n$-dimensional data points. ICP conventionally minimizes the difference between the sets of points by fixing one of the clouds as the target $\mathbf{X}$ and iteratively transforming the source cloud $\mathbf{P}$. The algorithm starts with an initial alignment $\mathbf{Y}$ between the points of each cloud and proceeds transforming the source points with a mapping matrix that minimizes the least-squared difference between $\mathbf{P}$ and the alignment $\mathbf{Y}$. This matrix is obtained by solving the orthogonal Procrustes problem (OPP) [123]. These steps are repeated until the process converges when the change in mean-squared error (MSE) after applying each step is below a chosen threshold $\varepsilon$. Time complexity is dominated by the closest point algorithm, which is $\mathscr{O}(n \log n)$ using a $k$-d tree [12]. Algorithm 3 illustrates the method.

**Algorithm 3** Iterative Closest Point

```
1: repeat
2:     $k = k + 1$.
3:     Compute $k$-th alignment matrix $\mathbf{Y}_k = ClosestPoint(\mathbf{P}_k, \mathbf{X})$.
4:     Find mapping matrix $\mathbf{C}_k = OPP(\mathbf{P}_k, \mathbf{Y}_k)$.
5:     Rotate $\mathbf{P}_k$ via $\mathbf{C}_k$ to obtain $\mathbf{P}_{k+1} = \mathbf{C}_k \mathbf{P}_k$.
6: until $MSE(\mathbf{P}_k, \mathbf{Y}_k) - MSE(\mathbf{P}_{k+1}, \mathbf{Y}_{k+1}) < \varepsilon$
```

We apply ICP in GRASP to refine the initial node alignment (mapping) matrix $\mathbf{C}_0$ that we obtain as in Section 4.4.3.

**Algorithm 4** Voting

```
1: Pick a number of different $k$'s. Compute alignments for all the $k$'s.
2: For each node collect all matches (number of different $k$'s where this match is
   chosen).
3: For each node add the most popular matches to the final alignment.
```

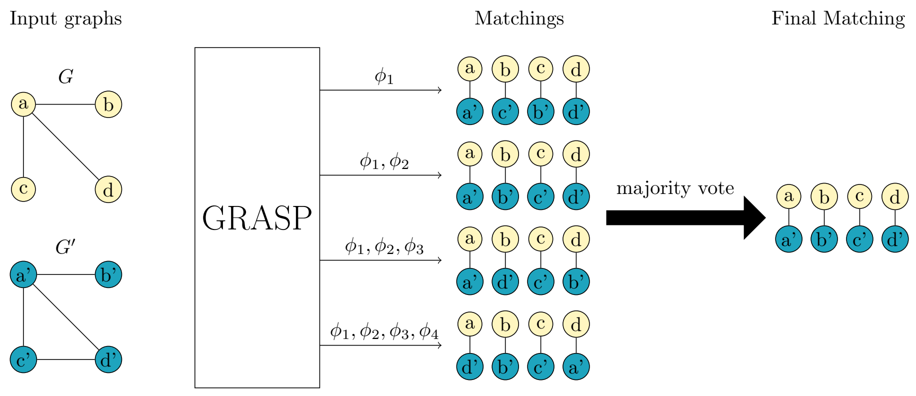


Figure 4.3: An example of the voting heuristic in action; GRASP runs four times, adding one eigenvector in each iteration, from $\phi_1$ to $\phi_4$. We omit eigenvectors $\psi_1, \ldots, \psi_4$ for the sake of readability. Each set of eigenvectors returns a matching from nodes $a,b,c,d$ to $a',b',c',d'$; on node $a$ the match $a$–$a'$ is selected by 3 out of 4 eigenvector sets and chosen as the output match.

### 4.5.4 Voting Heuristic

We observe that the parameter $k$ in GRASP that determines the number of eigenvectors used to approximate each vertex function (Section 4.4.3) has a significant impact on accuracy when GRASP is equipped with PageRank, even though it does not affect results significantly in the heat kernel case. Starting out from this observation, we devise a *voting heuristic* that exploits this variance. We compute alignments for different values of $k$, while using ICP to further align the embeddings in each iteration, and then treat each match a node obtain in any alignment as a vote. Based on those votes, we assign matched pairs in an one-to-one manner, by a heuristic called SortGreedy [32] that sorts matches by their order of frequency among all values of $k$ and processes them in that order to assign each node to its most frequent still-available match. Figure 4.3 shows a running example of the voting heuristic. Algorithm 4 shows the procedure.

This voting procedure can be time consuming, as it requires solving the problem once for each chosen value of $k$. Still, we only need to compute the eigendecomposition of the Laplacian once. We further save time by only computing the base alignment once for the largest value of $k$ and considering submatrices of the base alignment matrix, each time using only the rows and columns up to the current value of $k$. This simplification brings a significant speedup with a negligible accuracy penalty. However, this speedup is more prominent when using the heat kernel rather than PageRank as corresponding function, as the base alignment minimization runs for significantly more iterations with the heat kernel.

## 4.6 Experiments

In this section, we present our thorough experimental study. Table 4.1 gathers the characteristics of the 11 real-world network data sets we use; for three of those (Voles, MultiMagna and HighSchool), we have real-world network variants and ground-truth alignments. On others, unless stated otherwise, we randomly permute the node order in the original graph and inject noise in both graphs by deleting edges with probability (noise level) $p$ ranging from 0.05 to 0.25. Such noise injection has been used before [48, 70]; we render it more challenging by deleting edges in both graphs. For each noise level, we create 5 graphs and report average accuracy in terms of matching ground truth nodes, as in [48, 70]; note that none of the noisy graphs in a pair is a subset of the other. We run experiments on an Intel Core i9 10940X 3.3GHz 28-Core CPU with 256GB RAM.

| **Dataset** | $\lvert V \rvert$ | $\lvert E \rvert$ | **Type** | **GT** | **Bipartite** |
|---|---|---|---|---|---|
| Arenas Email [73] | 1 133 | 5 451 | communication | ✘ | ✘ |
| Facebook-ego [78] | 4 039 | 88 234 | social | ✘ | ✘ |
| CA-AstroPh [78] | 17 903 | 197 031 | collaboration | ✘ | ✘ |
| Hamsterer [73] | 2 000 | 16 097 | social | ✘ | ✘ |
| PPI [12] | 3 852 | 38 705 | biological | ✘ | ✘ |
| Voles [28] | 712 | 2391 | proximity | ✔ | ✘ |
| MultiMagna [138] | 1004 | 8323 | biological | ✔ | ✘ |
| HighSchool [39] | 327 | 5818 | proximity | ✔ | ✘ |
| plantpolrobertson [114] | 1 884 | 15 255 | biological | ✘ | ✔ |
| chowiki [73] | 195 | 352 | co-authorship | ✘ | ✔ |
| tpiwiktionary [73] | 861 | 2 079 | co-authorship | ✘ | ✔ |

Table 4.1: Datasets used in our evaluation, $\lvert V \rvert$ number of nodes, $\lvert E \rvert$ number of vertices, GT ground truth.

**Baselines.** We compare against the following established state-of-the art baselines for *unrestriced* graph alignment.

- **REGAL** [48]: An embedding-based method that utilizes local structural features. In its original formulation, REGAL allows for one-to-many matchings. For the sake of fairness, we let REGAL provide one-to-one matchings using the JV linear assignment algorithm, as GRASP does; we confirmed that, doing so, it fares better than using nearest neighbors.

- **Low Rank EigenAlign (LREA)** [95]: A spectral method that yields one-to-one matchings via the minimization of edge mismatches.

- **CONE** [19]: A modular method that aligns node embeddings of the input graphs. CONE accepts any input embedding and returns node alignments by the nearest neighbor algorithms; for the sake of fairness, we use JV to match the nodes.
- **S-GWL** [145]: A monolithic alignment method that jointly computes embeddings and alignments using optimal transport. S-GWL is a scalable variant of GWL [146].
- **GRAMPA** [36]: A monolithic alignment method that calculates an alignment based on a similarity matrix of embeddings derived from eigenvectors of the adjacency matrix.

**Metric.** We evaluate effectiveness based on the *accuracy* measure: given a set of ground truth alignments $A_{gt}$ and a set of correctly retrieved alignments $A_{cor} \subseteq A_{gt}$, accuracy is:

$$acc = \frac{|A_{cor}|}{|A_{gt}|}$$

**Implementation**. We provide an implementation in Python of GRASP, B-GRASP and the interchangeable components for the modular design of GRASP at `https://github.com/AU-DIS/GRASP`.

### 4.6.1 Parameter tuning.

First we study the impact of the parameters $k$ and $q$ on the quality GRASP achieves, and the core modular algorithmic choices.

**Varying the number of eigenvectors $k$.** The number of eigenvectors affects the quality of the alignment as different eigenvectors capture structures at different scales. The two charts on the left in Figure 4.4 show the accuracy of GRASP on Arenas and Facebook, as a function of the number of eigenvectors at 5% (top line), 10% (middle line), 15% (bottom line) noise level with a fixed number of corresponding functions $q = 100$. We observe that, in both cases, accuracy starts gently decreasing after some value of $k$. This behaviour is expected, as eigenvectors corresponding to larger eigenvalues represent medium-scale to small-scale structures and hence exhibit large noise. Interestingly, the first few eigenvectors can be computed efficiently with power iteration. For the sake of efficiency, we settle for a default value of $k = 20$ in subsequent experiments. **Varying the number of corresponding functions.** The number of corresponding functions $q$ determines the granularity at which we sample the heat kernel (in the default form of GRASP, as in Algorithm 1) linearly

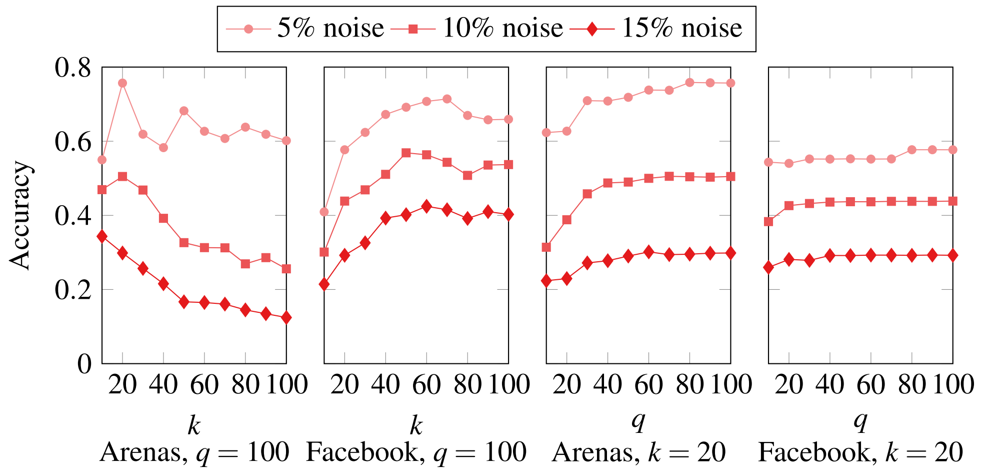


Figure 4.4: Accuracy on the Facebook and Arenas graphs for different numbers of corresponding functions $q$ and different numbers of eigenvectors $k$ for noise levels 5%, 10% and 15%.

over the time interval [0.1, 50]. The more corresponding functions we use, the more precise representation of the graph we get, at a cost of computation time. The two rightmost charts in Figure 4.4 show how the number of corresponding functions $q$ affects accuracy with a fixed number of eigenvectors $k = 20$. With both datasets, quality increases slightly with $q$ at different noise levels. In subsequent experiments, we set $q = 100$.

### 4.6.2 Selecting components

Here, we further exploit the modularity of GRASP (Section 4.5.1), aiming to select best-performing modular components. For the sake of a fair comparison, we also examine whether CONE [19] may benefit from a different choice of embedding other than the default NetMF [111].

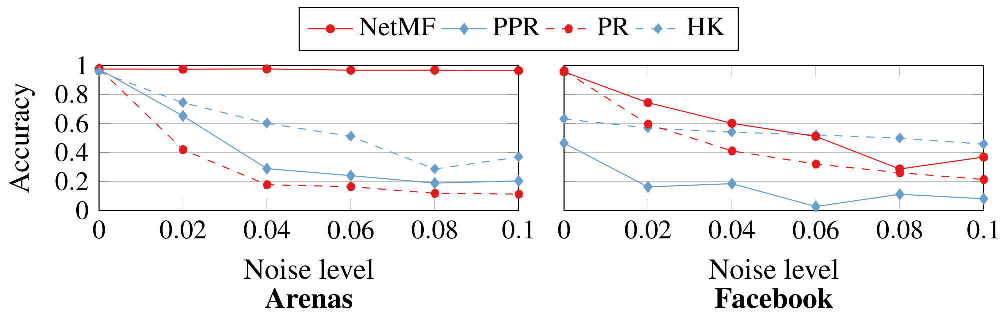


Figure 4.5: Accuracy of CONE-Align with different embeddings on the Arenas and Facebook datasets.

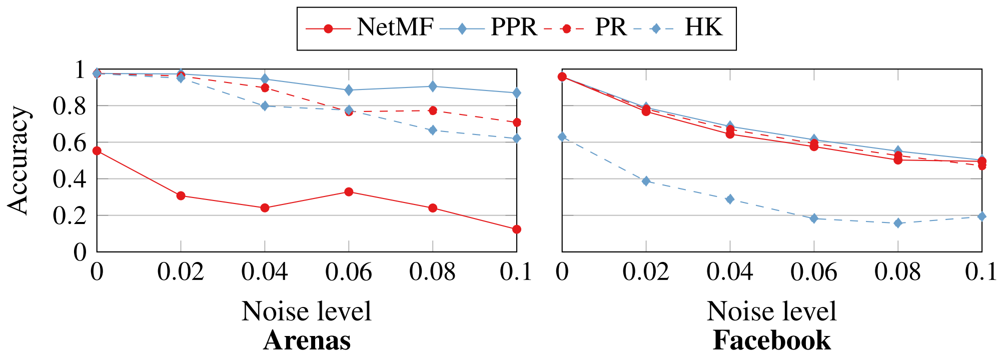


Figure 4.6: Accuracy of GRASP (ICP, voting and JV) with PPR, PR, HK and NetMF embeddings.

**Boosting EMBED.** We first investigate the effect of the choice of node embedding. We try out Heat Kernel (HK), PageRank (PR), Personalized PageRank (PPR), and NetMF. The default embedding in GRASP is the diagonal of the *heat kernel* $\mathbf{H}_t = \Phi e^{-t\Lambda}\Phi^\top = \sum_{j=1}^{n}$ for varying $t$. Figure 4.5 shows the accuracy for CONE on Arenas and Facebook. We observe that NetMF performs best, with an occasional advantage of HK on Facebook. Figure 4.6 juxtaposes different node embeddings for GRASP equipped with ICP, the voting heuristic, and JV. PPR dominates over the other embeddings, while NetMF displays alternating performance on Arenas and Facebook. Henceforward, we use GRASP-PPR as the variant of choice. We note that, as in GRASP embeddings are corresponding functions, thus they affect both EMBED and ALIGN.

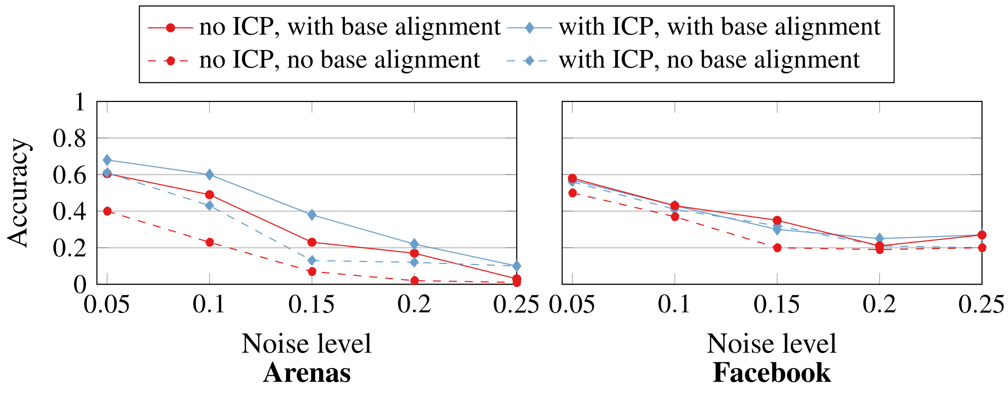


Figure 4.7: Accuracy of GRASP-PPR with ICP and BA.

**Boosting ALIGN.** Next, we investigate the impact of the enhancements in Section 4.5 on GRASP-PPR. Figure 4.7 shows results with and without Iterative Closest Point (ICP) and Base Alignment (BA). We observe that the combination of both ICP and BA yields the best performance.

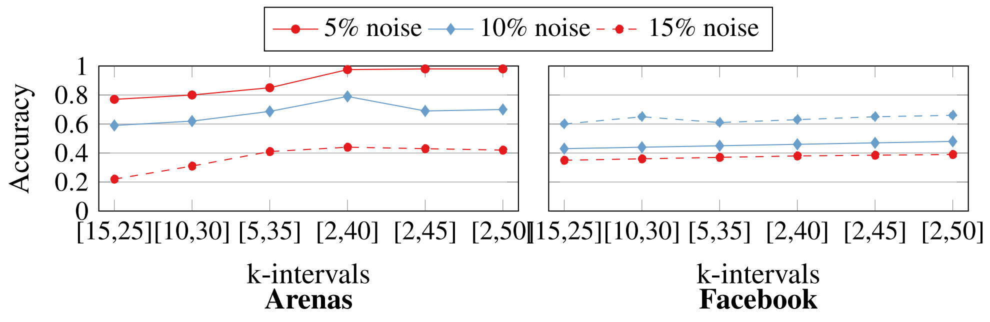


Figure 4.8: Accuracy of GRASP-PPR with ICP using the SortGreedy voting procedure with different intervals of $k$

Figure 4.8 showcases the performance of voting with different amounts of $k$, using SortGreedy (SG) [32] both for node assignment while collecting votes and for the final assignment based on collected votes; other, more computationally demanding choices brought negligible improvements.

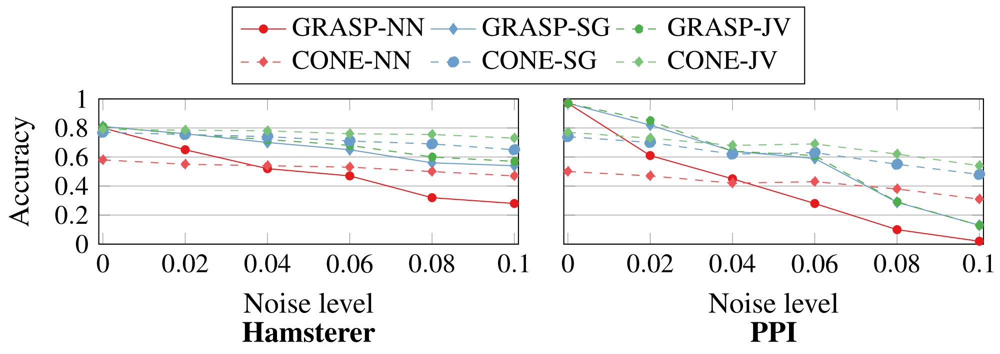


Figure 4.9: Accuracy with different assignment algorithms.

**Boosting Assign.** We also examine the effect of the linear assignment algorithm in more detail. We try out nearest-neighbor (NN), SortGreedy (SG) [32] and Jonker-Volgenant (JV) on the Hamsterer and PPI data. Figure 4.9 shows the results; performance is similar across datasets. SG fares similar to JV, while yielding better runtime. We observe that GRASP-PPR is sensitive on high noise levels, independent of the assignment algorithm, but outperforms CONE on lower noise levels.

**Cross-examination.** Lastly, to further corroborate our findings, we reexamine the combination of two particular modular choices made in the above, namely: (i) the choice of algorithm for node-to-node assignment (Section 4.4.4) and (ii) the usage of base alignment (Section 4.4.5). Figure 4.10 shows that both the JV linear assignment

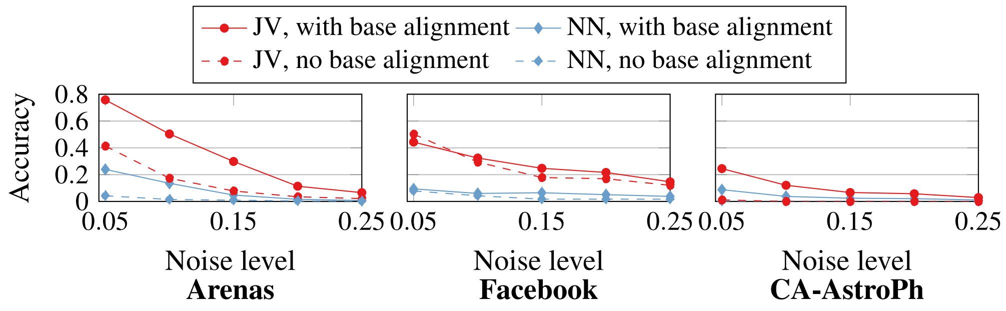


Figure 4.10: Accuracy of nearest neighbor and JV matching algorithms.

algorithm and base alignment bring a substantial advantage over their unrefined counterparts consistently across datasets.

**In conclusion**, based on our inspection of possible enhancements to GRASP, we propose a boosted variant GRASP, which we dub B-GRASP , which uses PPR embeddings, ICP, the JV assignment algorithm, base alignment, and voting in the interval [2,39]. In subsequent experiments we evaluate B-GRASP  extensively.

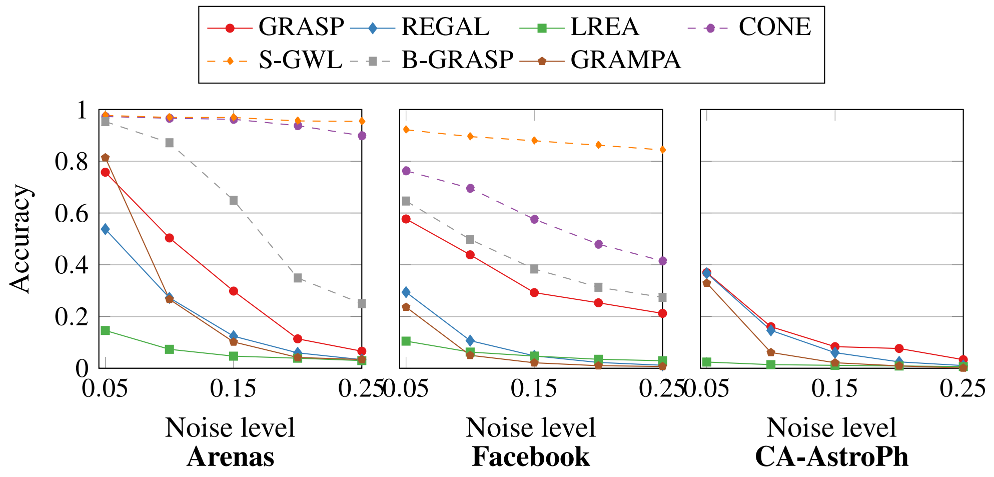


Figure 4.11: Accuracy comparison under synthetic noise.

