AARHUS UNIVERSITY

This is the accepted manuscript (AM)/author accepted manuscript (AAM) of the article

The content in the accepted manuscript version has been peer reviewed (when applicable) and accepted for publication, though any post-acceptance changes such as typography and layout may lead to differences between this version and the final published version.

How to cite this publication

Please cite the final published version:

Judith Hermanns, Konstantinos Skitsas, Anton Tsitsulin, Marina Munkhoeva, Alexander Kyster, Simon Nielsen, Alexander M. Bronstein, Davide Mottin, and Panagiotis Karras. 2023. GRASP: Scalable Graph Alignment by Spectral Corresponding Functions. ACM Trans. Knowl. Discov. Data 17, 4, Article 50 (May 2023), 26 pages. https://doi.org/10.1145/3561058

© Author 2023. This is the author's version of the work. It is posted here for your personal use. Not for redistribution. The definitive Version of Record was published in ACM Trans. Knowl. Discov. Dat, https://doi.org/10.1145/3561058

General Rights

Copyright and moral rights for the publications made accessible in the public portal are retained by the authors and/or other copyright owners and it is a condition of accessing publications that users recognize and abide by the legal requirements associated with these rights.

- *Users may download and print one copy of any publication from the public portal for the purpose of private study or research.*
- *You may not further distribute the material or use it for any profit-making activity or commercial gain.*
- *You may freely distribute the URL identifying the publication in the public portal.*

If you believe that this document breaches copyright please contact us at oa@kb.dk providing details, and we will remove access to the work immediately and investigate your claim.

If the document is published under a Creative Commons license, this applies instead of the general rights.

This cover sheet template is made available by AU Library
Version 3, October 2023

GRASP: Scalable Graph Alignment by Spectral Corresponding Functions

JUDITH HERMANNS, Aarhus University, Denmark
KONSTANTINOS SKITSAS, Aarhus University, Denmark
ANTON TSITSULIN, Google Research, USA
MARINA MUNKHOEVA, Max Planck Institute for Intelligent Systems, Germany
ALEXANDER FREDERIKSEN KYSTER, Aarhus University, Denmark
SIMON DAUGAARD NIELSEN, Aarhus University, Denmark
ALEX BRONSTEIN, Technion, Israel
DAVIDE MOTTIN, Aarhus University, Denmark
PANAGIOTIS KARRAS, Aarhus University, Denmark

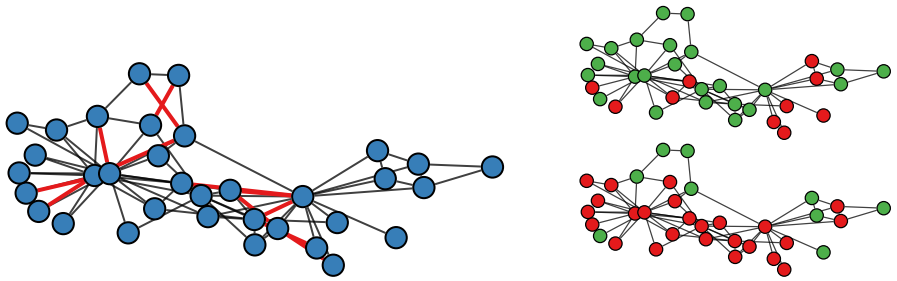
What is the best way to match the nodes of two graphs? This *graph alignment* problem generalizes graph isomorphism and arises in applications from social network analysis to bioinformatics. Some solutions assume that auxiliary information on known matches or node or edge attributes is available, or 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.