### 4.6.3 Comparison under synthetic noise

We compare GRASP and B-GRASP  to three scalable methods, namely REGAL [48], GRAMPA [36] and LREA [95], as well as two methods we have found to be non-scalable, namely CONE [19] and S-GWL [145], on real-world data with synthetically generated noise. Figure 4.11 shows that GRASP and B-GRASP  outperform REGAL, GRAMPA and LREA by a large margin, achieving 76% accuracy in Arenas and 59%

in Facebook with 5% noise, and fare at least as well as REGAL on the CA-AstroPH graph. B-GRASP outperforms GRASP on Arenas and Facebook, but falls behind S-GWL and CONE on these graphs. On CA-AstroPh, B-GRASP, CONE and S-GWL exceed 1 hour of runtime and are not listed in the results for this reason. Overall, we find GRASP to be the top-performing algorithm among the scalable methods in this experimental, i.e., among GRASP, REGAL, and LREA.

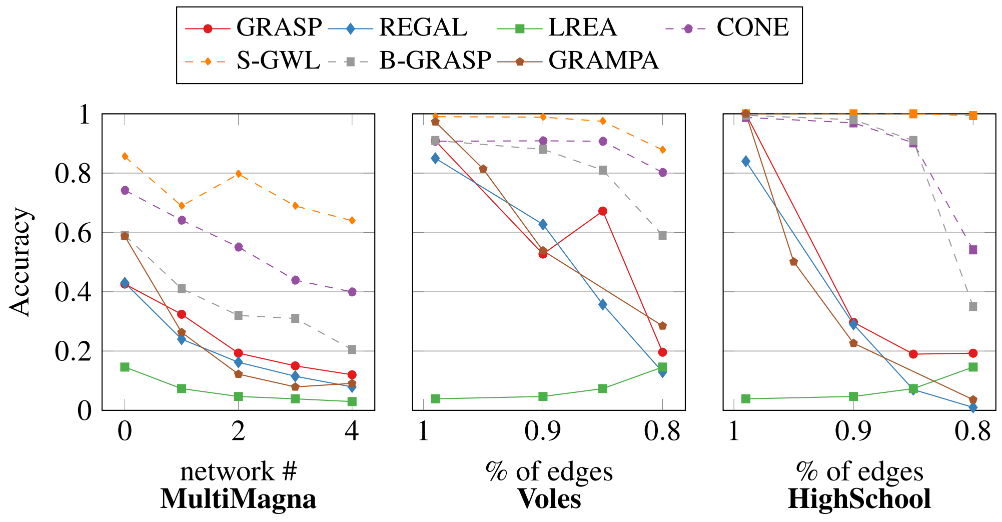


Figure 4.12: Accuracy compared to REGAL on three real datasets.

## 4.6.4 Comparison under real noise

Now we move on to matching among variants of real-world networks. MultiMagna is a collection of graphs consisting of a base yeast network and five variations thereof. We match these five variations to the original. HighSchool and Voles are two evolving proximity networks. We match their latest version to versions at time steps with 80%, 85%, 90%, and 99% of all edges. Figure 4.12 presents our results. We observe that B-GRASP achieves a significant improvement over GRASP as well as LREA and GRAMPA and the boosted version of REGAL with JV, and fares competitively with respect to CONE and S-GWL. We conclude that the advantage of GRASP as the best-performing scalable method, which we observed with synthetic noise, transfers to real-world alignment problems.

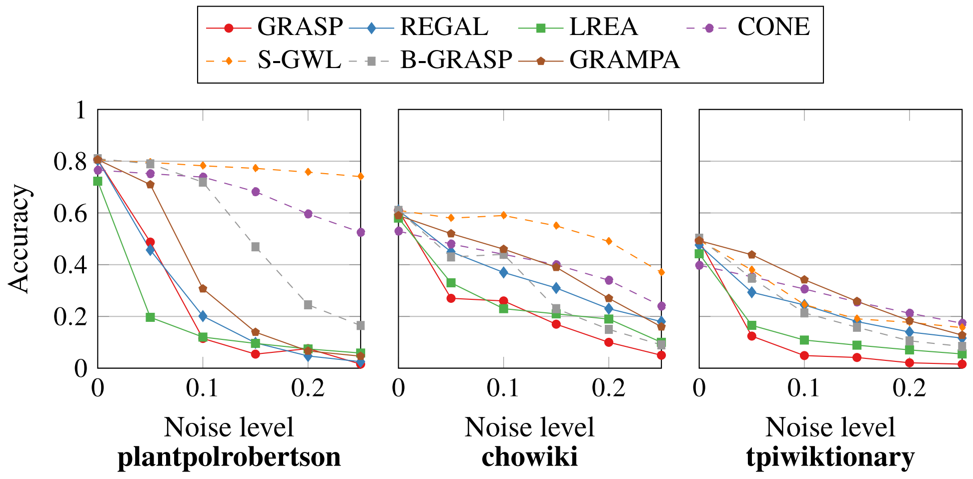


Figure 4.13: Accuracy comparison on bipartite graphs.

### 4.6.5 Bipartite Graphs

We now experiment with the three bipartite graphs; Figure 4.13 presents our results. We observe that GRASP performs well among scalable methods and B-GRASP obtains a significant performance boost over GRASP, while GRAMPA also does well. These results demonstrate the capacity of GRASP and B-GRASP to accommodate this challenging type of graph.

### 4.6.6 Scalability

**Efficiency vs. number of nodes.** In the previous sections, we divided methods into scalable ones (LREA, GRAMPA and REGAL) and non-scalable ones (CONE and S-GWL). Here, we provide experimental evidence for this characterization; we evaluate the running time of all compared methods on a set of random graphs consisting of $2^{10}$, $2^{12}$, $2^{14}$ and $2^{16}$ nodes with an average degree of 10, generated using the configuration model [96] with a degree distribution following a standard normal distribution. Figure 4.14 shows our results on both logarithmic and linear time axis, reporting results for all experiments that terminated within one hour. We observe that GRASP is positioned among the scalable methods, delivering results on graphs with $2^{14}$ nodes in less than 500 seconds. On the other hand, CONE and S-GWL are rendered impractical on large networks.

**Ablation study.** We now turn our attention to the efficiency of the best-of-breed methods, namely B-GRASP (i.e., GRASP with PPR, BA, ICP and voting) and CONE with NetMF embeddings. Figure 4.15 shows the runtime partitioned across the three

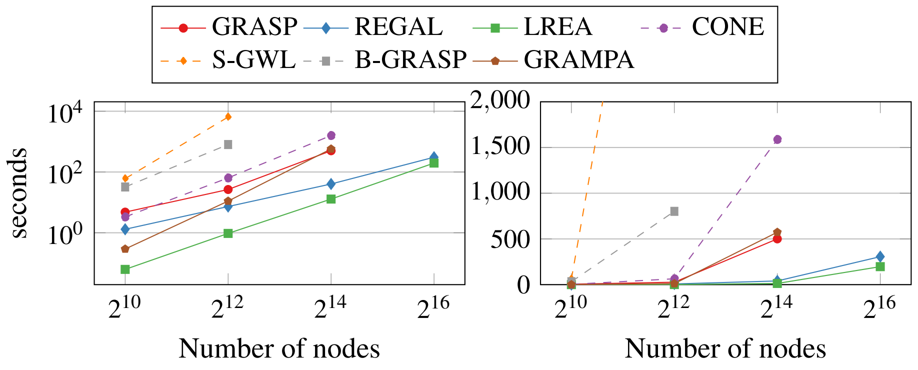


Figure 4.14: Runtime vs. number of nodes on four random graphs of increasing size.

steps. For B-GRASP , we report precomputation including EMBED and ALIGN, and the time for voting. For CONE, we report EMBED, ALIGN and ASSIGN, separately. While CONE is more efficient than B-GRASP on small graphs, its ALIGN turns out to be slower than BA and ICP; therefore, B-GRASP outperforms CONE on large graphs, especially when the number of edges is large, as in the Facebook data set.

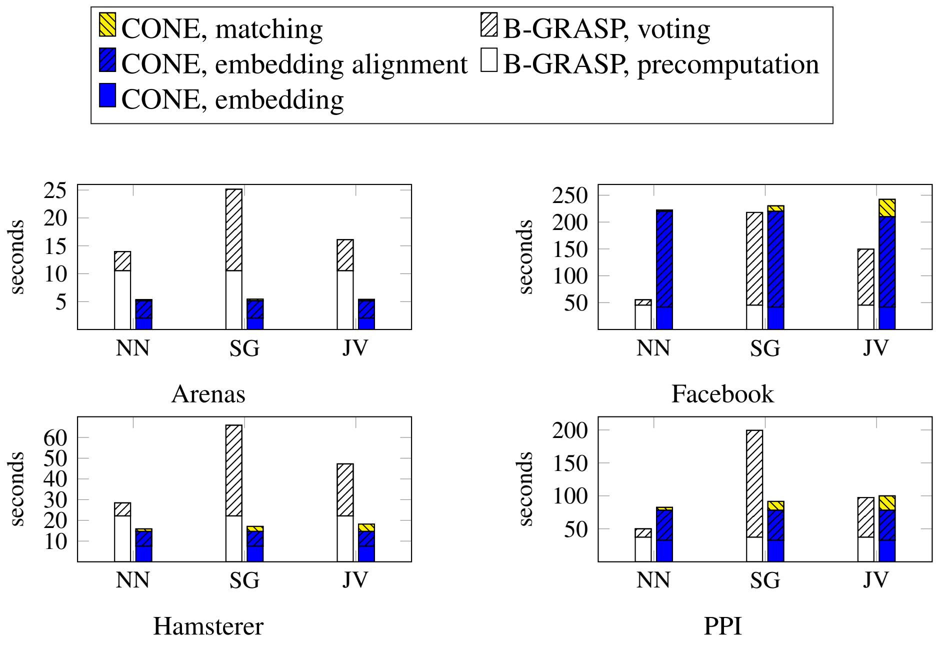


Figure 4.15: Time for B-GRASP and CONE with NetMF on four datasets.

Lastly, we compare the time efficiency of the three most scalable methods, namely GRASP, LREA, and REGAL, in terms of overall time, as well as time excluding the precomputation of information that can be reused.

**Precomputation.** Steps 1–3 of GRASP (Section 4.4.6) can be performed offline; REGAL also allows for precomputation of representations. Figure 4.16 shows the time to compute alignments after precomputation. Remarkably, GRASP outperforms both REGAL and LREA in the largest CA-AstroPh data. Figure 4.17 shows the time including online precomputation; REGAL does not exhibit any substantial advantage even in the smaller Arenas and Facebook graphs, while GRASP attains, as we have seen, more accurate results with a negligible increase in time. We obtained similar results on real-world network matching tasks.

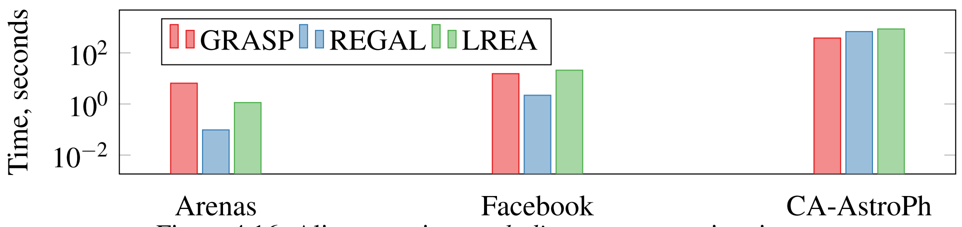


Figure 4.16: Alignment time *excluding* precomputation time.

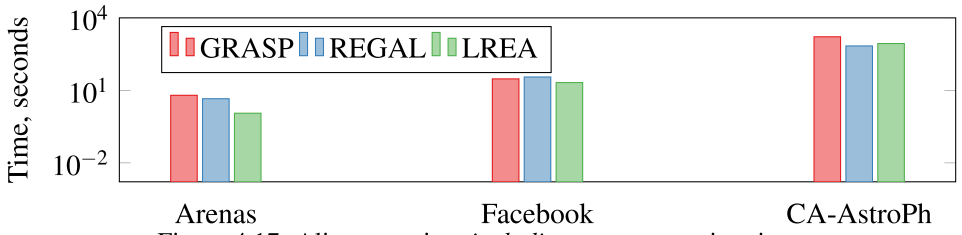


Figure 4.17: Alignment time *including* precomputation time.

### 4.6.7 Impact of degree distribution

The graphs we have used in experiments hitherto approximately follow a power law distribution. We estimate the power law exponents using the method in [97]. Table 4.2 lists the results.

| *Graph* | *Power Law Exponent* ($\gamma$) |
|---|---|
| Arenas | 1.56 |
| Facebook | 1.32 |
| Hamsterster, PPI | 1.45 |
| Voles | 1.64 |
| MultiMagna | 1.46 |
| HighSchool | 1.36 |

Table 4.2: Estimated power law exponents.

As a direction for further study, we examine the impact of the power-law exponent on alignment accuracy. We generate three synthetic power-law graphs with approximate power-law coefficient $\gamma$ 1.71, 2.34 and 3.35. Figure 4.18 shows the performance of GRASP with PPR and JV vs. CONE with JV as a function of noise. GRASP outperforms CONE and achieves nearly 100% accuracy on graphs with low power-law exponent, which is consistent with its good performance in the previous experiments. However, the performance of GRASP drops on graphs with highly skewed distribution as the noise level grows, while CONE with JV manages those cases. This finding illustrates that further work is needed to achieve high alignment accuracy on highly skewed power-law graphs.

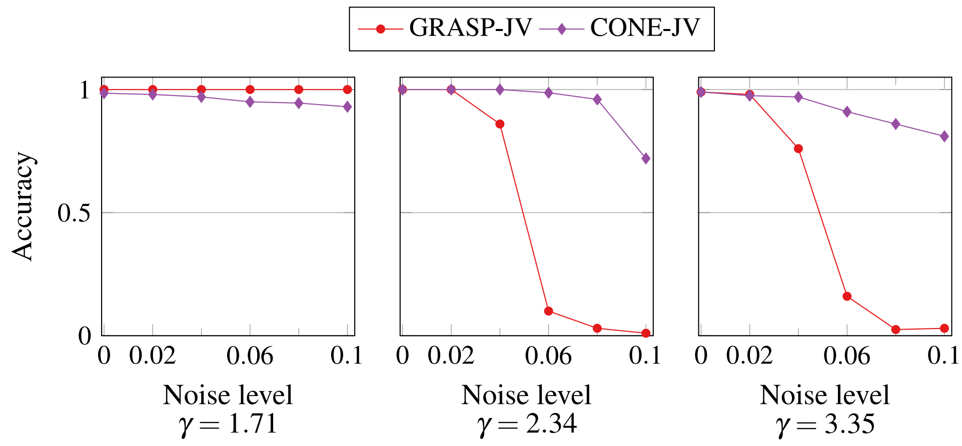


Figure 4.18: Accuracy of GRASP-PPR and CONE with JV assignment on generated power law graphs.

## 4.7 Conclusion

We proposed GRASP, a novel modular graph alignment method that matches graphs utilizing the eigenvectors of their Laplacian matrices. We establish a functional correspondence among the pre-aligned eigenvectors of the two graphs, extending the shape-analysis concept of functional maps and then extract a linear assignment among matrix columns. The functional correspondences we employ capture multi-scale graph properties, and lead to competitive alignment quality over the state-of-the-art scalable methods for graph alignment across noise levels and real-world graph types, while the noise levels we test are higher than anything used in previous studies. To our knowledge, this is the first work to apply a functional alignment primitive to graph alignment.

We expounded upon GRASP's modular design, which allows us to exchange components and develop improved versions thereof. Such an improved version, B-GRASP , delivers alignments that perform competitively compared to far less scalable state-of-the-art methods. We noted that some previous work could also benefit from their modularity, and provide the same advantage to those for the sake of a fair comparison. Eventually, compared to the only scalable algorithms on graphs with more than $2^{14}$ nodes, GRASP delivers the best quality of alignments.

In the future, we plan to extend our method to partial correspondences among graphs, and towards flexible definitions of subgraph isomorphism, including the case of matching graphs with unequal numbers of nodes.

# Chapter 5

# Evaluation of Unrestricted Graph Alignment Algorithms

The work presented in this paper was prepared in collaboration with Konstantinos Skitsas, Karol Orlowski, Davide Mottin and Panagiotis Karras. It has been accepted for publication at EDBT 2023 [127]. We applied minor corrections and adapted formatting and notations.

## 5.1 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.

## 5.2 Introduction

Graphs provide a general means to model relationships between entities in diverse areas of society, science and industry [118], 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 [27], a protein-protein interaction (PPI) network represents the interaction of proteins associated with a biological species [106, 137].

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 $f : V_A \to V_B$. 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 [60] or biological network [102], knowledge of protein sequences in PPI networks [8, 81, 102, 126], special types of graphs [69], or node attributes [155]. 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 [26] or neuroscience [91]; these solutions require additional domain information, such as protein affinity or brain functions. A recent comparative study [134] 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 [134] 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 [134], 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 [134]. 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.

## 5.3 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 $f : V_A \to V_B$ 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, $\mathscr{L} = I - \mathbf{D}^{-\frac{1}{2}}\mathbf{A}\mathbf{D}^{-\frac{1}{2}}$, 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 $\mathscr{L} = \Phi\Lambda\Phi^{\top}$, where $\Lambda$ is a diagonal matrix of eigenvalues; its diagonal, $\{\lambda_1, \ldots, \lambda_n\}$ is the *spectrum* of $\mathscr{L}$ and $\Phi_{\mathscr{L}} = [\phi_1 \phi_2 \ldots \phi_n]$ is a matrix of eigenvectors.

## 5.4 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 [85] or some graph function such as a heat kernel [11]. 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) [18, 48, 145, 146] that returns the unmatched node with the highest similarity, SortGreedy (SG) [66, 126] that progressively matches similarity-sorted pairs of nodes, the Maximum Weighted Matching (MWM) [8, 62, 95] and the Jonker-Volkenant (JW) [50] that directly solve the linear assignment problem.

Table 5.1 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 5.6.2); 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 [126] | 2008 | Yes | Yes | SG | Any | $\mathcal{O}(n^4)$ | $\alpha = 0.9$ |
| GRAAL [72] | 2010 | Yes | No | SG | Any | $\mathcal{O}(n^3)$ | $\alpha = 0.8$ |
| NSD [66] | 2011 | Both | No | SG | Any | $\mathcal{O}(n^2)$ | $\alpha = 0.8$ |
| LREA [95] | 2018 | No | No | MWM | Any | $\mathcal{O}(n \log n)$ | iterations=40 |
| REGAL [48] | 2018 | No | No | NN | Any | $\mathcal{O}(n \log n)$ | $k$=2, $p$=10 log $n$ |
| GWL [146] | 2019 | No | No | NN | Any | $\mathcal{O}(n^3)$ | epoch=1 |
| S-GWL [145] | 2019 | No | No | NN | Any | $\mathcal{O}(n^2 \log n)$ | $\beta \in \{0.025, 0.1\}$ |
| CONE [18] | 2020 | No | No | NN | MNC | $\mathcal{O}(n^2)$ | dim=512 |
| GRASP [50] | 2021 | No | No | JV | Any | $\mathcal{O}(n^3)$ | $q$=100, $k$=20 |

Table 5.1: Algorithms considered in the experiments.

### 5.4.1 IsoRank

IsoRank [126] is the first network alignment algorithm applied on biological networks; it assumes similarity scores are provided by the Blast algorithm [61]. IsoRank uses neighborhood similarity to extract structural graph information, inspired from the way PageRank [16] 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 5.1 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:

$$\mathbf{R}_{ij} = \sum_{u \in N(i)} \sum_{v \in N(j)} \frac{w(i,u)w(j,v)}{\sum\limits_{r \in N(u)} w(r,u) \sum\limits_{q \in N(v)} w(q,v)} \mathbf{R}_{uv} \tag{5.1}$$

The problem expressed by Equation 5.1 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 $\mathbf{R} = \alpha\mathbf{MR} + (1-\alpha)\mathbf{E}$ with parameter $0 \leq \alpha \leq 1$. A value of $\alpha$ 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 [32] heuristic. An extension of IsoRank, IsoRankN [81], solves the global multiple network alignment problem, i.e., aligns multiple networks instead of just two.

### 5.4.2 GRAph ALigner (GRAAL)

GRAAL [72] 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 $\mathcal{O}(n^5)$ 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 $v$:

$$\mathbf{C}_{uv} = 2 - \left( (1-\alpha) \cdot \frac{|N(u)| + |N(v)|}{\max\limits_{i \in V_A} |N(i)| + \max\limits_{i \in V_B} |N(i)|} + \alpha \cdot S(u,v) \right) \tag{5.2}$$

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 [32] method until all nodes are aligned.

### 5.4.3 Network Similarity Decomposition (NSD)

NSD [66] improves upon both the efficiency and topological similarity capture of IsoRank using the HITS [63] 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 [15] by approximating the matrix $\mathbf{R}$ in Equation 5.1 using power iteration, resulting in Equation 5.3, where $\alpha$ 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 $\tilde{\mathbf{A}} = \mathbf{D}^{-1}\mathbf{B}$.

$$\mathbf{X}^{(n)} = (1-\alpha)\sum_{k=0}^{n-1} \alpha^k \tilde{\mathbf{C}}^k h + \alpha^n \tilde{\mathbf{C}}^n h \tag{5.3}$$

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:

$$\mathbf{X}_i^{(n)} = (1-\alpha)\sum_{k=0}^{n-1} \alpha^k w_i^{(k)} z_i^{(k)^\top} + \alpha^n w_i^{(n)} z_i^{(n)^\top} \tag{5.4}$$

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

$$\mathbf{X}^{(n)} = \sum_{i=1}^{s} \mathbf{X}_i^{(n)} \tag{5.5}$$

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 [32].

### 5.4.4 Low-Rank EigenAlign (LREA)

LREA [95] is an alignment method that builds on the EigenAlign [37] 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:

$$\begin{aligned} \underset{\mathbf{y}}{\text{maximize}} \quad & \mathbf{y}^\top M \mathbf{y} \\ \text{subject to} \quad & y_i \in \{0,1\} \\ & \textstyle\sum_u y[u,v] \le 1 \text{ for all } v \in V_2 \\ & \textstyle\sum_v y[u,v] \le 1 \text{ for all } u \in V_1 \end{aligned} \tag{5.6}$$

Matrix $M$ 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. $M$ is rewritten as a linear combination of Kronecker products among the adjacency matrices **A** and **B** of the two graphs and all-one matrices of the same size **E**.

Thus, the quadratic assignment problem in Equation 5.6 is relaxed to the following, using $X$, the leading eigenvector of $M$:

$$\begin{aligned} &\underset{\mathbf{X}}{\text{maximize}} \;\; \mathbf{X} \bullet \left( c_1 \mathbf{A}\mathbf{X}\mathbf{B}^\top + c_2 \mathbf{A}\mathbf{X}\mathbf{E}^\top + c_2 \mathbf{E}\mathbf{X}\mathbf{B}^\top + c_3 \mathbf{E}\mathbf{X}\mathbf{E}^\top \right) \\ &\text{subject to} \;\;\; \|\mathbf{X}\|_F = 1, \mathbf{X} \in \mathbb{R}^{|V_A| \times |V_B|} \end{aligned} \tag{5.7}$$

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.

### 5.4.5 REGAL

REGAL [48] 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:

$$\mathbf{d}_u = \sum_{k=1}^{K} \delta^{k-1} \mathbf{d}_u^k \tag{5.8}$$

Equation 5.8 iterates over $k$-hop neighbors of node $u$ and stores neighborhood degrees in the vector $\mathbf{d}_u$; $\delta$ 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:

$$\mathrm{sim}(u,v) = \exp\left[-\gamma_s \cdot \|\mathbf{d}_u - \mathbf{d}_v\|_2^2 - \gamma_a \cdot \mathrm{dist}(\mathbf{f}_u, \mathbf{f}_v)\right] \tag{5.9}$$

Here, $\gamma_s$ and $\gamma_\alpha$ are scalar parameters that weigh the structural and attribute information, respectively. We set $\gamma_\alpha$ to 0. This cross-network similarity is used to calculate a similarity matrix 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 S as $S^{\sim} = \mathbf{C}\mathbf{W}^{\dagger}\mathbf{C}^{\top}$, 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 S.

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

$$\mathrm{sim}_{emb}(\tilde{\mathbf{Y}}_1[u], \tilde{\mathbf{Y}}_2[v]) = e^{-\|\tilde{\mathbf{Y}}_1[u] - \tilde{\mathbf{Y}}_2[v]\|_2^2} \tag{5.10}$$

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.

### 5.4.6 Gromov-Wasserstein Learning (GWL)

GWL [146] 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.

$$\begin{aligned}\min_{\mathbf{X}_A,\mathbf{X}_B}\ \min_{\mathbf{T}\in\Pi(\mu_A,\mu_B)} & \underbrace{\langle L(C_A(\mathbf{X}_A), C_B(\mathbf{X}_B), \mathbf{T}), \mathbf{T}\rangle}_{\text{Gromov-Wasserstein discrepancy}} \\ & + \underbrace{\alpha\langle K(\mathbf{X}_A, \mathbf{X}_B), \mathbf{T}\rangle}_{\text{Wasserstein discrepancy}} + \underbrace{\beta R(\mathbf{X}_A, \mathbf{X}_B)}_{\text{prior information}}\end{aligned} \tag{5.11}$$

In Equation (5.11), $L$ 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** [145] 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 [143] that decomposes the original GWL non-convex objective into smaller convex problems, achieving an additional speedup on sparse graphs.

### 5.4.7 CONE

CONE [18] finds alignments by focusing on matched neighborhood consistency (MNC), defined in Section 5.6.2 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:

$$\min_{\mathbf{Q}\in\mathscr{O}^d} \min_{\mathbf{P}\in\mathscr{P}^n} \|\mathbf{Y}_A\mathbf{Q} - \mathbf{P}\mathbf{Y}_B\|_2^2 \tag{5.12}$$

$\mathbf{P}$ 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.

### 5.4.8 GRASP

GRASP [50] 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 [135]. The normalized Laplacian, $\mathscr{L} = I - \mathbf{D}^{-\frac{1}{2}}\mathbf{A}\mathbf{D}^{-\frac{1}{2}}$, converges to the eigenfunctions of Laplace-Berltrami operator [10]. 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:

$$H_t = \Phi e^{-t\Lambda}\Phi^\top = \sum_{j=1}^{n} e^{-t\lambda_j}\phi_j\phi_j^\top \tag{5.13}$$

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 $\Phi$ and $\Psi$, using $\mathbf{F}$ and $\mathbf{G}$, and aligns the eigenvector matrices $\Phi$ and $\Psi$ via a diagonal base alignment matrix $\mathscr{M}$ that minimizes the objective:

$$\min_{\mathscr{M}\in\mathscr{S}(n,n)} \operatorname{off}\left(\mathscr{M}^\top\Lambda_2\mathscr{M}\right) + \mu\left\|\mathbf{F}^\top\Phi - \mathbf{G}^\top\Psi\mathscr{M}\right\|_F^2 \tag{5.14}$$

This objective strives for the diagonality of $\mathscr{M}$ 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 $\hat{\Psi} = \bar{\Psi}\mathscr{M}$.

The matching problem is solved as a Linear Assignment Problem (LAP); the authors chose the JV [57] algorithm for this purpose.

## 5.5 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 [105], MAGNA [121], and NETAL [100]. We exclude these methods as they require $\mathcal{O}(n^4)$ 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 [62] due to its $\mathcal{O}(n^5)$ time complexity and NetAlign [8] 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 5.7.1 and the JV assignment algorithm described in Section 5.7.2. Likewise, we exclude methods in which additional information is part of the algorithm definition, such as FINAL [155], BigAlign [69], and the extensive literature on supervised algorithms that require pre-aligned nodes.

## 5.6 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 [45] 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.

### 5.6.1 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.

#### Noise types

In line with previous work [18, 48, 50, 95, 146], 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 [18, 48], adding and removing edges with uniform probability [72, 88, 92, 95, 155], removing and adding nodes [72], generating noise based on the distance between nodes [66] or sampling from the Poisson distribution [146]. Typically, the authors test their methods using

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

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 $G_2$.

### Synthetic Graphs [8, 95]

We generate graphs using popular four graph models to evaluate the algorithms under various characteristics of real graphs.

- **Erdös-Rényi (ER)** [35]: random graphs where edges form with a fixed probability. We generate graphs with probability $p = 0.009$
- **Barabasi-Albert (BA)** [7]: scale-free graphs under the preferential attachment model. We generate graphs with initial node degree $m = 5$.
- **Watts-Strogats (WS)** [141]: 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)** [99]: 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)** [54]: 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 $p = 0.5$.

### Real Graphs

We also use real-world data. Table 5.2 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.

The last three data sets in Table 5.2 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 [50, 95] besides methods designed for biological networks [8, 62, 126] evaluate their performance on ground-truth alignment preferring synthetically generated noise.

| **Dataset** | n | m | $\ell$ | **Type** |
|---|---|---|---|---|
| Arenas [73] | 1 133 | 5 451 | 0 | communication |
| Facebook [78] | 4 039 | 88 234 | 0 | social |
| CA-AstroPh [78] | 17 903 | 197 031 | 0 | collaboration |
| inf-euroroad [6, 116] | 1 174 | 1 417 | 200 | infrastructure |
| inf-power [116, 141] | 4 941 | 6 594 | 0 | infrastructure |
| fb-Haverford76 [116, 132] | 1 446 | 59 589 | 0 | social |
| fb-Hamilton46 [116, 132] | 2 314 | 96 394 | 2 | social |
| fb-Bowdoin47 [116, 133] | 2 252 | 84 387 | 2 | social |
| fb-Swarthmore42 [116, 133] | 1 659 | 61 050 | 2 | social |
| soc-hamsterster [116] | 2 426 | 16 630 | 400 | social |
| bio-celegans [33, 116] | 453 | 2 025 | 0 | biological |
| ca-GrQc [79, 116] | 4 158 | 14 422 | 0 | collaboration |
| ca-netscience [98, 116] | 379 | 914 | 0 | collaboration |
| MultiMagna [138] | 1 004 | 8 323 | 0 | biological |
| HighSchool [39] | 327 | 5 818 | 0 | proximity |
| Voles [28] | 712 | 2 391 | 0 | proximity |

Table 5.2: Information about the real graphs, number of nodes $n$, number of edges $m$, number of nodes left out of the largest connected component $\ell$, and network type.

### 5.6.2 Quality measures

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

#### Matched Neighborhood Consistency (MNC) [18, 95]

is the Jaccard similarity of the mapped neighborhood $\widetilde{N}^f_{G_B}(i) = \{j \in V_B : \exists k \in N_{G_A}(i)$ s.t. $f(k) = j\}$ of a node $i \in V_A$ and a node $j \in V_B$.

$$MNC(i,j) = \frac{\left|\widetilde{N}^f_{G_B}(i) \cap N_{G_B}(j)\right|}{\left|\widetilde{N}^f_{G_B}(i) \cup N_{G_B}(j)\right|} \tag{5.15}$$

The final score is the average MNC among all nodes.

#### 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.

## 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) [105]* 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$^3$)* [121] corrects the deficiencies of EC and ICS with a normalization over both the source and the target graphs

$$\mathrm{S}^3(f) = \frac{|f(E_A)|}{|E_A| + |E(G_B[f(V_A)])| - |f(E_A)|} \tag{5.16}$$

| | *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 5.3: Summary results vs. graph model (first and second best method marked with 🏆), graph size and density.

## 5.7 Results

Here we present the results of our experimental study. Table 5.3 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 $\Delta$ higher than $10^3$ in less than 3 hours and within 256Gb.

### 5.7.1 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.

### 5.7.2 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) [32]; 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) [8, 43, 95] or the JV algorithm [50]. 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 $\{0, 0.01, 0.02, \dots, 0.05\}$ while keeping the graph connected. Figure 5.1 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

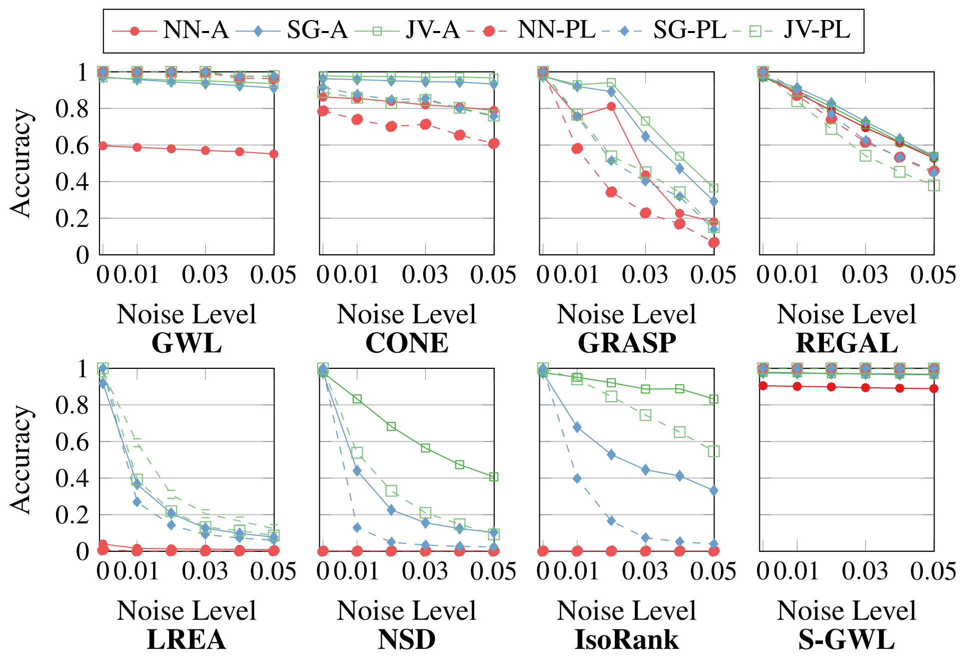


Figure 5.1: Different assignment methods; solid lines: Arenas real data; dashed lines: Power-Law (PL) synthetic data.

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 [8] 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 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.

### 5.7.3 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 5.7.3: Erdös-Rényi (ER) (Figure 5.2), Barabasi-Albert (BA) (Figure 5.3), Watts-Strogats (WS) (Figure 5.4), Newmann-Watts (NW) (Figure 5.5) and Power Law (PL) (Figure 5.6). 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 5.7). 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 5.3) and PL (Figure 5.6). 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 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 scores, both based on neighbor matches, are lower than accuracy. On the other hand, GWL is more robust to different noise types than CONE.

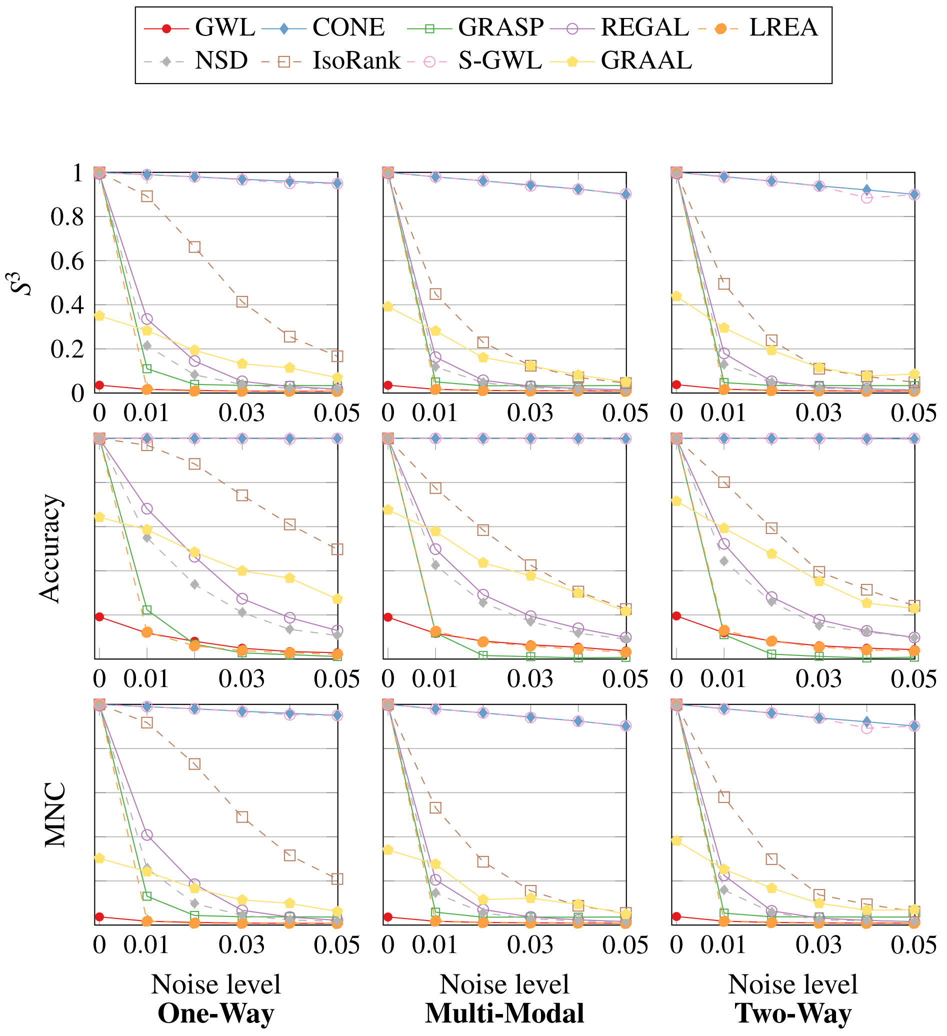


Figure 5.2: Accuracy, $S^3$, and MNC for Erdős-Rényi (ER) random graphs; noise up to 5% and different noise types.

**IsoRank** is among the most competitive algorithms, as opposed to previous comparisons. This is due to an appropriate choice of the weights in Equation (5.9) 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.

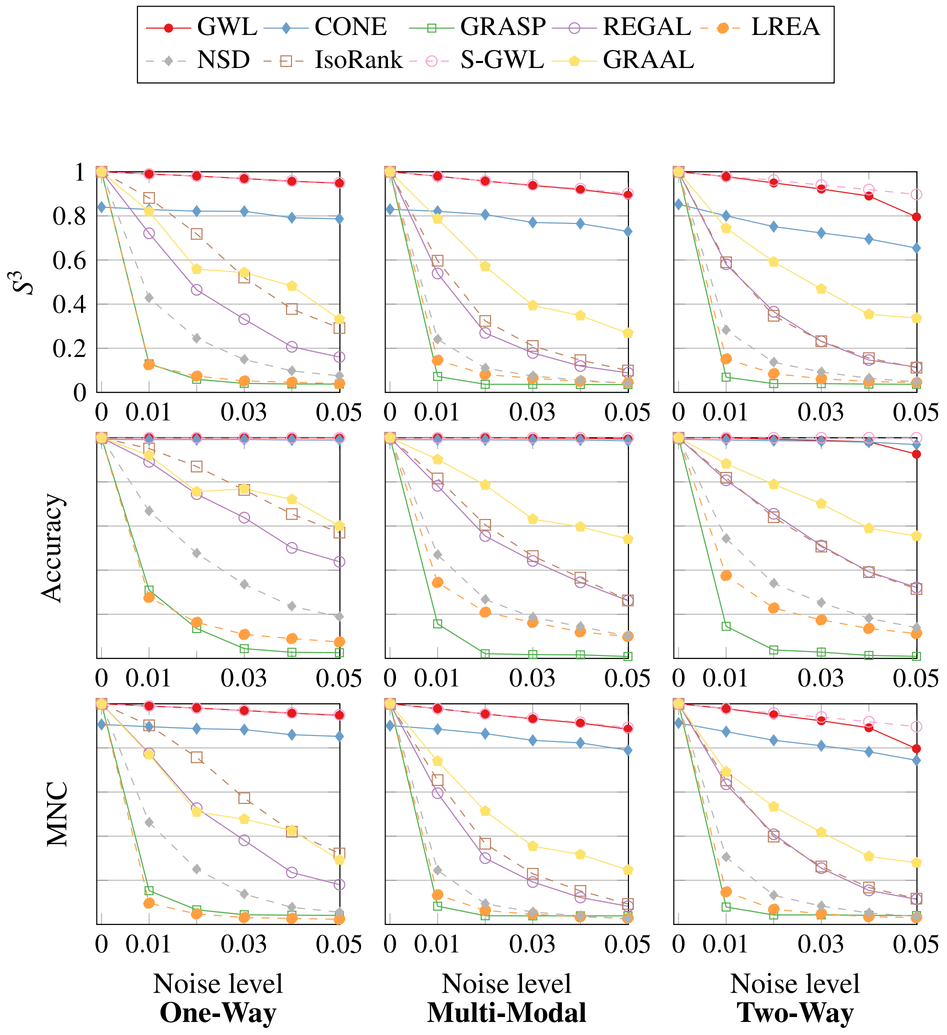


Figure 5.3: Accuracy, $S^3$, and MNC for Barabasi-Albert (BA) synthetic graphs; noise up to 5% and different noise types.

**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

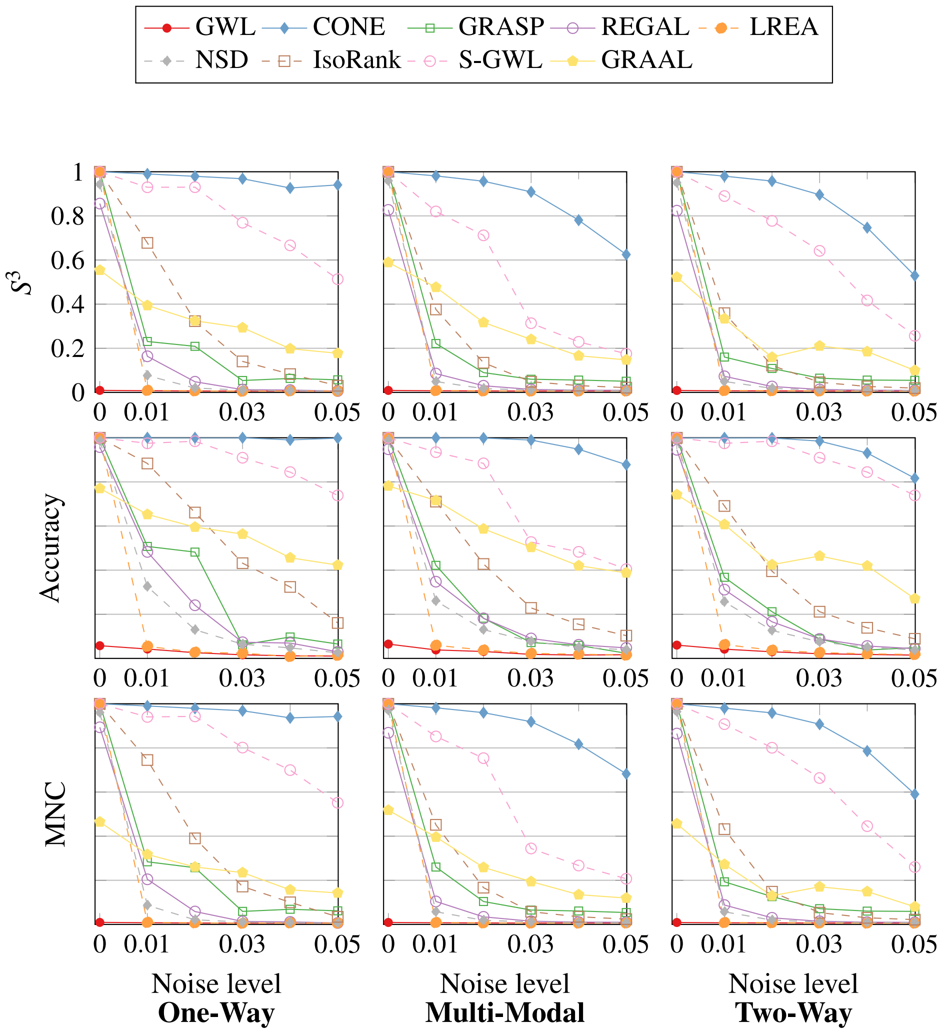


Figure 5.4: Accuracy, $S^3$, and MNC for Watts-Strogats (WS) synthetic graphs; noise up to 5% and different noise types.

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.