ACM Reference Format:
Judith Hermanns, Konstantinos Skitsas, Anton Tsitsulin, Marina Munkhoeva, Alexander Frederiksen Kyster, Simon Daugaard Nielsen, Alex Bronstein, Davide Mottin, and Panagiotis Karras. 2022. GRASP: Scalable Graph Alignment by Spectral Corresponding Functions. *ACM Trans. Knowl. Discov. Data.* 37, 4, Article 111 (August 2022), 25 pages. https://doi.org/XXXXXXX.XXXXXXX

Authors' addresses: Judith Hermanns, Aarhus University, Aarhus, Denmark; Konstantinos Skitsas, Aarhus University, Aarhus, Denmark; Anton Tsitsulin, Google Research, New York, USA; Marina Munkhoeva, Max Planck Institute for Intelligent Systems, Tübingen, Germany; Alexander Frederiksen Kyster, Aarhus University, Aarhus, Denmark; Simon Daugaard Nielsen, Aarhus University, Aarhus, Denmark; Alex Bronstein, Technion, Haifa, Israel; Davide Mottin, Aarhus University, Aarhus, Denmark; Panagiotis Karras, Aarhus University, Aarhus, Denmark.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
© 2022 Association for Computing Machinery.
1556-4681/2022/8-ART111 $15.00
https://doi.org/XXXXXXX.XXXXXXX



(a) Karate club; red edges removed. (b) Alignment by GRASP (top), REGAL (bottom).

Fig. 1. With a few removed edges, REGAL [20], 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).

1 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 [28]. 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 de-anonymizing [44, 63] and identifying users in different social networks [25], matching objects in images by establishing feature correspondences and comprehending protein response in the body [26].

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 [8, 38]. 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 [20], *exploiting additional information* such as node attributes [67] or bipartite networks [31], or optimizing objectives based only on *local connections* among nodes [15, 36, 41]. On the other hand, the spectra of *Laplacian matrices* have been successfully employed to devise a similarity measure among graphs [55]. 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 [15, 36, 41] 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 [9]. We transfer the methodology of matching among shapes based on *corresponding functions* [45] 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 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 [20] 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 [36, 41] or integer programming [26] that were hard to adapt, are designed for a specific graph type, e.g., biological [3] or bipartite networks [31], and fare poorly on other types [20]. Embedding-based methods [64, 71] 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 [20] and CONE [7], that leverage advances in node embeddings [47, 49, 56, 57]; 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, [21] and [34]; it expands and completes the experimental study in [21] by applying to GRASP the enhancements introduced in [34] 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 5).
(2) We include the recently proposed graph alignment methods S-GWL [61] and CONE [7] in our experimental comparison (Section 6).
(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 6.1).
(4) We conduct new experiments on real-world data, namely two temporal proximity networks and different variants of a PPI-network (Table 2, Section 6.4).
(5) We provide additional experiments examining scalability in the number of nodes in a graph (Section 6.6).

2 RELATED WORK

We distinguish graph alignment methods introduced in related work in two main categories: (i) **restricted alignment** methods, which require a ground-truth mapping or information besides a binary adjacency matrix; and (ii) **unrestricted alignment** methods, which require neither supervision nor information besides an adjacency matrix. Table 1 overviews the mean features of related works.

Method	Unrestrict.	Spectral	Flexible	Precomp.	Multiscale	Modular	Scalable	Functional
BigAlign [31]	✘	✘	✘	✘	✘	✘	✘	✘
FINAL [67]	✘	✘	✘	✘	✘	✘	✔	✘
IsoRank [54]	✘	✘	✔	✔	✘	✘	✘	✘
REGAL [20]	✘	✘	✔	✔	✘	✔	✔	✘
GRAMPA [14]	✔	✔	✔	✘	✘	✘	✔	✘
LaplMatch [27]	✔	✔	✘	✔	✘	✘	✘	✘
LREA [41]	✔	✔	✘	✘	✘	✘	✔	✘
CONE [7]	✔	✘	✔	✔	✘	✔	✘	✘
S-GWL [61]	✔	✘	✔	✔	✔	✘	✘	✘
GRASP	✔	✔	✔	✔	✔	✔	✔	✔

Table 1. Related work in terms of present (✔) and absent (✘) properties: *unrestricted* methods work by default with *plain* graph structures as input; IsoRank requires auxiliary node similarity input; FINAL requires node or edge attribute information to perform reasonably well [20]; *spectral* methods use the spectra of alignment matrices; *flexible* methods accommodate different alignment algorithms (e.g. bipartite matching, nearest neighbors). Among unrestricted methods (rows 5–10), 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, REGAL, GRAMPA, LREA, 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.