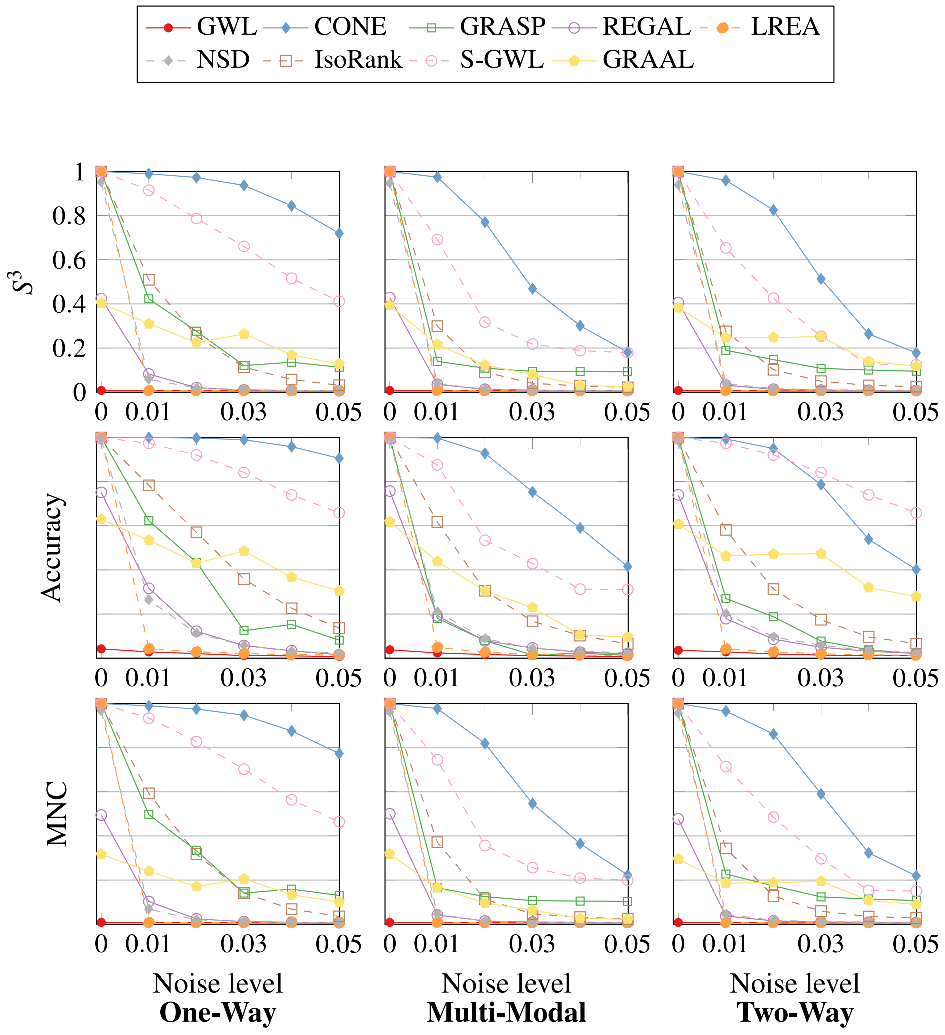


Figure 5.5: Accuracy, $S^3$, and MNC for Newman-Watts (NW) synthetic graphs; noise up to 5% and different noise types.

**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

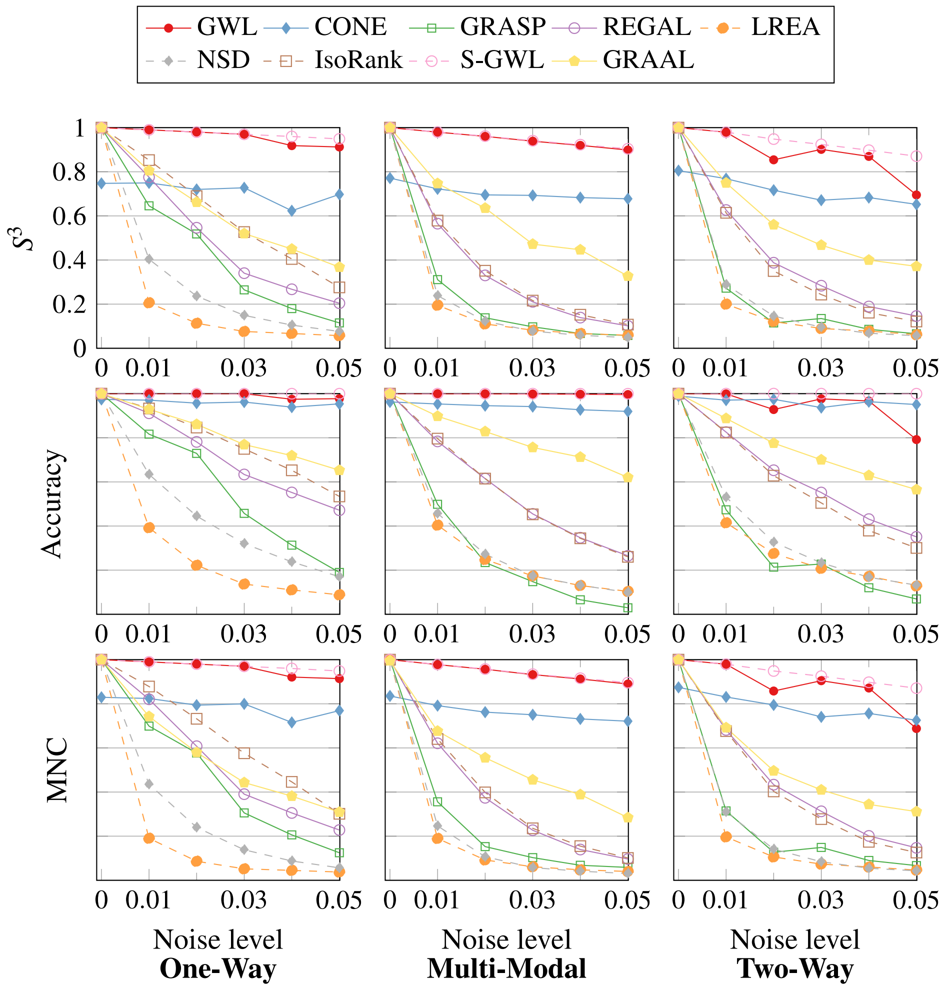


Figure 5.6: Accuracy, $S^3$, and MNC for Powerlaw (PL) synthetic graphs; noise up to 5% and different noise types.

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.

**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 5.7.6.

### 5.7.4 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 1 hour.

#### Low Noise

Figure 5.7 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 1 hour, 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 5.6.1: 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 5.6.1, 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.

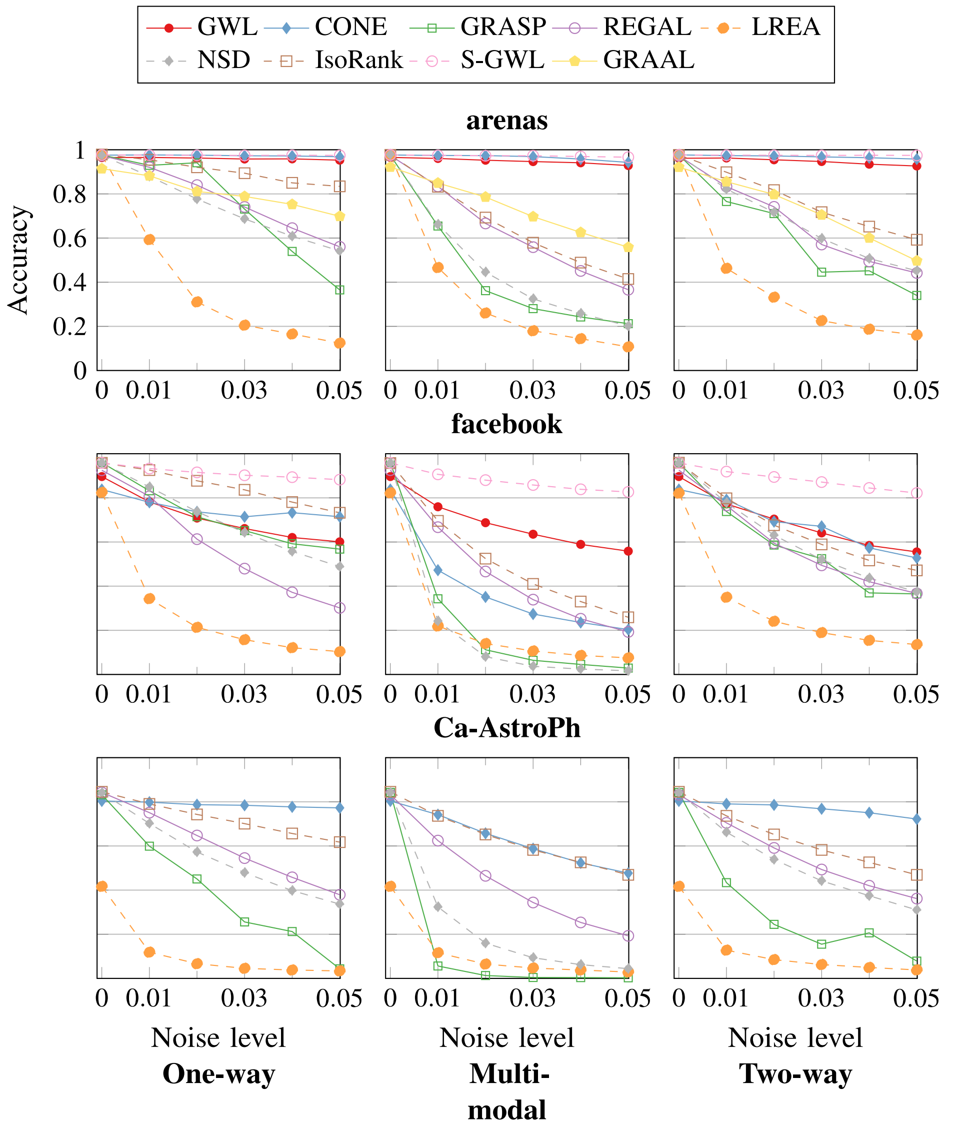


Figure 5.7: Accuracy on real graphs; noise up to 5%.

**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.

**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.

### High Noise

In this experiment we use graphs from the network-repository website [116] shown in Table 5.2. 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 5.8.

**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.

**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

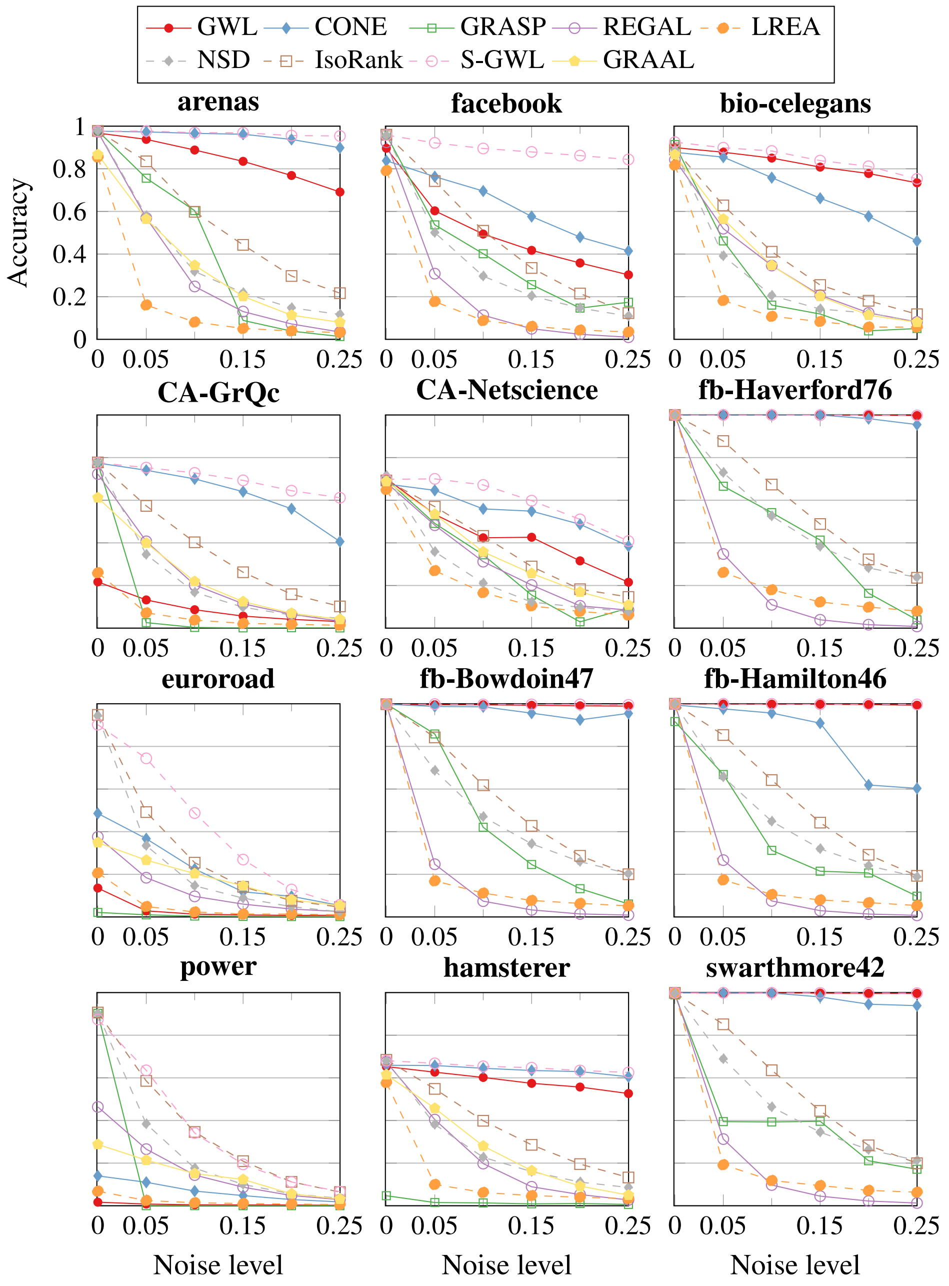


Figure 5.8Accuracy on real graphs; one-way noise up to 25%.

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 5.9 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.

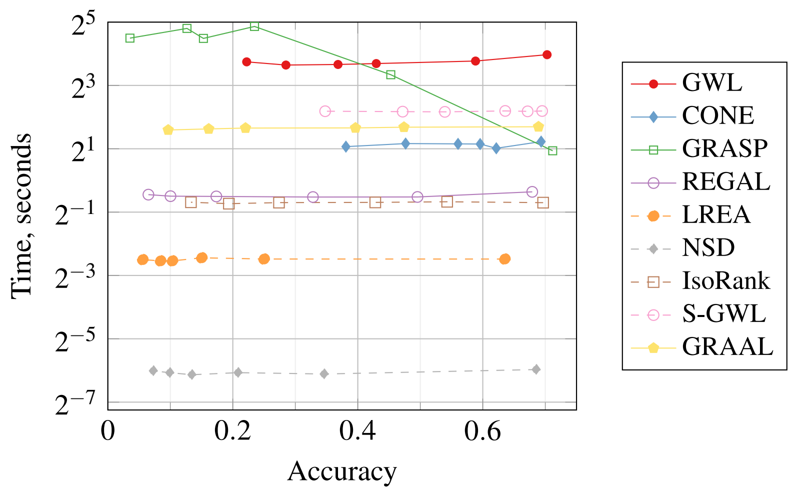


Figure 5.9: Time vs. accuracy on NetScience; marks show results with One-way noise in $[0.25, 0.2, 0.15, 0.1, 0.05, 0]$.

## 5.7.5 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 5.10 shows our results. **GWL** and **CONE** do well in all three measures. GWL perfectly aligns

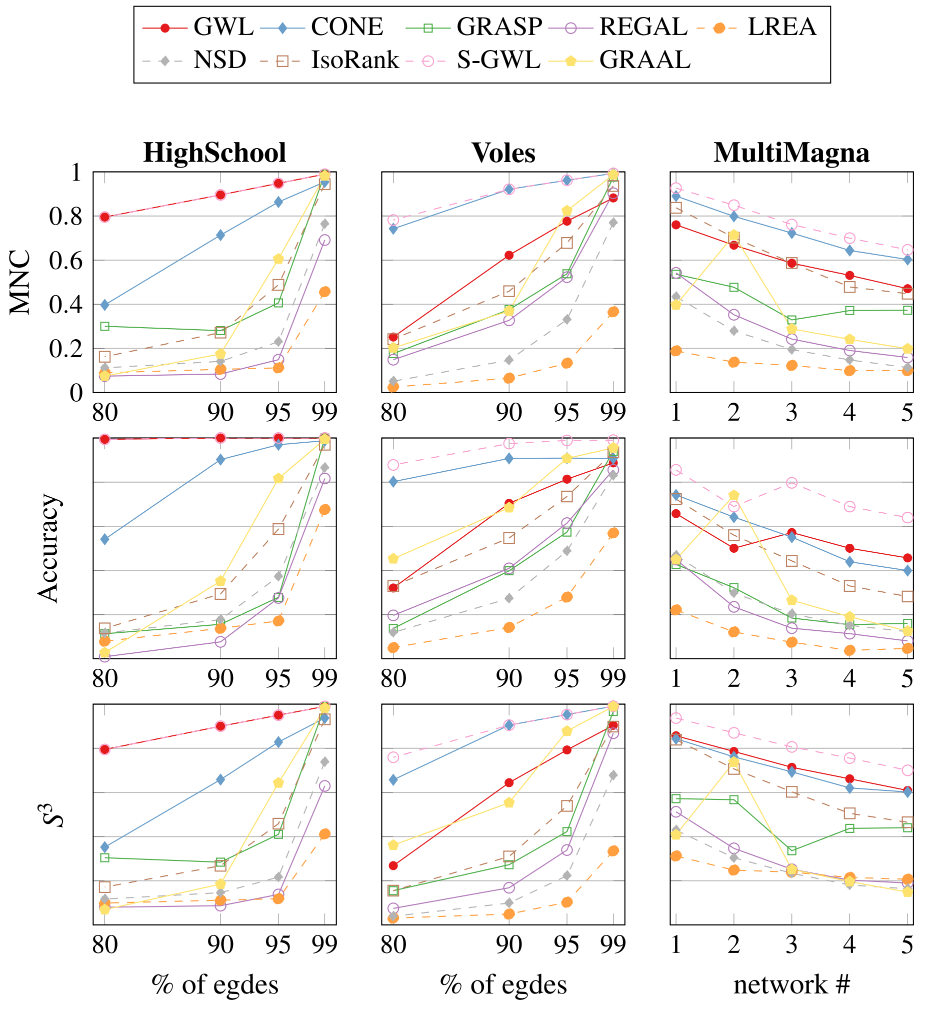


Figure 5.10: Accuracy, MNC and $S^3$ results for the real graphs *HighSchool*, *Voles* and *MultiMagna*.

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**, 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.

### 5.7.6 Scalability

Lastly, we study runtime and memory usage vs. network size and average degree on *configuration model* [96] 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 1 hour. We exclude GRAAL due to its *quintic* ($O(n^5)$) pre-processing time. Figures 5.11 and 5.13 show results when tuning the number of nodes from $2^{10}$ to $2^{16}$ with average degree 10, while Figures 5.12 and 5.14 show results for average degree in $[10, 10^2, 10^3, 10^4]$ with size $2^{14}$.

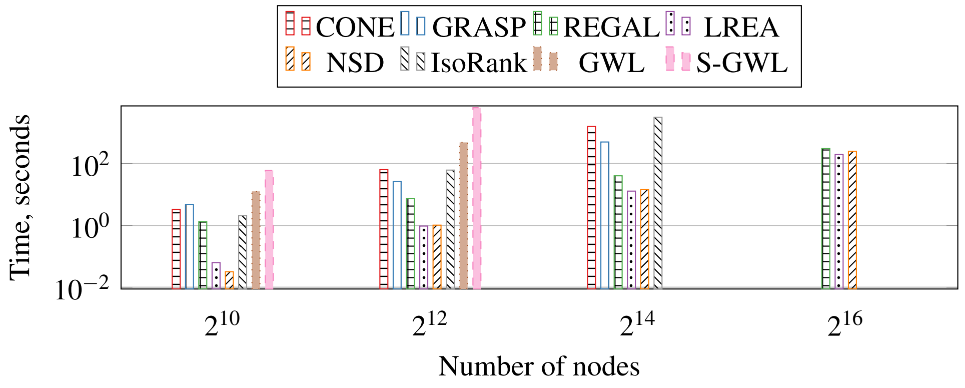


Figure 5.11: Time vs. # nodes; configuration model graphs.

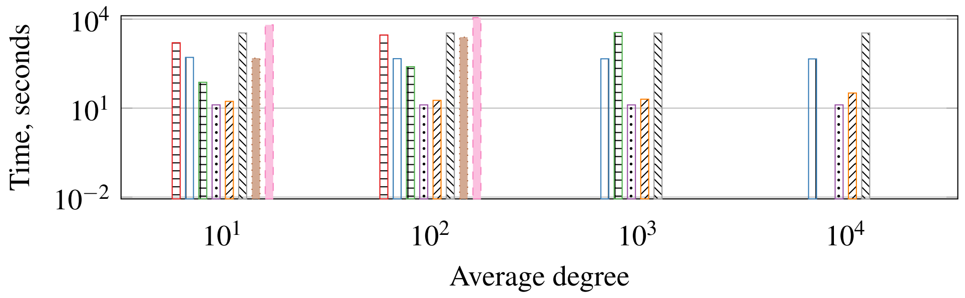


Figure 5.12: Time vs. avg degree, uniform distr.; $2^{14}$ nodes.

By algorithmic complexity, we expect LREA, NSD, and REGAL to be fastest and IsoRank and GWL slowest. The results in Figures 5.11–5.12 confirm our expectation. IsoRank has lower runtime than expected as we let it return a similarity matrix

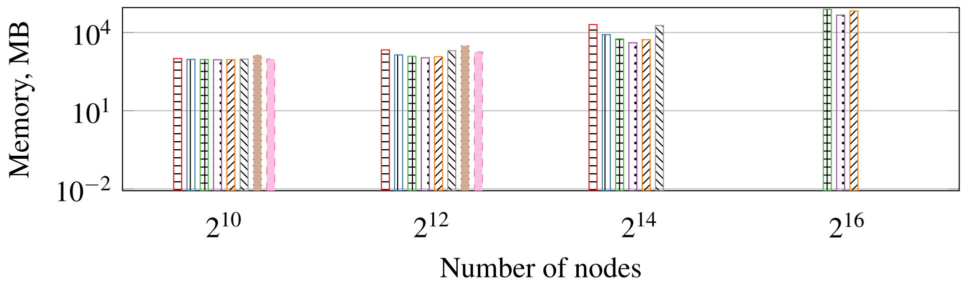


Figure 5.13: Memory vs. # nodes; configuration model.

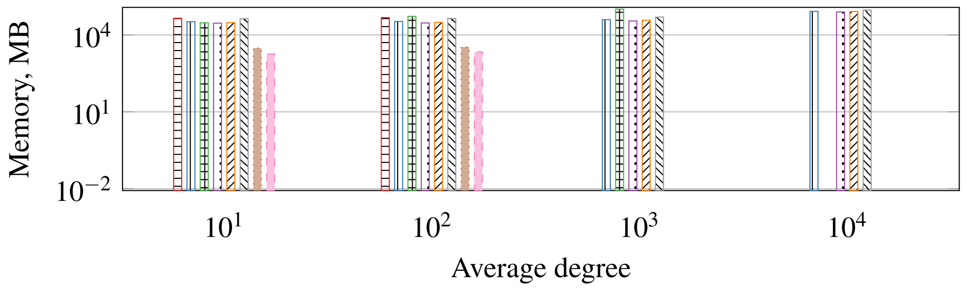


Figure 5.14: Memory vs. avg degree, uniform distr.; $2^{14}$ nodes.