2.1 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) [25, 65] propagates ground-truth alignments across the network. *Representation learning* approaches, such as IONE [38], PALE [40], and DeepLink [69], 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 [8]. Other recent embedding-based alignment methods, such as cM^{2}NE [60], SSPDG [71] and NEXTALIGN [68] require pre-aligned seed nodes. BRIGHT [64] uses embeddings and *neighborhood consistency*, by which an alignment should preserve the neighborhoods of aligned nodes. *Active network alignment* [39] and, recently, ATTENT [70] apply active learning to elicit expert guidance on alignments. DANA [12], 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 [31] focuses on bipartite graphs; however, most graphs are not bipartite. FINAL [67] aligns nodes based on similarity of topology and attributes. IsoRank [54] 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 [29] and in IsoRankN [36], 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 [66] lets pairwise distances to seed nodes guide the matching. Another variant matches *weigthed* matrices using their spectra [58];

that is inapplicable to the unweighted case. Karakasis et al. [23] propose a refinement method for precomputed alignments, which qualifies as an assisted method rather than a stand-alone alignment algorithm. WAlign [18] 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. REGAL [20] constructs node embeddings based on the connectivity structure and node attributes, and uses the similarity between these features for node alignment. Similarly to IsoRank and FINAL, REGAL also works in the unrestricted case where no attributes are available, and, moreover, performs well at that; hence we use its unrestricted version in our experiments.

2.2 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 [26] 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. CONE [7] realigns node representations, without prejudice to the representation used. GWL [62] 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 [61], a scalable version of GWL, achieves scalability by aligning progressively smaller graph partitions.

Matrix decomposition methods. EigenAlign [15] 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) [41] 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 [4] solves a specific *sparse* variant of the graph alignment problem by a message-passing algorithm.

2.3 Shape Matching

Our work is inspired by shape matching methods that employ spectral properties [32, 37, 45]. Functional maps [45] 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 [32], and match a part of a shape to another full shape in the spectral domain [37] without requiring spatially modeling the part of a shape.

2.4 Spectral Methods

Graph spectra [9] facilitate problem-solving in graph analysis, image partitioning, graph search, and machine learning [53]. NetLSD [55] uses Laplacian spectral signatures to detect graph similarity, *but not to align graphs*, in a multi-scale fashion. LaplMatch [27] 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 [10].

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* $A \in \{0,1\}^{n \times n}$ is a binary matrix where $A_{ij} = 1$ if there is an edge between nodes i and j and $A_{ij} = 0$ otherwise.

Definition 1. *Given two graphs $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$, a graph alignment $R : V_1 \longrightarrow V_2$ 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 P, minimizes the difference $\|PA_1P^\top - A_2\|^2$. In case of isomorphic graphs, there exists a P such that $PA_1P^\top = A_2$, 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 $\tau : V_1 \longrightarrow V_2$ 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 τ is a bijection that admits an inverse mapping $\tau^{-1} : V_2 \longrightarrow V_1$. The composition of the indicator function $\delta_i : V_1 \longrightarrow \{0,1\}$ with τ^{-1}, $\delta_i \circ \tau^{-1} : V_2 \longrightarrow \{0,1\}$ 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 $f_1, ..., f_q, f_i : V_1 \longrightarrow \mathbb{R}$ and $g_1, ..., g_q, g_i : V_2 \longrightarrow \mathbb{R}$ 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 τ. 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 [45] from the continuous to the discrete case.