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

### 5.7.7 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 5.15 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

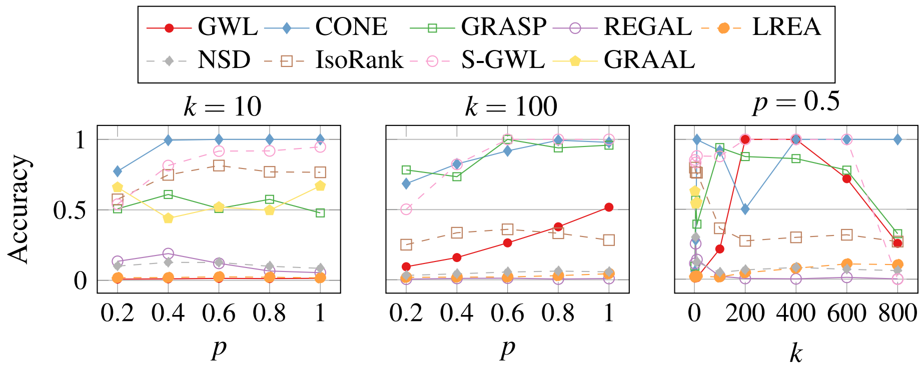


Figure 5.15: Accuracy for 1% One-way noise on Newman–Watts–Strogatz synthetic graphs with 2000 nodes.

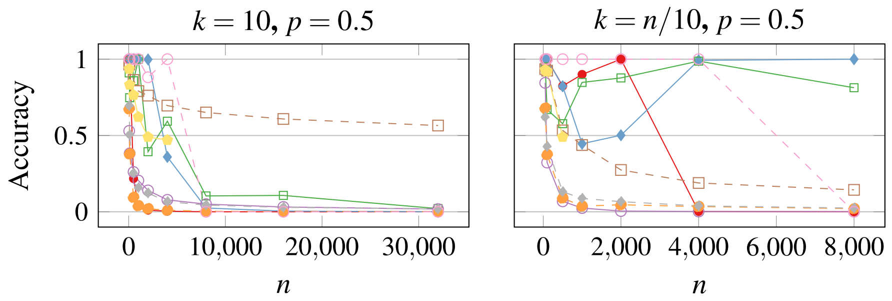


Figure 5.16: Accuracy for 1% One-way noise on Newman–Watts–Strogatz synthetic graphs of increasing size.

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 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 5.10. 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 5.16 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 $\mathbf{M}$ 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 5.16 shows our results. GRASP and CONE manage graphs with variable degree, while S-GWL and GWL, as in Figure 5.15, fail to align networks of either too low or too high average degree.

## 5.8 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.

# Chapter 6

# Spectral Subgraph Localization

The work presented in this paper was prepared in collaboration with Amit Boyarski, Alex Bronstein, Davide Mottin and Panagiotis Karras. It is currently under review for ICDE 2023 [52]. We applied minor corrections and adapted formatting and notation and removed the section on related work, as it is already presented in Chapter 3.

## 6.1 Abstract

Several graph mining problems are based on some variant of the *subgraph isomorphism* problem: Given two graphs, $G$ and $Q$, does $G$ contain a subgraph isomorphic to $Q$? As this problem is **NP**-hard, many methods avoid addressing it explicitly. In this paper, we propose a method that solves the problem by *localizing*, i.e., finding the position of, $Q$ in $G$, by means of an alignment among graph spectra. Finding a node correspondence from $Q$ to $G$ thereafter is relegated to a separate task, as an instance of the *graph alignment* problem. We demonstrate that our spectral approach outperforms a baseline based on the state-of-the-art method for graph alignment in terms of accuracy on real graphs and scales to hundreds of nodes as no other method does.

## 6.2 Introduction

Graph analysis tasks frequently require *localizing* a smaller target graph $Q$ within a larger source graph $G$, i.e., finding a subgraph of $G$ that is best aligned with $Q$. This type of problem may appear as *subgraph discovery* [13, 74], where we need to find any target graph in $G$, in *subgraph querying* [59, 129], where we find out

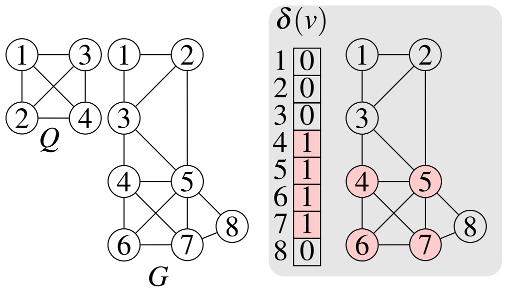


Figure 6.1: An instance of *subgraph localization* (left) and its solution (right); given graphs $Q$ and $G$, we compute an indicator function $\delta(v)$ that reveals the location of $Q$ in $G$.

whether a target subgraph match exists within a collection of source graphs, or *graph matching* [155], where we have to align corresponding nodes across two graphs, potentially of different sizes. Such subgraph localization is of interest in practical applications such as localizing a smaller electronic circuit within a large circuit [40], detecting sub-molecules in bigger molecules [94], and localizing parts of shapes in computational geometry [112]. For instance, the task of *subcircuit detection* [40] involves sampling multiple subgraphs and comparing the spectra of their adjacency matrices to that of the query subgraph. Despite the prevalence of the problem, current research has avoided tackling it directly, due to its **NP**-hardness.

In this paper, we propose a novel *spectral* solution to the problem of subgraph localization, built around the notion of identifying the spectrum $\lambda_Q$ of a graph $Q$ within that of another graph $G$. Figure 6.1 visualizes an instance of the subgraph localization problem by our formulation; we aim to find a function $\delta$ that indicates which nodes in $G$ correspond to $Q$. Our solution effectively recovers both the nodes belonging to the part and the edges that connect the part to the rest of the graph. This problem is an instance of *inverse eigenvalues* problems [21], the class of problems which aim to reconstruct a matrix from its spectrum.

Our experimental study demonstrates that our approach tackles the subgraph localization problem more effectively than state-of-the-art neural competitors and showcases its applicability to the real world problem of *subgraph alignment*.

In summary, our contributions are as follows:

- We propose a spectral formulation for the subgraph localization problem.
- We show that our solution achieves the optimum value under mild conditions.
- We experimentally validate the effectiveness of our solution on real and synthetic graphs.

## 6.3 Subgraph localization

All aforementioned problems have in common the search for one graph within another. We study the most generic form of this problem, which corresponds to the problem named *Subgraph localization* in our previous discussion. That is, we aim to identify a subset of the nodes of a graph $G$ corresponding to an input graph $Q$; we do not aim at an exact 1-to-1 correspondence among all graph elements, but to simply detect a set of best matches.

**Problem 1.** *The* subgraph localization *problem for a graph $G = \langle V, E \rangle$, where $V$ is a set of $n$ nodes and $E \subseteq V \times V$ is a set of edges, and a query graph $Q = \langle V_Q, E_Q \rangle$ with $n_Q = |V_Q|$, $n_Q < n$, calls to find a set of nodes $V_S \subset V$ such that $|V_S| = |V_Q|$ and there exists a bijective function $f : V_S \to V_Q$ between the nodes in $V_S$ and those in $V_Q$ such that for each $(i, j) \in E_Q$ there exists $(f(i), f(j)) \in E_Q$ and vice versa.*

In many applications, solving subgraph localization, we do not need to explicitly materialize the correspondence function $f$. Such a one-to-one correspondence is not explicitly sought for. Thus, we can eschew recovering an exact $f$ and instead aim at finding an indicator function $\delta : V \to \{0, 1\}$ such that:

$$\delta(v) = \begin{cases} 1 & \text{if } v \in V_Q \\ 0 & \text{otherwise} \end{cases} \tag{6.1}$$

At a first glance, finding such an indicator function seems easier than recovering a bijective function $f$. However, even in this identity-function formulation, the problem corresponds to the decision version of the subgraph isomorphism problem, which asks whether a graph $G$ contains a subgraph isomorphic to another graph $Q$. Thus, the problem is still **NP**-complete. Even so, we further relax our requirements, allowing the function $\delta$ to be a binary version of a continuous function $\mathbf{v} : V \to \mathbb{R}$ on values below a threshold $\tau$:

$$\delta(v) = \begin{cases} 1 & \text{if } \mathbf{v}(v) < \tau \\ 0 & \text{otherwise} \end{cases} \tag{6.2}$$

This relaxed problem calls to find a real function, or, equivalently, a real vector $\mathbf{v} \in \mathbb{R}^n$, with $n = |V|$, for a known permutation of nodes in the graph. To overcome the requirement for a known node permutation, we consider a permutation-invariant spectral alignment approach reminiscent of the Hamiltonian operator used in shape analysis [20, 112]. Before delving into the approach, we introduce the necessary notation.

**Background.** The *adjacency matrix* of graph $G$ with $n$ nodes is a $n \times n$ matrix $\mathbf{A} \in \{0,1\}^{n\times n}$ where $\mathbf{A}_{ij} = 1$ if $(i, j) \in E$, 0 otherwise. The *degree matrix* $\mathbf{D}$ is an $n \times n$ diagonal matrix where each entry $d_{ii} = \sum_{j\neq i} \mathbf{A}_{ij}$ holds the degree of node $i$.

The *graph Laplacian matrix* is defined as

$$\mathscr{L} = \mathbf{D} - \mathbf{A}. \tag{6.3}$$

The Laplacian matrix of undirected graphs is a positive semi-definite symmetric matrix, hence its eigenvalues $\lambda_1, \dots, \lambda_n$, are real and non-negative. The *spectrum* $\lambda(\mathbf{M})$ of a matrix $\mathbf{M}$ is the ordered sequence $\lambda_1 \leq ... \leq \lambda_n$ of its eigenvalues. Correspondingly, a graph's spectrum is the spectrum of its Laplacian matrix.

## 6.4 Spectral Subgraph localization

We examine how the presence of a subgraph within a graph affects the graph's spectrum. Spectral theory establishes that the spectrum of a subgraph interlaces with the spectrum of the graph. However, the problem is also non-trivially affected by nodes beside the subgraph. Still, if we could compensate for the effect of nodes other than the subgraph's nodes, the two spectra would be indistinguishable. Following this reasoning, we devise a novel objective for subgraph localization. To that end, we first propose an original connection between subgraph localization and inverse eigenvalue problems with structural constraints [21].

**Inverse Eigenvalue Problem.** The general *additive inverse eigenvalue problem* (AIEP) is defined as follows:

**Problem 2** (AIEP, Problem 3.6 in [21]). *Given an $n \times n$ matrix* $\mathrm{A}$*, a special class of matrices $\mathscr{N}$, and a set of scalars $\{\lambda_{Q_i}\}_{i=1}^k$, find* $\mathrm{X} \in \mathscr{N}$ *such that* $\{\lambda(\mathrm{A}+\mathrm{X})_i\}_{i=1}^k = \{\lambda_{Q_i}\}_{i=1}^k$.

A vast literature on this problem (see [21] and references therein) explores questions regarding the existence of solutions and numerical approximation algorithms for various special classes of matrices $\mathscr{N}$. A common variant of Chapter 2 expresses the problem as a least squares problem between the spectra:

$$\min_{\mathrm{X}\in\mathscr{N}} \|\lambda(\mathrm{A}+\mathrm{X}) - \lambda_Q\|^2. \tag{6.4}$$

In what follows, we establish a connection between the subgraph localization problem (Problem 1) and the additive inverse eigenvalue problem (Problem 2). Under the above formulation, we aim to find a $\mathbf{v}$ that, added to the diagonal of the Laplacian of $G$, renders its first $n_Q$ eigenvalues equal to those of the query graph. In addition to finding $\mathbf{v}$, we aim to remove from $G$ the edges that connect the identified part to the remaining nodes. To the best of our knowledge, this is the first time such a connection has been established, and the first time an AIEP with structural Laplacian constraints is considered.

To devise our solution for subgraph localization, we commence with an intuitive scenario. We assume that $G$ has a number of clearly separated communities, one of which corresponds to the query graph $Q$. A community is defined by a cut, as nodes within the same community are more well connected than nodes across communities. Without loss of generality, assume the graph comprises two distinct communities. In this case, $G$'s Laplacian is a block matrix with two *diagonal* blocks $\mathscr{L}_{11} \in \mathbb{R}^{n_Q \times n_Q}$ and $\mathscr{L}_{22} \in \mathbb{R}^{(n-n_Q)\times(n-n_Q)}$ and only a few entries in the blocks $\mathscr{L}_{12} \in \mathbb{R}^{n_Q\times(n-n_Q)}$ and $\mathscr{L}_{21} \in \mathbb{R}^{(n-n_Q)\times n_Q}$ representing edges across the two communities. The spectra $\lambda(\mathscr{L})$ of $G$ and $\lambda(\mathscr{L}_Q)$ of $Q$ differ on the nodes in $\mathscr{L}_{22}$ and the edges in $\mathscr{L}_{12}$ and $\mathscr{L}_{22}$.

We aim to find a *Hamiltonian* transformation that cancels out this difference. According to [112, Lemma 1], if we add to the diagonal of $\mathscr{L}$ a vector $\mathbf{v}$ having non-zero values, $\mathbf{v}(v) > \tau$, *limited to* nodes in $\mathscr{L}_{22}$, i.e., outside $V_Q$, then eigenvectors corresponding to eigenvalues $\lambda_i < \tau$ of the resulting spectrum $\lambda(\mathscr{L} + \mathsf{diag}(\mathbf{v}))$ will have non-zero values *limited to* the positions corresponding to nodes in $V_Q$, in effect rendering $\lambda(\mathscr{L} + \mathsf{diag}(\mathbf{v}))$ similar to $\lambda(\mathscr{L}_Q)$. Still, the non-zero entries between communities in $\mathscr{L}_{12}, \mathscr{L}_{21}$ affect the spectrum. To cancel that effect, we introduce a *Laplacian editing matrix* that removes the contribution of such edges to the Laplacian of the graph $G$:

$$\mathbf{E} = \begin{bmatrix} -\mathsf{diag}(\mathscr{L}_{12}1) & \mathscr{L}_{12} \\ \mathscr{L}_{21} & -\mathsf{diag}(\mathscr{L}_{21}1) \end{bmatrix}$$

In effect, the corrected Laplacian $\mathscr{L} - \mathbf{E}$ is equivalent to the Laplacian of a graph with two connected components, one of which isomorphic to the query graph $Q$. Thus, the solution $\mathbf{v}$ renders the $|V_Q|$ smallest eigenvalues of the corrected Laplacian indistinguishable from the spectrum of $Q$, $\lambda_Q$, i.e., $\lambda(\mathscr{L} - \mathbf{E} + \mathsf{diag}(\mathbf{v})) = \lambda_Q$, where, with a slight abuse of notation, $\lambda(\mathscr{L} - \mathbf{E} + \mathsf{diag}(\mathbf{v}))$ refers to the $|V_Q|$ smallest eigenvalues of $\mathscr{L} - \mathbf{E} + \mathsf{diag}(\mathbf{v})$.

Since both $\mathbf{v}$ and $\mathbf{E}$ are unknown, we optimize the objective:

$$\begin{aligned}
&\min_{\mathbf{v},\mathbf{E}} \ \|\lambda(\mathscr{L} - \mathbf{E} + \mathsf{diag}(\mathbf{v})) - \lambda_Q\|_2^2 \\
&\text{s.t. } \mathbf{E} = \mathbf{E}^\top, \mathbf{E}1 = 0, \ \mathsf{off}(\mathscr{L} - \mathbf{E}) \leq 0, \\
&\quad \|\mathbf{v}\| = c.
\end{aligned} \tag{6.5}$$

This objective is not convex, yet it only depends on the spectrum, for which there exists efficient approximations [24]; it leads to a solution even if the initial value of $\mathbf{v}$ is a noisy version of the ground truth. As constraints, we postulate that $\mathbf{E}$ should be: (i) symmetric, $\mathbf{E} = \mathbf{E}^\top$; (ii) row- (and, by symmetry, also column-) centered, $\mathbf{E}1 = 0$, with every row summing to 0; and (iii) yielding only non-positive off-diagonal entries $\mathsf{off}(\mathscr{L} - \mathbf{E}) \leq 0$. In addition, we enforce that $\mathbf{v}$ be a point on the surface of a sphere of radius $c$, via the constraint $\|\mathbf{v}\| = c$. Chapter 6.4.1 provides a sufficient condition on $c$ for the optimality of Equation (6.5), considering the noiseless case where $G$ exactly contains the subgraph $Q$.

**Proposition 6.4.1.** *When $c > \sqrt{n - n_Q}\max(\lambda_Q)$, the global optimum of Equation* (6.5) *is obtained at*

$$\mathbf{v} = \begin{cases} 0 & \textit{if } v_i \in V_Q \\ \frac{c}{\sqrt{n-n_Q}} & \textit{otherwise} \end{cases} \tag{6.6}$$

*with*

$$\tilde{\mathbf{v}} = \frac{\mathbf{v} - \min(\mathbf{v})}{\max(\mathbf{v}) - \min(\mathbf{v})}, \tag{6.7}$$

$$S_{ij} = |\tilde{v}_i - \tilde{v}_j| A_{ij}, \tag{6.8}$$

$$\mathbf{E} = \mathsf{diag}(\mathrm{S}1) - \mathrm{S}. \tag{6.9}$$

*Proof.* Let $\mathbf{E}$ be constructed from Equations (6.6)–(6.9). $\mathscr{L} - \mathbf{E}$ is the Laplacian of a graph composed of two disjoint components, one of which is exactly the component indicated by Equation (6.6), i.e., the query subgraph $Q$. Then there is a permutation $\Pi$ such that $\Pi\mathscr{L}\Pi^\top$ is a block diagonal matrix with the Laplacian of each component on the diagonal. Without loss of generality, we assume that the Hamiltonian operator attains this block diagonal form:

$$\mathscr{L} - \mathbf{E} + \mathsf{diag}(\mathbf{v}) = \begin{bmatrix} \mathscr{L}_Q & \\ & \mathscr{L}_{\bar{Q}} + \frac{c}{\sqrt{n-n_Q}} 1 \end{bmatrix}. \tag{6.10}$$

When $c$ satisfies the stated condition, the spectrum of the bottom-right block contains only eigenvalues larger than $\max(\lambda_Q)$. It follows that the first $n_Q$ eigenvalues of $\mathscr{L} -$

$\mathbf{E}+\mathsf{diag}(\mathbf{v})$ are exactly those of $\mathscr{L}_Q$, rendering the objective of Equation (6.5) equal to zero. □

In effect, by Proposition 6.4.1, we can recover the optimal solution if $\mathbf{v}$ is appropriately normalized and $c$ is no less than a certain value. We exploit this result in Section 6.4.2 to design our algorithm by numerical optimization. We first introduce a regularization term.

**Regularization.** The objective in Equation 6.5 does not prevent $\mathbf{v}$ from taking arbitrary values. However, since $\mathscr{L}-\mathbf{E}$ has two connected components, $\mathbf{v}$ plays a role similar to that of Fiedler's vector in the minimization of the normalized cut [125]. This observation leads us to the *spectral regularization* $\mathbf{v}^\top(\mathscr{L}-\mathbf{E})\mathbf{v}$ that exhorts $\mathbf{v}$ to take values in the null-space of $\mathscr{L}-\mathbf{E}$. In other words, the spectral regularizer drives $\mathbf{v}$ to be a stepwise function. We combine the spectral regularization with our objective as follows:

$$\begin{aligned}&\min_{\mathbf{v},\mathbf{E}} \underbrace{\|\lambda(\mathscr{L}-\mathbf{E}+\mathsf{diag}(\mathbf{v}))-\lambda_Q\|_2^2}_{\text{Data term}}+\mu \underbrace{\mathbf{v}^\top(\mathscr{L}-\mathbf{E})\mathbf{v}}_{\text{Spectral regularizer}}\\ &\text{s.t. } \mathbf{E}=\mathbf{E}^\top, \mathbf{E}1=0,\ \mathsf{off}(\mathscr{L}-\mathbf{E})\leq 0,\\ &\quad \|\mathbf{v}\|=c\end{aligned} \tag{6.11}$$

where $\mu \geq 0$ is a regularization coefficient.

**Corollary.** *Chapter 6.4.1 applies also in the presence of the spectral regularization term in Equation* (6.11).

*Proof.* Let $\mathbf{E}$ be constructed from Equations (6.6)–(6.9). $\mathscr{L}-\mathbf{E}$ is the Laplacian of a graph composed of two disjoint components, one of which is exactly indicated by Equation (6.6), i.e., the query subgraph $Q$. Then $\mathbf{v}$ in Equation (6.6) belongs to the null-space of $\mathscr{L}-\mathbf{E}$, rendering the regularization term 0, hence Equation (6.6) also provides the global minimum of Equation (6.11). □

### 6.4.1 Localizing disconnected subgraphs

A special case of subgraph localization is that of a graph with a number of connected components, one of which corresponds to the query graph $Q$. In this case the editing matrix $\mathbf{E}=0$, leading to the simpler objective:

$$\begin{aligned}&\min_{\mathbf{v}} \|\lambda(\mathscr{L}+\mathsf{diag}(\mathbf{v}))-\lambda_Q\|_2^2+\mu\mathbf{v}^\top\mathscr{L}\mathbf{v}\\ &\text{s.t. } \|\mathbf{v}\|=c.\end{aligned} \tag{6.12}$$

Interestingly, Equation 6.12 also covers the case in which $\mathbf{E} = \mathbf{E}_{gt}$, i.e., the ground-truth editing Laplacian matrix.

### 6.4.2 Numerical optimization

We exploit Chapter 6.4.1 to craft a numerical procedure that minimizes the objective in Equation (6.5), collaterally optimizing for $\mathbf{E}$ and $\mathbf{v}$. In the first iteration $q = 0$, we initialize $\mathbf{E}_q = 0$. In iteration $q+1$ we minimize $f(\mathbf{v}, \mathbf{E}_q) = \lambda(\mathscr{L} - \mathbf{E}_q + \mathsf{diag}(\mathbf{v})) - \lambda_Q\|_2^2 + \mu\mathbf{v}^\top(\mathscr{L} - \mathbf{E}_q)\mathbf{v}$ for $\mathbf{v}$ given $\mathbf{E}_q$:

$$\mathbf{v}_{q+1} = \arg\min_{\mathbf{v}:\|\mathbf{v}\|=c} f(\mathbf{v}, \mathbf{E}_q), \tag{6.13}$$

via *projected gradient descent*, until convergence; an iteration $k+1$ of projected gradient descent performs the step:

$$\begin{aligned} \mathrm{x}_{k+1} &= \mathrm{x}_{k+1} - \alpha\nabla_{\mathbf{v}} f(\mathbf{v}, \mathbf{E}_q) \\ \mathbf{v}_{k+1} &= c\frac{\mathrm{x}_k}{\|\mathrm{x}_k\|}, \end{aligned} \tag{6.14}$$

where $\alpha > 0$ regulates the learning rate. The gradient $\nabla_{\mathbf{v}}$ for Equation (6.14) requires a *differentiable eigendecomposition*, which is achievable by extant methods [140].