Functional maps. The operator $T_{\mathcal{F}} : (V_1 \times \mathbb{R}) \longrightarrow (V_2 \times \mathbb{R})$ maps functions f on the nodes in G_1 to functions g on the nodes in G_2, i.e. $T_{\mathcal{F}}(f) = f \circ \tau^{-1} = g$. This operator is linear in the function space, i.e., $T_{\mathcal{F}}(c_1 f_1 + c_2 f_2) = (c_1 f_1 + c_2 f_2) \circ \tau^{-1} = c_1 f_1 \circ \tau^{-1} + c_2 f_2 \circ \tau^{-1} = c_1 T_{\mathcal{F}}(f_1) + c_2 T_{\mathcal{F}}(f_2)$. 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 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 $T_{\mathcal{F}}$,

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

where $T_{\mathcal{F}}(\phi_i) = \sum_{j=1}^{n} c_{ij} \psi_j$. 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 $T_{\mathcal{F}}$. In conclusion, in order to align real-valued functions on the nodes of two graphs, we need to find a *mapping matrix* $C \in \mathbb{R}^{n \times n}$ of coefficients among those functions; such a mapping matrix C maps functions from G_1 to G_2, even when the ground-truth mapping τ is unknown. In a nutshell, GRASP obtains such a mapping matrix C for a well-chosen function and applies that C to mapping the indicator function δ 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.

[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.

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 4.1 we choose a basis for the functions. In Section 4.2, we choose functions that capture important graph properties. Then, in Section 4.3, we define a mapping matrix to map functions across graphs. In Section 4.4, we compute node-to-node alignments from such a map, and Section 4.5 describes a method to refine the mapping matrix.

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 $\mathcal{L} = I - D^{-\frac{1}{2}}AD^{-\frac{1}{2}}$, where D a diagonal degree matrix of node degrees $D_{ii} = \sum_{j=1}^{n} A_{ij}$ and A is the graph adjacency matrix; its eigendecomposition is $\mathcal{L} = \Phi\Lambda\Phi^\top$, where Λ 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 Φ is a matrix of corresponding eigenvectors, $\Phi_{\mathcal{L}} = [\phi_1\phi_2 \ldots \phi_n]$. The eigenvectors form an orthogonal basis, which we use a standard basis. We use ϕ (ψ) 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 [5], which measures the smoothness of continuous surfaces.

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* [55]:

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

where $H_{t[ij]}$ 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 (1) emphasises large λ, 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 $F \in \mathbb{R}^{n\times q}$, $F = [f_1, \ldots, f_q]$ be the matrix containing the diagonals of the heat kernel of G_1, $H_t^{G_1}$, over q time[2] steps, $f_i = \sum_{j=1}^{n} e^{-t_i\lambda_j}\phi_j \odot \phi_j$, where $\odot$ denotes the element-wise vector product. Likewise, the matrix

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

$G \in \mathbb{R}^{n \times q}, G = [g_1, \ldots, g_q]$ contains the diagonals of $H_t^{G_2}$, 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.

4.3 Mapping matrix

We approximate each function f_i using only the first k eigenvectors, as done, by analogy, on shapes analysis [5], and thereby calculate the corresponding function matrices F and G. F and G can be thought as coefficient matrices used to produce linear combinations, $F^\top \Phi$ and $G^\top \Psi$, of the Laplacian eigenvectors of G_1 and G_2, respectively. With a slight abuse of notation, we denote with Φ and Ψ the first k eigenvectors, hence $F^\top \Phi$ and $G^\top \Psi$ 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 $C \in \mathbb{R}^{k \times k}$. In the case of isomorphic graphs, it holds that $F^\top \Phi = G^\top \Psi C$

$$\begin{bmatrix} \mathrm{diag}(g_1^\top \Psi) \\ \vdots \\ \mathrm{diag}(g_q^\top \Psi) \end{bmatrix} \begin{bmatrix} c_{11} \\ \vdots \\ c_{kk} \end{bmatrix} = \begin{bmatrix} \Phi^\top f_1 \\ \vdots \\ \Phi^\top f_q \end{bmatrix} \tag{2}$$

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. (2) using the ordinary least squares method, as in [32]. In Section 4.5 we delve into the case of non-isomorphic graphs.