We subsequently update $\mathbf{E}$ according to:

$$\tilde{\mathbf{v}} = \frac{\mathbf{v}_q - \min(\mathbf{v}_q)}{\max(\mathbf{v}_q) - \min(\mathbf{v}_q)}, \tag{6.15}$$

$$S_{ij} = |\tilde{v}_i - \tilde{v}_j| A_{ij}, \tag{6.16}$$

$$\mathbf{E}_{q+1} = \mathsf{diag}(\mathrm{S}1) - \mathrm{S}. \tag{6.17}$$

We obtain a threshold $\tau$ of the indicator function $\delta(\mathbf{v})$ in Equation (6.2) for the nodes comprising the subgraph by splitting the elements of $\mathbf{v}$ into two clusters minimizing sum-of-squares error from the mean (i.e., optimizing the $k$-means objective in one dimension) and compute the matrix $\mathbf{E}$ from this thresholded $\mathbf{v}$ by Equations (6.15)–(6.17).

**The SSL algorithm.** We eventually present our *Spectral Subgraph Localization* (SSL) algorithm 5 for Problem 1. SSL takes as input the adjacency matrix $\mathbf{A}$ of the full graph $G$ and the spectrum of a query subgraph, and returns the vector $\mathbf{v}$ and the threshold $\tau$ of the indicator function $\delta$; it additionally requires some hyperparameters, such as the number of outer iterations $\texttt{maxiter}_{\texttt{out}}$, the number of inner iterations $\texttt{maxiter}_{\texttt{in}}$, the learning rate $\alpha$, and the regularization coefficient $\mu$. We empirically found that the number of iterations and the learning rate do not significantly

**Algorithm 5** SSL

```
Require: A adjacency matrix of the full graph; $\lambda_Q$ spectrum of the query subgraph.
Params: $\mu$ regularization coefficient; $a_{\texttt{tol}}$ loss tolerance; $\alpha$ gradient step size;
    $\texttt{maxiter}_{\texttt{in}}$ maximum number of inner iterations; $\texttt{maxiter}_{\texttt{out}}$ maximum number
    of outer iterations
Ensure: Vector $\mathbf{v}$, threshold $\tau$
 1: $\mathscr{L} \leftarrow \mathbf{D} - \mathbf{A}$
 2: loss $\leftarrow \infty$
 3: $c \leftarrow 2\sqrt{n - n_q}\max(\lambda_Q)$
 4: $\mathbf{v}_0 \leftarrow \frac{c}{|V|}1$
 5: $\mathbf{E}_0 \leftarrow 0$
 6: while $q \leq \texttt{maxiter}_{\texttt{out}}$ and loss $\geq a_{\texttt{tol}}$ do
    // Compute $\mathbf{v}_{q+1}$ by iterating (6.14) $\texttt{maxiter}_{\texttt{in}}$
 7:     $\mathbf{v}_{q+1} \leftarrow \arg\min_{\mathbf{v}:\|\mathbf{v}\|=c} f(\mathbf{v}, \mathbf{E}_q)$
    // Update $\mathbf{E}_{q+1}$ via (6.15))-(6.17)
 8:     $\mathbf{E}_{q+1} \leftarrow$ E_from_v$(\mathbf{v}_{q+1})$
    // Update the threshold $\tau$
 9:     $\tau \leftarrow$ k_means_1d$(\mathbf{v}_{q+1})$
10:     loss $\leftarrow f(\mathbf{v}_\tau, \mathbf{E}_\tau)$
11:     $q \leftarrow q + 1$
12: return $\mathbf{v}_q, \tau$
```

| Parameter | Value/range | Description |
|---|---|---|
| $\texttt{maxiter}_{\texttt{in}}$ | 500–1000 | number of inner iterations |
| $\texttt{maxiter}_{\texttt{out}}$ | 3–5 | number of outer iterations |
| $a_{\texttt{tol}}$ | $10^{-5}$ | loss tolerance |
| $\alpha$ | 0.02 | gradient step size |

Table 6.1: SSL hyperparameters and default values.

affect results across datasets if chosen within some range; we report those ranges and recommended values in Table 6.1. On the other hand, the regularization coefficient $\mu$ in Equation (6.11) requires tuning for each dataset. We thus first normalize the value of $\mu$ by $c^2$ to remove the dependency on $\mathbf{v}$'s magnitude and then perform grid search on a range of values for $\mu$ to select an appropriate value.

The optimization process alternates the projected gradient optimization in Equation (6.14) and the update of $\mathbf{E}$ using Equations (6.15)–(6.17) until it converges or reaches the maximum number of iterations $\texttt{maxiter}_{\texttt{out}}$.

Figure 6.2 illustrates the solution's progressive convergence through iterations, while Figure 6.3 shows an example result.

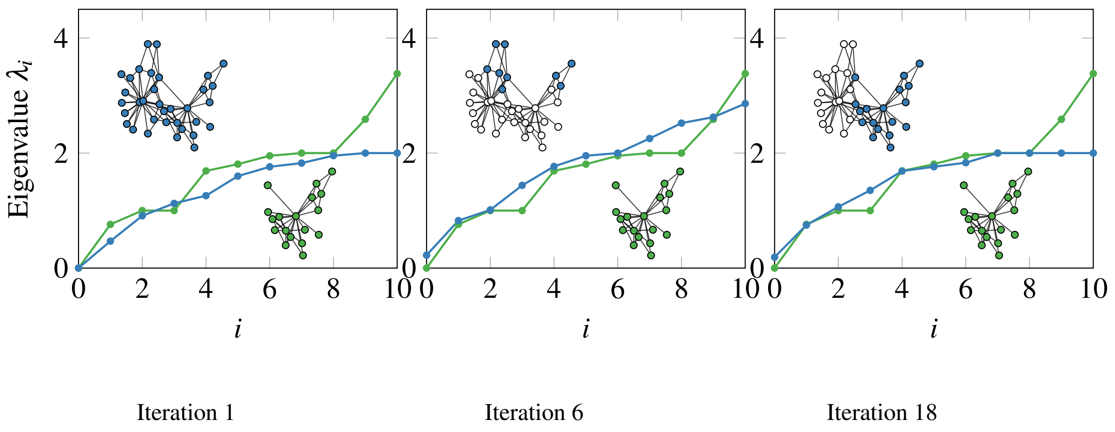


Figure 6.2: Alignment of the spectrum $\lambda_Q$ of $Q$ and the part of the spectrum $\lambda(\mathscr{L} - \mathbf{E} + \mathrm{diag}(\mathbf{v}))$ of $G$ corresponding to $Q$ at 1, 6, and 18 iterations. As two spectra progressively converge, especially in smaller eigenvalues, $Q$ is correctly localized in $G$.

### 6.4.3 Complexity Analysis

We express the worst-time complexity of the algorithm in terms of the number of nodes $n$ in the graph $G$. The eigendecomposition in Equation (6.14) takes $\mathscr{O}(n^3)$ per iteration; the computation in Equation (6.17) takes $\mathscr{O}(n^2)$ for the matrix-vector multiplication; 1-D k-means in Line 9 takes $\mathscr{O}(n)$ with the best algorithm [46]. In

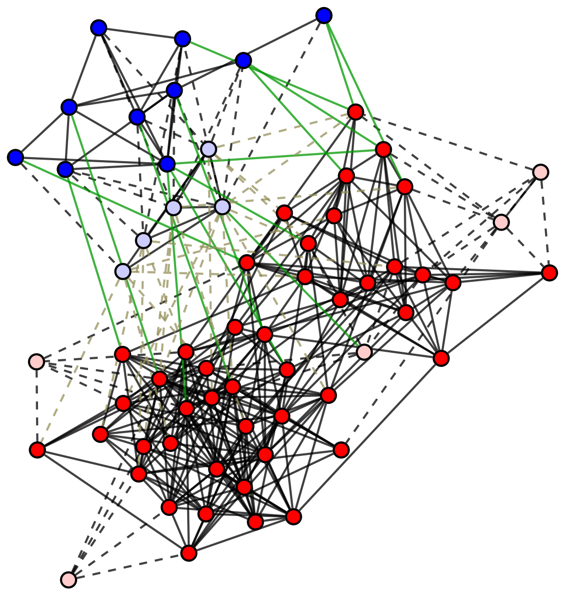


Figure 6.3: Example SSL result: blue nodes show the ground-truth subgraph $Q$; red nodes are the remaining nodes of $G$; shaded nodes are classified wrongly; solid lines show edges correctly altered (green) or unaltered (black) by $E$; dotted lines show edges incorrectly unaltered (green) or altered (black).

effect, the total time is $\mathcal{O}(\texttt{maxiter}_{\texttt{out}} \cdot (\texttt{maxiter}_{\texttt{in}} * n^3 + n^2 + n))$, where the $\mathcal{O}(n^3)$ term dominates. However, as $\mathscr{L} - \mathbf{E} + \mathsf{diag}(\mathbf{v})$ is a graph's Laplacian, its spectrum can be efficiently approximated through sampling [24].

## 6.5 Experiments

Here we empirically evaluate our method, SSL, on a number of datasets and against several hypotheses. Our evaluation aims to answer the following questions:

**(Q1)** Do the regularization term and the constraint $||\mathbf{v}|| = c$ in Equation (6.11) help the localization?

**(Q2)** How does the number of edges between the part corresponding to $Q$ and its complement affect the quality of the localization?

**(Q3)** How does SSL fare against state-of-the-art methods for graph alignment?

**(Q4)** What kind of cases are challenging for SSL?

### 6.5.1 Experiment Design

We implemented SSL in Python 3.6 and ran experiments on an 8-core Intel Core i7-8565U machine with 16GB RAM. Unless otherwise stated, we choose $\texttt{maxiter}_{\texttt{out}} = 2$, $\texttt{maxiter}_{\texttt{in}} = 500$ and $\mu = 1000$.

**Datasets.** We evaluate SSL on the two real-world graphs presented in Table 6.2. Additionally, we generate graphs following the stochastic block model (SBM) [53]. SBM graphs allow for generating graphs with community structure and control the number of inter- and intra-community edges.

| **Dataset** | $\lvert V\rvert$ | $\lvert E\rvert$ | **Network type** |
|---|---|---|---|
| Football [44] | 115 | 613 | contact |
| HighSchool [39] | 327 | 5 818 | proximity |

Table 6.2: Graphs used in our evaluation: number of nodes $|V|$, number of vertices $|E|$, and graph type.

**Choosing $Q$.** Given a number $k$, we generate a query workload of size $V_Q = k$ from a real-world graph $G$ to evaluate our subgraph localization method as follows.

1. Randomly select a node $u$, add it to $V_Q$ and place all its neighbors into a set $N$.
2. Randomly select a node $u'$ from $N$, add it to $V_Q$, place all its neighbors which are not in $V_Q$ into $N$.

3. Repeat the previous step until $|V_Q| = k$.

4. Set $Q$ as the subgraph induced by $V_Q$ in $G$.

For graphs generated by the stochastic block model, we set $Q$ as the smallest community.

**Quality measure.** We assess results by **Balanced Accuracy (BA)**; given the query graph $V_Q$ and the subgraph $V_S$ returned by a localization algorithm, balanced accuracy $\mathrm{BA}(\mathbf{v})$ is defined as the arithmetic mean of sensitivity (or recall) and specificity:

$$\mathrm{BA}(\mathbf{v}) = \frac{{}^{|V_Q \cap V_S|}/_{|V_Q|} + {}^{|\neg V_Q \cap \neg V_S|}/_{|\neg V_Q|}}{2} \tag{6.18}$$

### 6.5.2 Ablation Study

We commence our study by examining how the terms in SSL's objective function (Equation 6.11) affect the result. Recall that the objective function consists of: (1) the **data term**, that drives the alignment between the spectrum of the part and that of the query, (2) a **spectral regularization** term that exhorts $\mathbf{v}$ to be in the null space of $\mathscr{L} - \mathbf{E}$ and (3) the **sphere constraint** that enforces a constant norm on the potential $\mathbf{v}$. To study the contribution of each term on the results, we compare SSL against two variants thereof:

1. A method only optimizing the data term $||\lambda(\mathscr{L} - \mathbf{E} + \mathsf{diag}(\mathbf{v})) - \lambda_Q||_2^2$.

2. A method optimizing a linear combination of the data term $||\lambda(\mathscr{L} - \mathbf{E} + \mathsf{diag}(\mathbf{v})) - \lambda_Q||_2^2$ and the spectral regularization $\mathbf{v}^\top(\mathscr{L} - \mathbf{E})\mathbf{v}$, without a sphere constraint.

We experiment on graphs with $|V| = 200$ nodes sampled form the stochastic block model, letting the size of the query subgraph increase from 20% of the graph to 45%. Figure 6.4 reports on the results of this ablation study in terms of average balanced accuracy over 5 sampled graphs for each subgraph size. Unsurprisingly, the optimization of the data term yields the worst results, although the method performs well on small subgraphs. Still, the addition of the spectral regularizer and sphere constraint enhances the results up to 20% accuracy. For small subgraphs the sphere constraint brings only marginal gains compared to the spectral regularization. On the other hand, on large query subgraphs, the sphere constraint boosts the accuracy by an additional 8%.

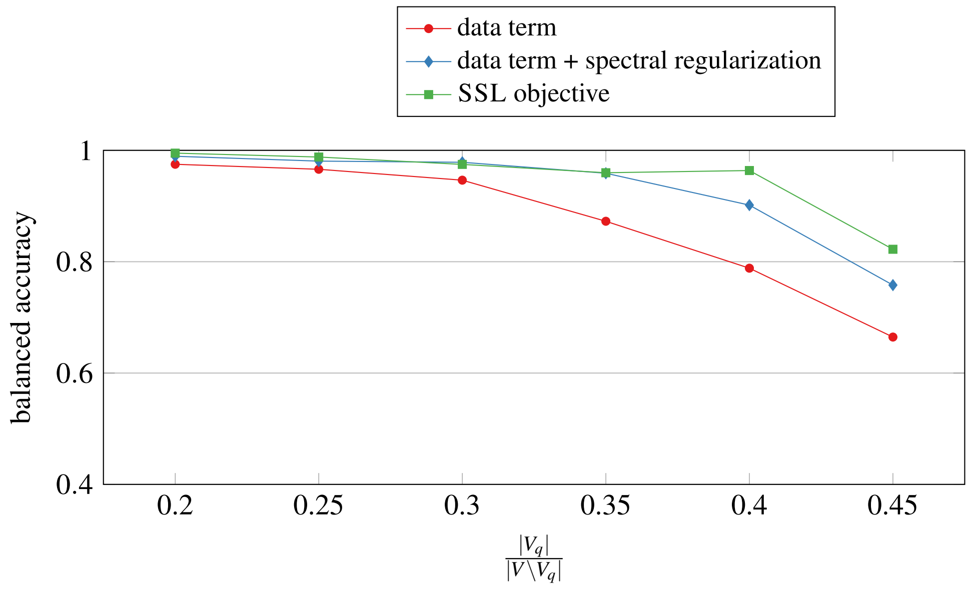


Figure 6.4: Balanced Accuracy for variants of the objective function in Equation 6.11 with: (1) data term only; (2) data term and spectral regularizer; and (3) data term, spectral regularizer. and sphere constraint (SSL objective).

To further corroborate these results, Figure 6.5 shows an example of how the terms impact the potential **v**, on a 40-node graph sampled from the SBM with two communities with 20 nodes each; the query graph is one of the two communities. Ideally, we would like to obtain a **v** clearly separating values between the part corresponding to the query graph and the rest. In that case, we say that **v** forms a step function. The optimization of the *data term* (left chart) alone leads to no clear separation between the two parts. Introducing the *spectral regularization* (middle chart) yields a result closer to a step function, though some nodes are incorrectly assigned to the part. Finally, the full objective in Equation 6.11 produces to a clearly separated potential vector **v**. Visualizing the mapping of this potential to the graph $G$, we clearly recognize the part $G_S$ as the light-colored nodes.

### 6.5.3 Competing methods

Here we assess our method against previous work. However, to the best of our knowledge, no extant unsupervised method is capable to answer localization queries in graphs with more than 15 nodes [117]. Therefore, we compare SSL to the nearest feasible competitor, namely the state-of-the-art method for unsupervised graph alignment, CONE [19]. To set up CONE so that it detects subgraphs, we inject in the query nodes with degree 0, so that the size of the query $Q$ corresponds to that of the graph $G$,

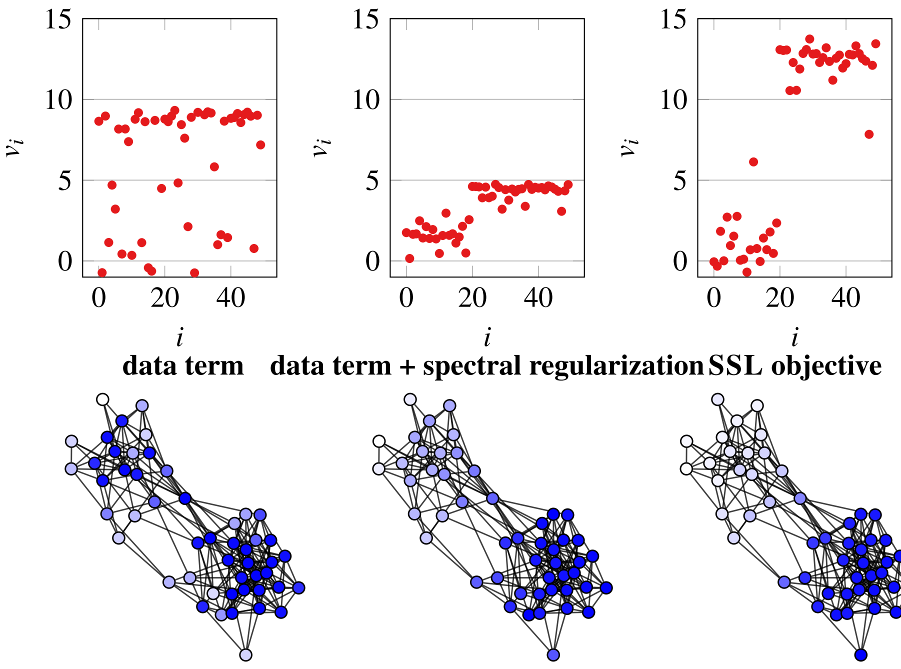


Figure 6.5: Plots of $\mathbf{v}$ elements sorted in ascending order, and $\mathbf{v}$ mapped to $G$ for different designs of the objective function: (1) only with data term; (2) with data term and spectral regularizer; and (3) with data term, spectral regularizer, and sphere constraint; nodes in $|V_S|$ correspond to the 20 smallest entries in $\mathbf{v}$, lighter shades indicating smaller values; applying only the data term, the method fails to localize $G_S$; applying regularization renders the separation clearer; the addition of the sphere constraint induces a perfectly clear separation.

i.e., $|V_Q = V|$. We extract the ensuing localization vector as the matches of query nodes in $G$.

In the same experiment we also investigate the effect of the number of edges connecting the query graph to the rest of the parent graph. This experimental regime be seen as a supervision for the edge correction matrix $\mathbf{E}$, since deleting a connecting edge implies setting the corresponding default entry in $\mathbf{E}$ to ground truth. We extract query subgraphs of 20%, 30% and 40% of the size of the full parent graph. In each case, we start with a parent graph where the part $G_S$ is fully detached from the rest and iteratively add edges until we reach the original parent graph.

Figure 6.6 presents our results on the Football (top) and HighSchool (bottom) data. Remarkably, SSL effectively localizes the subgraph in both datasets, outperforming CONE by up to 50% balanced accuracy. Given 50% of the original edges, SSL yields balanced accuracy of 0.8. Notably, with **highschool** dataset, the problem is more challenging; SSL effectively localizes a query graph as large as 20% of the full graph, yet fares closer to CONE as the size of the query graph grows. This outcome potentially arises from the graph's topology and the presence of *dangling* nodes (i.e.,

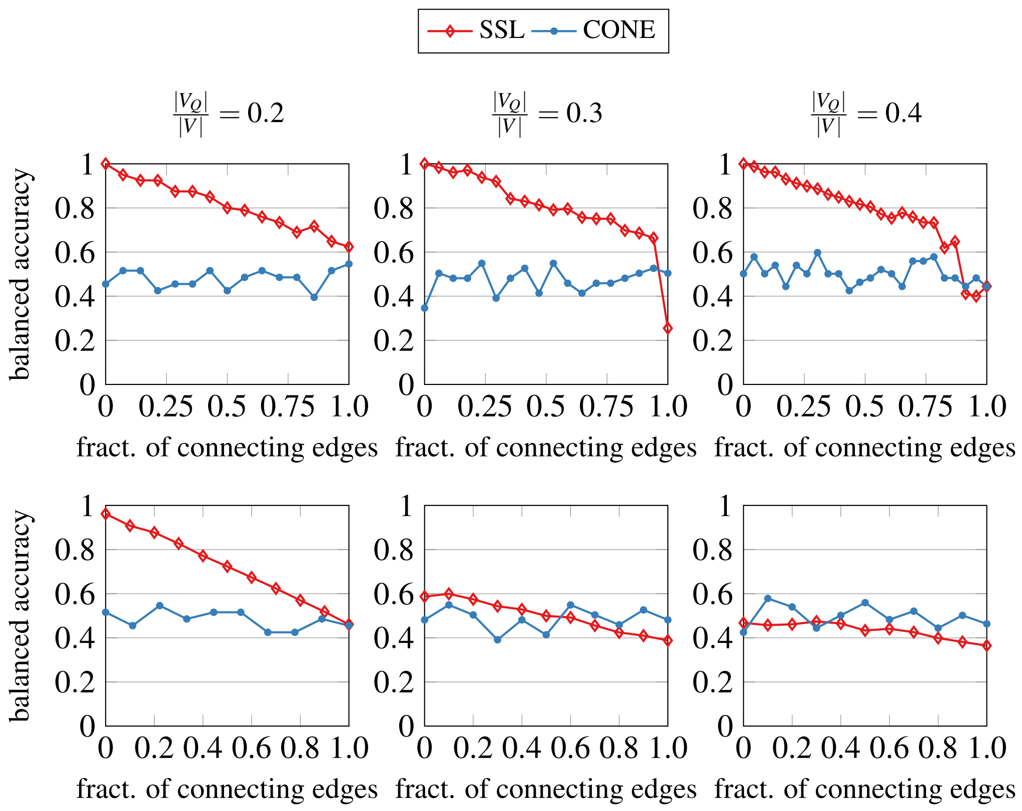


Figure 6.6: Results on localizing query graphs of 20%, 30%, 40% of the **football** (upper row) and **highschool** (lower row) graphs vs. fraction of connecting edges between the query graph and rest of the parent graph; we start with a disconnected query graph and add connecting edges, until the original parent graph is reconstructed.

nodes with degree 1). We investigate this matter in the following.

### 6.5.4 Challenging Cases

In this section, we investigate some examples of challenging cases for SSL, illustrated in Figures 6.7 and 6.8. In the results of Figure 6.7, we observe that the spectrum of the query graph and that of the detected subgraph are well aligned. However, the localized subgraph deviates substantially from the ground truth. Similarly, in the results of Figure 6.8, while SSL does not perfectly align the two spectra, it yields a correlated spectrum. Nevertheless, SSL detects a subgraph comprising nodes that are only connected by one edge. In both cases, the challenge arises from the sensitivity of the spectrum at weakly connected parts of the graph. Changing the Laplacian in such parts by adding **v** has a larger impact on the spectrum than changing the Laplacian in a well-connected neighborhood. These types of graphs force the optimization process

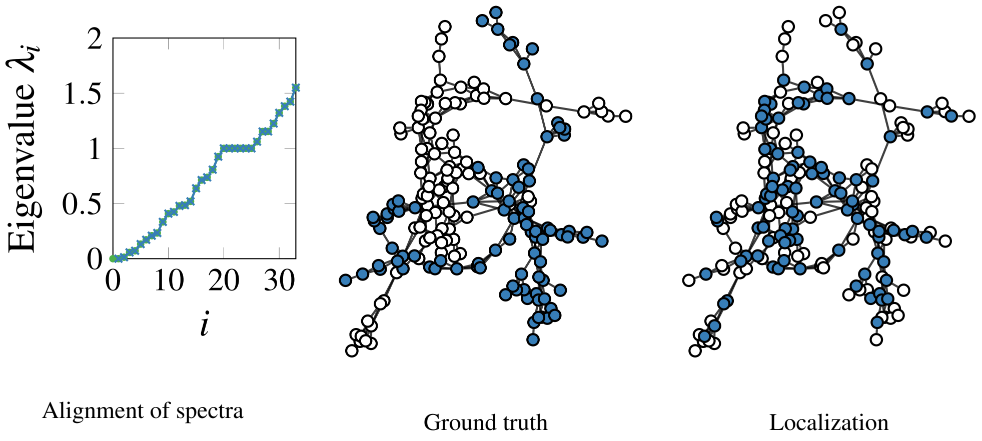


Figure 6.7: Alignment of the spectrum $\lambda_Q$ of $Q$ and the corresponding part of the spectrum $\lambda(\mathscr{L} - \mathbf{E} + \text{diag}(\mathbf{v}))$ of $G$ after convergence, ground truth $V_S$ (blue) and $V \backslash V_S$ (white), and corresponding localization by SSL; while the spectra are perfectly aligned, the detected subgraph is not the ground truth. The depicted graph is a protein-protein interaction network from the D&D dataset [31].

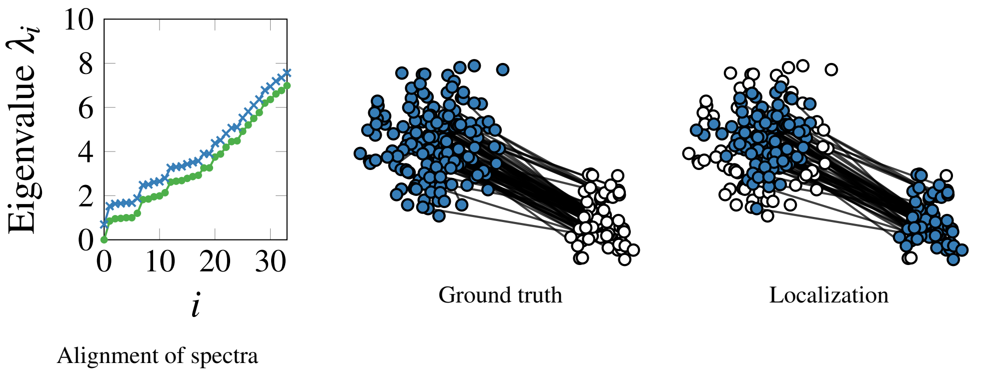


Figure 6.8: Alignment of the spectrum $\lambda_Q$ of $Q$ and the corresponding part of the spectrum $\lambda(\mathscr{L} - \mathbf{E} + \text{diag}(\mathbf{v}))$ of $G$ after convergence, ground truth $V_S$ (blue) and $V \backslash V_S$ (white), and corresponding localization by SSL; while the spectra are well aligned, the detected subgraph is not the ground truth; the detected subgraph coincides with nodes that have degree 0 or 1. The depicted graph is **arenas** [73].

into a local optimum, as the optimizer has a large incentive to separate these weakly connected parts of the graph.

## 6.6 Conclusion

We studied the challenging data engineering problem of subgraph localization, which calls to find a set of nodes in a larger graph that best corresponds to a given query subgraph. We devised a novel spectral solution that identifies the query match by adding a penalty to the Laplacian matrix so as to obtain a spectrum similar to that of the query graph. This novel approach requires solving a non-convex, non-smooth problem for which we devised a numerical method. Our results demonstrate that our spectral method localizes query subgraphs more effectively than a baseline based on the state-of-the-art method for graph alignment. To our knowledge, this is the first endeavor in effective subgraph localization that can handle graphs of any size in the order of magnitude of hundreds of nodes.

Chapter 7

# Conclusion

In this section, we summarize the outcomes of this dissertation and discuss potential avenues for future work.

## 7.1 Outcome

This dissertation deals with methods for graph correspondences for unattributed graphs. We introduced three contributions: An algorithm for graph alignment [51] in Chapter 4, an evaluation of unrestricted graph alignment algorithms [127] in Chapter 5 and an algorithm for subgraph localization [52] in Chapter 6.

**Spectral Graph Alignment**. In Chapter 4, we investigated the applicability of spectral graph theory to the task of full graph alignment. We developed a powerful functional maps based approach [51], which establishes a correspondence between eigenvectors of both graphs. At the same time, we identified a modular framework which many state-of-the-art graph alignment algorithms follow. We introduced different add-ons for our method, which can be also applied to other modular graph alignment algorithms. Our developed methods GRASP and B-GRASP are scalable methods which yield a higher alignment quality than other scalable graph alignment algorithms.

**Evaluation of unrestricted graph alignment algorithms**. In the previous work on graph alignment, we noticed a lack of comparative evaluation methodologies for unrestricted graph alignment. Unrestricted graph alignment methods emerge from a lot of different fields, and they have never been all evaluated against each other. In the work presented in Chapter 5, we performed the first thorough evaluation of unrestricted algorithms against each other in a unified evaluation framework and on a clear benchmark with both real and synthetic data [127].

**Spectral subgraph localization**. In the last project, presented in Chapter 6, we developed SSL, a spectral approach for subgraph localization [52]. In order to establish a correspondence between a query graph $Q$ and a graph $G$, we aligned the spectra of these two graphs. Our approach is the first graph localization approach able scale to graphs of hundreds of nodes. It outperforms the baseline which is based on a graph alignment approach.

These contributions structure the field of graph alignment and demonstrate the conceptual usefulness of spectral graph theory for graph correspondence tasks where only the graph is available. The work conducted opens a set of interesting possibilities for future work, which we discuss in the next section.

## 7.2 Future Work

Several interesting directions for future work emerge from the work carried out during the course of this dissertation.

**Additional information.** Throughout this thesis we worked with unattributed graphs. Our work was guided by the question: How can we solve graph correspondence problems only given its structure? This made us independent of noisy or missing node attributes and the requirement for supervision by ground truth data. But undoubtedly, additional information can and has helped in solving graph correspondence problems [60, 155]. While currently our methods GRASP [51] and SSL [52] do not require it, they can be adapted.

We will present ideas to extend GRASP and SSL to using *node attributes* and *supervision*. A central element of GRASP is the computation of the functional map. The computation of a high-quality functional map requires a thoughtful choice of corresponding functions. In GRASP, we use the diagonal of the heat kernel at different time steps as corresponding functions rooted in the assumption that graphs will be structurally similar. If we are given continuous node attributes and can assume that they are similar across graphs, we can directly transfer node attributes into a corresponding function. The same holds true for known ground truth alignments, so called *seed nodes*. As for a pair of seed nodes, if we are sure that they correspond across graphs, we can construct corresponding functions from them. Ground truth nodes can also guide the localization process of SSL. In this case, ground truth knowledge would mean that we do know whether a node in $G$ belongs to the localized part or not. This can directly be introduced as a fixed element in the optimization process for the potential vector $v$, which at the positions of ground truth nodes can be initialized with the intended value.

**Scalability.** The major bottleneck in applying spectral methods to graph mining is the eigendecomposition of the Laplacian, which is computationally expensive. Thus spectral methods do hardly scale to graphs with millions or billions of nodes. However, as a lot of real world graphs are of these scales, and as spectral methods have been proven to successfully work on smaller graphs, scaling these methods up is an obvious next step.

We have identified four starting points for this.

1. *Decreasing the number of eigenvalues and eigenvectors.* In SSL and GRASP only a fixed number of eigenvalues and/or eigenvectors are required. We thus can go to methods that estimate only the first $k$ eigenvalues and eigenvectors. Power iteration methods [14] provide a promising starting point here.

2. *Approximating the spectrum.* In SSL only the spectrum is required. A promising approach for approximating the spectrum of a graph was presented in [24], which could be applied for SSL.

3. *Tracking changes in the Laplacian.* In SSL, the Laplacian is edited in every iteration, which requires an expensive recomputation of the spectrum. Tracking the amount of change possible in one iteration and by that, bounding the change that can happen to the spectrum in that step is an appealing idea. Bounding possible changes in Laplacians has been investigated in the area of GNNs [160].

4. *Replacing the Laplacian.* In both GRASP and SSL, eigenvectors and eigenvalues of the Laplacian can be interpreted as graph descriptors and could be replaced with other, less computationally expensive descriptors based on GNNs [34]. In GRASP, they occur in the heat kernel, a graph descriptor from which corresponding functions are computed and in the final node embeddings. In SSL, we could move to aligning other graph descriptors than the spectrum.

**Other graph correspondence problems.** In this thesis, we have handled full graph alignment and subgraph localization. However, tackling related problems with similar approaches is a natural next step. GRASP currently requires graphs of the same size. It could be extended to partial matching. Different scenarios are possible here: a smaller graph has to be aligned to a larger graph or two graphs share an overlap which has to be identified and aligned. This problems show a component of subgraph localization, so a combination with SSL seems possible.

Also subgraph localization can be extended. We have worked with cases where we aim at detecting an isomorphic subgraph. However, in real world applications we also have to consider changes in structure, as we might aim at not exact subgraph isomorphism, but more at detecting a structurally similar community. Also tackling more complex subgraph localization problems is conceivable, for example detecting multiple subgraphs at once, or detecting multiple potential occurrences of one subgraph as a ranking problem.

To conclude, this dissertation has made fundational contributions to the field of graph correspondences, advancing the state-of-the-art in graph alignment and subgraph localization. We have shown that spectral methods can be applied to solve complicated real-world problems in an elegant, mathematically sound way. Last but not least, several exciting possibilities for future work emerge from this dissertation.

# Bibliography

[1] Mohammad Abdulkader Abdulrahim. *Parallel Algorithms for Labeled Graph Matching*. PhD thesis, Colorado School of Mines, USA, 1998. 28

[2] P-A Absil, Christopher G Baker, and Kyle A Gallivan. Trust-region methods on riemannian manifolds. *FoCM*, 7(3):303–330, 2007. 36

[3] Charu C Aggarwal, Haixun Wang, et al. *Managing and mining graph data*, volume 40. 2010. 3

[4] Geir Agnarsson and Raymond Greenlaw. *Graph theory: Modeling, applications, and algorithms*. 2006. 10, 16

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

[6] David A Bader, Henning Meyerhenke, Peter Sanders, and Dorothea Wagner. Graph partitioning and graph clustering. In *10th DIMACS Implementation Challenge Workshop*, 2012. 68

[7] Albert-László Barabási and Réka Albert. Emergence of scaling in random networks. *science*, 286(5439):509–512, 1999. 67

[8] Mohsen Bayati, David F. Gleich, Amin Saberi, and Ying Wang. Message-passing algorithms for sparse network alignment. *TKDD*, 7(1), 2013. ISSN 1556-4681. 22, 56, 59, 66, 67, 70, 71

[9] Mikhail Belkin and Partha Niyogi. Convergence of laplacian eigenmaps. In *Annual Conference on Neural Information Processing Systems (NeurIPS)*, pages 129–136, 2006. 33, 34

[10] M. Berger. *A Panoramic View of Riemannian Geometry*. 2007. ISBN 9783540653172. 65

[11] Nicole Berline, Ezra Getzler, and Michele Vergne. *Heat kernels and Dirac operators*. 2003. 58

[12] Paul J Besl and Neil D McKay. Method for registration of 3-d shapes. In *Sensor fusion IV: control paradigms and data structures*, volume 1611, pages 586–606. International Society for Optics and Photonics, 1992. 41, 43

[13] Monica Bianchini, Giovanna Maria Dimitri, Marco Maggini, and Franco Scarselli. Deep neural networks for structured data. In *Computational Intelligence for Pattern Recognition*, pages 29–51. 2018. 23, 89

[14] Thomas E Booth. Power iteration method for the several largest eigenvalues and eigenfunctions. *Nuclear science and engineering*, 154(1):48–62, 2006. 109

[15] Claude Brezinski and Michela Redivo-Zaglia. The pagerank vector: Properties, computation, approximation, and acceleration. *SIAM J. Matrix Anal. Appl.*, 28 (2):551–575, June 2006. ISSN 0895-4798. 61

[16] Sergey Brin and Lawrence Page. The anatomy of a large-scale hypertextual web search engine. *Computer networks and ISDN systems*, 30(1-7):107–117, 1998. 59

[17] Utkan Onur Candogan and Venkat Chandrasekaran. Finding planted subgraphs with few eigenvalues using the schur–horn relaxation. *SIAM Journal on Optimization*, 28(1):735–759, 2018. 24

[18] Xiyuan Chen, Mark Heimann, Fatemeh Vahedian, and Danai Koutra. CONE-align: Consistent network alignment with proximity-preserving node embedding. In *CIKM*, page 1985–1988, 2020. 59, 64, 66, 68

[19] Xiyuan Chen, Mark Heimann, Fatemeh Vahedian, and Danai Koutra. Cone-align: Consistent network alignment with proximity-preserving node embedding. In *Proceedings of the 29th ACM International Conference on Information & Knowledge Management*, pages 1985–1988, 2020. 6, 20, 21, 29, 30, 39, 44, 45, 48, 101

[20] Yoni Choukroun, Alon Shtern, Alex Bronstein, and Ron Kimmel. Hamiltonian operator for spectral shape analysis. *IEEE Trans. Vis. Comput. Graph (TVCG)*, 26(2):1320–1331, 2018. 92

[21] Moody Chu and Gene Golub. *Inverse eigenvalue problems: theory, algorithms, and applications*. 2005. 90, 92

[22] Xiaokai Chu, Xinxin Fan, Di Yao, Zhihua Zhu, Jianhui Huang, and Jingping Bi. Cross-network embedding for multi-network alignment. In *TheWebConf*, pages 273–284, 2019. 20, 28

[23] Fan RK Chung and Fan Chung Graham. *Spectral graph theory*, volume 92. 1997. 4, 12, 13, 22, 29

[24] David Cohen-Steiner, Weihao Kong, Christian Sohler, and Gregory Valiant. Approximating the spectrum of a graph. In *International Conference on Knowledge Discovery and Data Mining (KDD)*, pages 1263–1271, 2018. 22, 94, 99, 109

[25] Diane J Cook and Lawrence B Holder. *Mining graph data*. 2006. 3, 4, 10

[26] Joseph Crawford, Yihan Sun, and Tijana Milenković. Fair evaluation of global network aligners. *Algorithms for Molecular Biology*, 10(1):1–17, 2015. 57

[27] Mahashweta Das and Gautam Das. Structured analytics in social media. *PVLDB*, 8(12):2046–2047, 2015. 56

[28] Stephen Davis, Babak Abbasi, Shrupa Shah, Sandra Telfer, and Mike Begon. Spatial analyses of wildlife contact networks. *Journal of the Royal Society Interface*, 2015. 43, 68

[29] Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. Convolutional neural networks on graphs with fast localized spectral filtering. In *Annual Conference on Neural Information Processing Systems (NeurIPS)*, pages 3844–3852, 2016. 4

[30] Tyler Derr, Hamid Karimi, Xiaorui Liu, Jiejun Xu, and Jiliang Tang. Deep adversarial network alignment. In *CIKM*, pages 352–361, 2021. 19, 20

[31] Paul D Dobson and Andrew J nodoig. Distinguishing enzyme structures from non-enzymes without alignments. *Journal of molecular biology*, 330(4):771–783, 2003. ix, 104

[32] Katerina Doka, Mingqiang Xue, Dimitrios Tsoumakos, and Panagiotis Karras. k-anonymization by freeform generalization. In *AsiaCCS*, pages 519–530, 2015. 42, 47, 60, 61, 70

[33] J. Duch and A. Arenas. Community identification using extremal optimization. *Phys Rev. E*, 72:027104, 2005. 68

[34] Chi Thang Duong, Trung Dung Hoang, Hongzhi Yin, Matthias Weidlich, Quoc Viet Hung Nguyen, and Karl Aberer. Efficient streaming subgraph isomorphism with graph neural networks. *Proceedings of the VLDB Endowment*, 14(5):730–742, 2021. 6, 23, 109

[35] P. Erdös and A. Rényi. On random graphs i. *Publicationes Mathematicae Debrecen*, 6:290, 1959. 67

[36] Zhou Fan, Cheng Mao, Yihong Wu, and Jiaming Xu. Spectral graph matching and regularized quadratic relaxations: Algorithm and theory. In *International Conference on Machine Learning (ICML)*, pages 2985–2995, 2020. 20, 44, 48

[37] Soheil Feizi, Gerald Quon, Mariana Recamonde-Mendoza, Muriel Medard, Manolis Kellis, and Ali Jadbabaie. Spectral alignment of graphs. *IEEE Trans. Netw. Sci. Eng.*, 7(3):1182–1197, 2019. 21, 28, 29, 61

[38] Santo Fortunato. Community detection in graphs. *Physics reports*, 486(3-5): 75–174, 2010. 3

[39] Julie Fournet and Alain Barrat. Contact patterns among high school students. *PloS one*, 2014. 43, 68, 99

[40] Marc Fyrbiak, Sebastian Wallat, Sascha Reinhard, Nicolai Bissantz, and Christof Paar. Graph similarity and its applications to hardware security. *IEEE Transactions on Computers*, 69(4):505–519, 2019. 90

[41] Sylvestre Gallot, Dominique Hulin, and Jacques Lafontaine. *Riemannian geometry*. 1990. 38

[42] Ji Gao, Xiao Huang, and Jundong Li. Unsupervised graph alignment with wasserstein distance discriminator. In *International Conference on Knowledge Discovery and Data Mining (KDD)*, pages 426–435, 2021. 21

[43] M. Gerritsen, M. Bayati, D. F. Gleich, A. Saberi, and Y. Wang. Algorithms for large, sparse network alignment problems. In *IEEE ICDM*, pages 705–710, 2009. 70

[44] Michelle Girvan and Mark EJ Newman. Community structure in social and biological networks. *Proceedings of the National Academy of Sciences*, 99(12): 7821–7826, 2002. 99

[45] Klaus Greff, Aaron Klein, Martin Chovanec, Frank Hutter, and Jürgen Schmidhuber. The Sacred Infrastructure for Computational Research. In *Proceedings of the 16th Python in Science Conference*, pages 49–56, 2017. 66

[46] Allan Grønlund, Kasper Green Larsen, Alexander Mathiasen, Jesper Sindahl Nielsen, Stefan Schneider, and Mingzhou Song. Fast exact $k$-means, $k$-medians and Bregman divergence clustering in 1d. *arXiv preprint arXiv:1701.07204*, 2017. 98

[47] William L Hamilton. Graph representation learning. *Synthesis Lectures on Artifical Intelligence and Machine Learning*, 14(3):1–159, 2020. 39

[48] Mark Heimann, Haoming Shen, Tara Safavi, and Danai Koutra. Regal: Representation learning-based graph alignment. In *CIKM*, pages 117–126, 2018. vii, x, 6, 14, 20, 21, 28, 29, 39, 43, 48, 59, 62, 66

[49] Mark Heimann, Xiyuan Chen, Fatemeh Vahedian, and Danai Koutra. Refining network alignment to improve matched neighborhood consistency. In *Proceedings of the 2021 SIAM International Conference on Data Mining, SDM 2021, Virtual Event, April 29 - May 1, 2021*, pages 172–180, 2021. 19

[50] Judith Hermanns, Anton Tsitsulin, Marina Munkhoeva, Alexander Bronstein, Davide Mottin, and Panagiotis Karras. GRASP: Graph alignment through spectral signatures. In *APWeb-WAIM*, 2021. 5, 6, 27, 30, 59, 65, 66, 67, 70

[51] Judith Hermanns, Konstantinos Skitas, Anton Tsitsulin, Marina Munkhoeva, Alexander Frederiksen Kyster, Simon Daugaard Nielsen, Davide Mottin, Alexander Bronstein, and Panagiotis Karras. Grasp: Scalable graph alignment by spectral corresponding functions. *ACM Trans. Knowl. Discov. Data*, 2022, *under review*. 6, 13, 14, 15, 19, 27, 107, 108

[52] Judith Hermanns, Amit Boyarski, Davide Mottin, Alexander Bronstein, and Panagiotis Karras. Spectral subgraph localization. In *International Conference on Data Engineering, (ICDE)*, 2023, *under review*. 6, 12, 19, 89, 107, 108

[53] Paul W Holland, Kathryn Blackmond Laskey, and Samuel Leinhardt. Stochastic blockmodels: First steps. *Social networks*, 5(2):109–137, 1983. 99

[54] Petter Holme and Beom Jun Kim. Growing scale-free networks with tunable clustering. *Physical review E*, 65(2):026107, 2002. 67

[55] Nan Hu, Raif M. Rustamov, and Leonidas J. Guibas. Stable and informative spectral signatures for graph matching. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2313–2320, 2014. 5

[56] Chuntao Jiang, Frans Coenen, and Michele Zito. A survey of frequent subgraph mining algorithms. *The Knowledge Engineering Review*, 28(1):75–105, 2013. 3

[57] Roy Jonker and Anton Volgenant. A shortest augmenting path algorithm for dense and sparse linear assignment problems. *Computing*, 38(4):325–340, 1987. 35, 40, 65

[58] Paris A. Karakasis, Aritra Konar, and Nicholas D. Sidiropoulos. Joint graph embedding and alignment with spectral pivot. In *International Conference on Knowledge Discovery and Data Mining (KDD)*, pages 851—859, 2021. 19, 21

[59] Foteini Katsarou, Nikos Ntarmos, and Peter Triantafillou. Performance and scalability of indexed subgraph query processing methods. *Proc. VLDB Endow.*, 8(12):1566–1577, 2015. 6, 23, 89

[60] Ehsan Kazemi, S Hamed Hassani, and Matthias Grossglauser. Growing a graph matching from a handful of seeds. *PVLDB*, 8(10):1010–1021, 2015. 5, 15, 19, 28, 56, 108

[61] Brian P Kelley, Bingbing Yuan, Fran Lewitter, Roded Sharan, Brent R Stockwell, and Trey Ideker. Pathblast: a tool for alignment of protein interaction networks. *Nucleic acids research*, 32(suppl_2):W83–W88, 2004. 59

[62] Gunnar W Klau. A new graph-based method for pairwise global network alignment. *BMC bioinformatics*, 10(1):1–9, 2009. 5, 21, 28, 29, 59, 66, 67

[63] Jon M. Kleinberg. Authoritative sources in a hyperlinked environment. *J. ACM*, 46(5):604–632, September 1999. ISSN 0004-5411. 60

[64] David Knossow, Avinash Sharma, Diana Mateus, and Radu Horaud. Inexact matching of large and sparse graphs using laplacian eigenvectors. In *GdR Int. workshop.*, pages 144–153, 2009. 20, 22

[65] Johannes Kobler, Uwe Schöning, and Jacobo Torán. *The graph isomorphism problem: its structural complexity*. 2012. 4, 28

[66] Giorgos Kollias, Shahin Mohammadi, and Ananth Grama. Network similarity decomposition (nsd): A fast and scalable approach to network alignment. *TKDE*, 24(12):2232–2243, 2011. 6, 59, 60, 66

[67] Giorgos Kollias, Madan Sathe, Shahin Mohammadi, and Ananth Grama. A fast approach to global alignment of protein-protein interaction networks. *BMC research notes*, 6(1):35, 2013. 4, 21

[68] Ioannis Koutis, Alex Levin, and Richard Peng. Faster spectral sparsification and numerical algorithms for sdd matrices. *TALG*, 12(2):1–16, 2015. 38

[69] Danai Koutra, Hanghang Tong, and David Lubensky. Big-align: Fast bipartite graph alignment. In *ICDM*, pages 389–398, 2013. 56, 66

[70] Danai Koutra, Hanghang Tong, and David Lubensky. BIG-ALIGN: fast bipartite graph alignment. In *ICDM*, pages 389–398, 2013. 20, 28, 29, 43

[71] Artiom Kovnatsky, Michael M Bronstein, Alexander M Bronstein, Klaus Glashoff, and Ron Kimmel. Coupled quasi-harmonic bases. In *Comput Graph Forum*, volume 32, pages 439–448, 2013. 13, 22, 34, 35

[72] Oleksii Kuchaiev, Tijana Milenković, Vesna Memišević, Wayne Hayes, and Nataša Pržulj. Topological network alignment uncovers biological function and phylogeny. *Journal of the Royal Society Interface*, 7(50):1341–1354, 2010. 6, 59, 60, 66

[73] Jérôme Kunegis. Konect: the Koblenz network collection. In *WWW*, pages 1343–1350. ACM, 2013. ix, 43, 68, 104

[74] Michihiro Kuramochi and George Karypis. Frequent subgraph discovery. In *ICDM*, pages 313–320. IEEE, 2001. 6, 23, 89

[75] Michihiro Kuramochi and George Karypis. Grew-a scalable frequent subgraph discovery algorithm. In *ICDM*, pages 439–442. IEEE, 2004. 23

[76] Alexander Frederiksen Kyster, Simon Daugaard Nielsen, Judith Hermanns, Davide Mottin, and Panagiotis Karras. Boosting graph alignment algorithms. In *CIKM*, 2021. 6, 27, 30, 39

[77] Victor E Lee, Ning Ruan, Ruoming Jin, and Charu Aggarwal. A survey of algorithms for dense subgraph discovery. In *Managing and Mining Graph Data*, pages 303–336. 2010. 23

[78] Jure Leskovec and Andrej Krevl. SNAP Datasets: Stanford large network dataset collection. urlhttp://snap.stanford.edu/data, June 2014. 43, 68

[79] Jure Leskovec, Jon Kleinberg, and Christos Faloutsos. Graph evolution: Densification and shrinking diameters. *ACM Trans. Knowl. Discov. Data*, 1(1):2–es, March 2007. ISSN 1556-4681. 68

[80] Yujia Li, Chenjie Gu, Thomas Dullien, Oriol Vinyals, and Pushmeet Kohli. Graph matching networks for learning the similarity of graph structured objects. In *ICML*, pages 3835–3845. PMLR, 2019. 23

[81] 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. 5, 21, 28, 29, 56, 60

[82] Or Litany, Emanuele Rodolà, Alexander M. Bronstein, and Michael M. Bronstein. Fully spectral partial shape matching. *Comput. Graph. Forum*, 36(2): 247–258, 2017. 13, 22

[83] Li Liu, William K Cheung, Xin Li, and Lejian Liao. Aligning users across social networks using network embedding. In *IJCAI*, pages 1774–1780, 2016. 4, 20, 28

[84] Zhaoyu Lou, Jiaxuan You, Chengtao Wen, Arquimedes Canedo, Jure Leskovec, et al. Neural subgraph matching. *arXiv preprint arXiv:2007.03092*, 2020. 23

[85] Ilya Makarov, Dmitrii Kiselev, Nikita Nikitinsky, and Lovro Subelj. Survey on graph embeddings and their applications to machine learning problems on graphs. *PeerJ Computer Science*, 7, 2021. 58

[86] Eric Malmi, Aristides Gionis, and Evimaria Terzi. Active network alignment: a matching-based approach. In *CIKM*, pages 1687–1696, 2017. 20

[87] Tong Man, Huawei Shen, Shenghua Liu, Xiaolong Jin, and Xueqi Cheng. Predict anchor links across social networks via an embedding approach. In *IJCAI*, 2016. 20

[88] Hermina Petric Maretic, Mireille El Gheche, Giovanni Chierchia, Matthias Minder, and Pascal Frossard. Wasserstein-based graph alignment. *IEEE Transactions on Signal and Information Processing over Networks*, 2022. 66

[89] Brendan D McKay et al. Practical graph isomorphism. 1981. 4

[90] Mark Meyer, Mathieu Desbrun, Peter Schröder, and Alan H Barr. Discrete differential-geometry operators for triangulated 2-manifolds. In *Visualization and mathematics III*, pages 35–57. 2003. 13

[91] Marianna Milano, Pietro Hiram Guzzi, Olga Tymofieva, Duan Xu, Christofer Hess, Pierangelo Veltri, and Mario Cannataro. An extensive assessment of network alignment algorithms for comparison of brain connectomes. *BMC bioinformatics*, 18(6):31–45, 2017. 57

[92] Shahin Mohammadi, David F Gleich, Tamara G Kolda, and Ananth Grama. Triangular alignment (tame): A tensor-based approach for higher-order network alignment. *TCBB*, 14(6):1446–1458, 2016. 66

[93] James Moody. The structure of a social science collaboration network: Disciplinary cohesion from 1963 to 1999. *American sociological review*, 69(2): 213–238, 2004. 3

[94] Rafael Najmanovich, Natalja Kurbatova, and Janet Thornton. Detection of 3d atomic similarities and their use in the discrimination of small molecule protein-binding sites. *Bioinformatics*, 24(16):i105–i111, 2008. 90

[95] Huda Nassar, Nate Veldt, Shahin Mohammadi, Ananth Grama, and David F. Gleich. Low rank spectral network alignment. In *TheWebConf*, 2018. 6, 20, 21, 28, 29, 40, 43, 48, 59, 61, 66, 67, 68, 70

[96] Mark E. J. Newman. The structure and function of complex networks. *SIAM Rev.*, 45(2):167–256, 2003. 50, 84

[97] Mark EJ Newman. Power laws, pareto distributions and zipf's law. *Contemporary physics*, 46(5):323–351, 2005. 52

[98] Mark EJ Newman. Finding community structure in networks using the eigenvectors of matrices. *Physical review E*, 74(3):036104, 2006. 68

[99] Mark EJ Newman and Duncan J Watts. Renormalization group analysis of the small-world network model. *Physics Letters A*, 263(4-6):341–346, 1999. 67

[100] Behnam Neyshabur, Ahmadreza Khadem, Somaye Hashemifar, and Seyed Shahriar Arab. NETAL: a new graph-based method for global alignment of protein-protein interaction networks. *Bioinformatics*, 29(13):1654–1662, 05 2013. 65

[101] Andrew Ng, Michael Jordan, and Yair Weiss. On spectral clustering: Analysis and an algorithm. *Advances in neural information processing systems*, 14, 2001. 3, 13

[102] Thanh Toan Nguyen, Minh Tam Pham, Thanh Tam Nguyen, Thanh Trung Huynh, Quoc Viet Hung Nguyen, Thanh Tho Quan, et al. Structural representation learning for network alignment with self-supervised anchor links. *Expert Systems with Applications*, 165:113857, 2021. 56

[103] Maks Ovsjanikov, Mirela Ben-Chen, Justin Solomon, Adrian Butscher, and Leonidas J. Guibas. Functional maps: a flexible representation of maps between shapes. *ACM Trans. Graph.*, 31(4):30:1–30:11, 2012. 13, 22, 29, 31

[104] Lawrence Page, Sergey Brin, Rajeev Motwani, and Terry Winograd. The pagerank citation ranking: Bringing order to the web. Technical report, Stanford InfoLab, 1999. 11, 40

[105] Rob Patro and Carl Kingsford. Global network alignment using multiscale spectral signatures. *Bioinformatics*, 28(23):3105–3114, 2012. 65, 69

[106] Georgios A Pavlopoulos, Maria Secrier, Charalampos N Moschopoulos, Theodoros G Soldatos, Sophia Kossida, Jan Aerts, Reinhard Schneider, and Pantelis G Bagos. Using graph theory to analyze biological networks. *BioData mining*, 4(1):1–27, 2011. 56

[107] Marco Pegoraro, Riccardo Marin, Arianna Rampini, Simone Melzi, Luca Cosmo, and Emaneule Rodolà. Harnessing spectral representations for subgraph alignment. *arXiv preprint arXiv:2205.14938*, 2022. 19

[108] Shichao Pei, Lu Yu, Guoxian Yu, and Xiangliang Zhang. Graph alignment with noisy supervision. In *The Web Conference 2022 (TheWebConf)*, pages 1104–1114, 2022. 19

[109] Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. Deepwalk: Online learning of social representations. In *International Conference on Knowledge Discovery and Data Mining (KDD)*, pages 701–710, 2014. 29, 39

[110] Lu Qin, Rong-Hua Li, Lijun Chang, and Chengqi Zhang. Locally densest subgraph discovery. In *International Conference on Knowledge Discovery and Data Mining (KDD)*, pages 965–974, 2015. 23

[111] Jiezhong Qiu, Yuxiao Dong, Hao Ma, Jian Li, Kuansan Wang, and Jie Tang. Network embedding as matrix factorization: Unifying deepwalk, line, pte, and node2vec. In *WSDM*, pages 459–467, 2018. 39, 45

[112] Arianna Rampini, Irene Tallini, Maks Ovsjanikov, Alex M Bronstein, and Emanuele Rodolà. Correspondence-free region localization for partial shape similarity via hamiltonian spectrum alignment. In *International conference on 3D Vision (3DV)*, pages 37–46, 2019. 13, 90, 92, 93

[113] Leonardo FR Ribeiro, Pedro HP Saverese, and Daniel R Figueiredo. struc2vec: Learning node representations from structural identity. In *International Conference on Knowledge Discovery and Data Mining (KDD)*, pages 385–394, 2017. 29

[114] Charles Robertson. *Flowers and insects: lists of visitors to four hundred and fifty-three flowers*. 1928. 43

[115] Yu Rong, Wenbing Huang, Tingyang Xu, and Junzhou Huang. Dropedge: Towards deep graph convolutional networks on node classification. *arXiv preprint arXiv:1907.10903*, 2019. 3

[116] Ryan A. Rossi and Nesreen K. Ahmed. The network data repository with interactive graph analytics and visualization. `http://networkrepository.com`, 2015. 68, 80

[117] Indradyumna Roy, Venkata Sai Velugoti, Soumen Chakrabarti, and Abir De. Interpretable neural subgraph matching for graph retrieval. In *AAAI*, 2022. 23, 101

[118] Siddhartha Sahu, Amine Mhedhbi, Semih Salihoglu, Jimmy Lin, and M Tamer Özsu. The ubiquity of large graphs and surprising challenges of graph processing. *PVLDB*, 11(4):420–431, 2017. 56

[119] Shigeru Saito, Takatsugu Hirokawa, and Katsuhisa Horimoto. Discovery of chemical compound groups with common structures by a network analysis approach (affinity prediction method). *Journal of chemical information and modeling*, 51(1):61–68, 2011. 3

[120] Jose Costa Sapalo Sicato, Pradip Kumar Sharma, Vincenzo Loia, and Jong Hyuk Park. Vpnfilter malware analysis on cyber threat in smart home network. *Applied Sciences*, 9(13):2763, 2019. 3

[121] Vikram Saraph and Tijana Milenković. Magna: maximizing accuracy in global network alignment. *Bioinformatics*, 30(20):2931–2940, 2014. 65, 69

[122] Shruti Saxena, Roshni Chakraborty, and Joydeep Chandra. Hcna: Hyperbolic contrastive learning framework for self-supervised network alignment. *Information Processing & Management*, 59(5):103021, 2022. 19

[123] Peter H Schönemann. A generalized solution of the orthogonal procrustes problem. *Psychometrika*, 31(1):1–10, 1966. 41

[124] Fa-Bin Shi, Xiao-Qian Sun, Hua-Wei Shen, and Xue-Qi Cheng. Detect colluded stock manipulation via clique in trading network. *Physica A: Statistical Mechanics and its Applications*, 513:565–571, 2019. 3

[125] Jinbao Shi and Jitendra Malik. Normalized cuts and image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(8):888–905, 2000. 3, 4, 22, 95

[126] Rohit Singh, Jinbo Xu, and Bonnie Berger. Global alignment of multiple protein interaction networks with application to functional orthology detection. *PNAS*, 105(35):12763–12768, 2008. 6, 20, 56, 59, 67

[127] Konstantinos Skitas, Karol Orlowski, Judith Hermanns, Davide Mottin, and Panagiotis Karras. Comprehensive evaluation of algorithms for unrestricted graph alignment. In *Proceedings of the 26th International Conference on Extending Database Technology (EDBT)*, 2023, *to appear*. 6, 55, 107

[128] L. Sun, Z. Zhang, J. Zhang, F. Wang, Y. Du, S. Su, and P. S. Yu. Perfect: A hyperbolic embedding for joint user and community alignment. In *2020 IEEE International Conference on Data Mining (ICDM)*, pages 501–510, 2020. 4

[129] Shixuan Sun and Qiong Luo. Scaling up subgraph query processing with efficient subgraph matching. In *International Conference on Data Engineering (ICDE)*, pages 220–231, 2019. 23, 89

[130] Shixuan Sun, Xibo Sun, Yulin Che, Qiong Luo, and Bingsheng He. Rapidmatch: a holistic approach to subgraph query processing. *Proc. VLDB Endow.*, 14(2): 176–188, 2020. 23

[131] Alexandru Topirceanu, Gabriel Barina, and Mihai Udrescu. Musenet: Collaboration in the music artists industry. In *2014 European Network Intelligence Conference*, pages 89–94. IEEE, 2014. 3

[132] Amanda L Traud, Eric D Kelsic, Peter J Mucha, and Mason A Porter. Comparing community structure to characteristics in online collegiate social networks. *SIAM Rev.*, 53(3):526–543, 2011. 68

[133] Amanda L Traud, Peter J Mucha, and Mason A Porter. Social structure of facebook networks. *Phys. A*, 391(16):4165–4180, Aug 2012. 68

[134] Huynh Thanh Trung, Nguyen Thanh Toan, Tong Van Vinh, Hoang Thanh Dat, Duong Chi Thang, Nguyen Quoc Viet Hung, and Abdul Sattar. A comparative study on network alignment techniques. *Expert Systems with Applications*, 140: 112883, 2020. 57

[135] Anton Tsitsulin, Davide Mottin, Panagiotis Karras, Alexander M. Bronstein, and Emmanuel Müller. Netlsd: Hearing the shape of a graph. In *International Conference on Knowledge Discovery and Data Mining (KDD)*, pages 2347–2356, 2018. 4, 22, 29, 33, 38, 65

[136] Shinj Umeyama. An eigendecomposition approach to weighted graph matching problems. *IEEE Trans. Pattern Anal. Mach. Intell.*, 10(5):695–703, 1988. 21

[137] Alexei Vázquez, Alessandro Flammini, Amos Maritan, and Alessandro Vespignani. Modeling of protein interaction networks. *Complexus*, 1(1), 2003. 56

[138] Vipin Vijayan and Tijana Milenković. Multiple network alignment via multimagna++. *IEEE/ACM transactions on computational biology and bioinformatics*, 15(5):1669–1682, 2017. 43, 68

[139] Hua Wang and Barry Wellman. Social connectivity in america: Changes in adult friendship network size from 2002 to 2007. *American behavioral scientist*, 53(8):1148–1169, 2010. 3

[140] Wei Wang, Zheng Dang, Yinlin Hu, Pascal Fua, and Mathieu Salzmann. Backpropagation-friendly eigendecomposition. 2019. 96

[141] Duncan J Watts and Steven H Strogatz. Collective dynamics of 'small-world' networks. *nature*, 393(6684):440–442, 1998. 67, 68

[142] John Henry Wilkinson, Friedrich Ludwig Bauer, and C Reinsch. *Linear algebra*, volume 2. 2013. 9

[143] Yujia Xie, Xiangfeng Wang, Ruijia Wang, and Hongyuan Zha. A fast proximal point method for computing exact Wasserstein distance. In *UAI*. PMLR, 2020. 64

[144] Hao Xiong, Junchi Yan, and Li Pan. Contrastive multi-view multiplex network embedding with applications to robust network alignment. In *International Conference on Knowledge Discovery and Data Mining (KDD)*, pages 1913––1923, 2021. 20

[145] Hongteng Xu, Dixin Luo, and Lawrence Carin. Scalable gromov-wasserstein learning for graph partitioning and matching. *NeurIPS*, 32, 2019. 6, 20, 21, 30, 40, 44, 48, 59, 64

[146] Hongteng Xu, Dixin Luo, Hongyuan Zha, and Lawrence Carin Duke. Gromov-Wasserstein learning for graph matching and node embedding. In *ICML*, volume 97 of *PMLR*, pages 6932–6941, 09–15 Jun 2019. 6, 21, 44, 59, 63, 66

[147] Mengmeng Xu, Chen Zhao, David S Rojas, Ali Thabet, and Bernard Ghanem. G-tad: Sub-graph localization for temporal action detection. In *IEEE/CVF CVPR*, pages 10156–10165, 2020. 23

[148] Yuchen Yan, Si Zhang, and Hanghang Tong. BRIGHT: A bridging algorithm for network alignment. In *The Web Conference (TheWebConf)*, pages 3907–3917, 2021. 19, 20, 29

[149] Jaewon Yang and Jure Leskovec. Community-affiliation graph model for overlapping network community detection. In *2012 IEEE 12th international conference on data mining*, pages 1170–1175. IEEE, 2012. 3

[150] Lyudmila Yartseva and Matthias Grossglauser. On the performance of percolation graph matching. In *ACM COSN*, pages 119–130, 2013. 19

[151] Abdurrahman Yasar and Ümit V Çatalyürek. An iterative global structure-assisted labeled network aligner. In *International Conference on Knowledge Discovery and Data Mining (KDD)*, pages 2614–2623, 2018. 21

[152] Tianshu Yu, Junchi Yan, Yilin Wang, Wei Liu, and Baoxin Li. Generalizing graph matching beyond quadratic assignment model. In *Annual Conference on Neural Information Processing Systems (NeurIPS)*, pages 861–871, 2018. 13

[153] Minying Zhang, Kai Liu, Yidong Li, Shihui Guo, Hongtao Duan, Yimin Long, and Yi Jin. Unsupervised domain adaptation for person re-identification via heterogeneous graph alignment. In *Proceedings of the AAAI conference on artificial intelligence*, volume 35, pages 3360–3368, 2021. 4

[154] Muhan Zhang, Zhicheng Cui, Marion Neumann, and Yixin Chen. An end-to-end deep learning architecture for graph classification. In *Proceedings of the AAAI conference on artificial intelligence*, volume 32, 2018. 3

[155] Si Zhang and Hanghang Tong. Final: Fast attributed network alignment. In *KDD*, pages 1345–1354, 2016. 4, 5, 20, 23, 28, 56, 66, 90, 108

[156] Si Zhang, Hanghang Tong, Long Jin, Yinglong Xia, and Yunsong Guo. Balancing consistency and disparity in network alignment. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, pages 2212–2222, 2021. 19, 20

[157] Fan Zhou, Lei Liu, Kunpeng Zhang, Goce Trajcevski, Jin Wu, and Ting Zhong. Deeplink: A deep learning approach for user identity linkage. In *INFOCOM*, pages 1313–1321, 2018. 20

[158] Qinghai Zhou, Liangyue Li, Xintao Wu, Nan Cao, Lei Ying, and Hanghang Tong. Attent: Active attributed network alignment. In *The Web Conference 2021*, pages 3896–3906, 2021. 20

[159] Yang Zhou, Zeru Zhang, Sixing Wu, Victor S. Sheng, Xiaoying Han, Zijie Zhang, and Ruoming Jin. Robust network alignment via attack signal scaling and adversarial perturbation elimination. In *The Web Conference 2021*, pages 3884–3895, 2021. 20, 29

[160] Daniel Zügner and Stephan Günnemann. Certifiable robustness of graph convolutional networks under structure perturbations. In *International Conference on Knowledge Discovery and Data Mining (KDD)*, pages 1656–1665, 2020. 109