4.4 Node-to-node correspondence

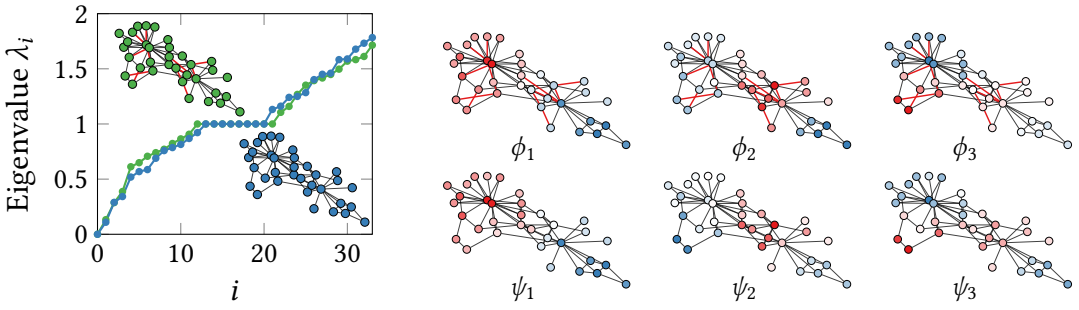
We consider the delta function $\delta_i(\cdot)$ as corresponding function; these functions yield an $n \times n$ identity matrix. We express such a function as a vector of coefficients, since the vector of δ_i is the ith row of the heat kernel at $t = 0$:

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

The computation for delta functions on G_2 follows equivalently using Ψ in place of Φ. 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 δ_i and δ_j for Φ and Ψ should be identical. In particular, the coefficients expressing δ_i as a linear combination of the first k eigenvectors are $\phi_{i1}, \ldots, \phi_{ik}$. We set $\Phi^\top$ and $C\Psi^\top$ in $\mathbb{R}^{k \times n}$ as coefficient matrices of the delta functions, aligned by 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 $C\Psi^\top$, 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)** [22], 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.

4.5 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 C exists. Still, if the graphs are not isomorphic, then their eigenvectors diverge and the diagonal matrix C, which we enforce, cannot capture their relationship well. Figure 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* [32] in terms of an alignment matrix M.



(a) Two graphs spectra (b) Their first 3 eigenvectors

Fig. 2. We remove **red** edges from the **green** graph to obtain the **blue** graph. Eigenvalues interlace (a); eigenvectors ϕ_1, ϕ_2, ϕ_3 for **green** and ψ_1, ψ_2, ψ_3 for **blue** highlight common structures in their corresponding entries (b).

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

Since the eigenvectors are orthonormal, $\Psi^\top\Psi = I$ and for G_2's graph Laplacian eigenvectors Λ_2, $\Psi^\top\mathcal{L}_2\Psi = \Psi^\top\Psi\Lambda_2 = \Lambda_2$, and $M^\top\Psi^\top\mathcal{L}_2\Psi M = M^\top\Lambda_2 M$. Putting the above together, our diagonalizing term is:

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

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

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

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

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

where F and 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 μ[3]:

$$\min_{M \in S(n,n)} \text{off}(M^\top\Lambda_2 M) + \mu\|F^\top\Phi - G^\top\Psi M\|_F^2 \tag{3}$$

Given that the eigenvectors of isomorphic graphs match with sign changes, we initialize M as a diagonal matrix with $M_{ii} = 1$ if $\|F^\top\phi_i - G^\top\psi_i\| \leq \|F^\top\phi_i + G^\top\psi_i\|$, $M_{ii} = -1$ otherwise. Eq. (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 C (see Section 4.3). So we only align the first k eigenvectors of Ψ to the first k eigenvectors of Φ, i.e $\bar{\Phi} = \hat{\Psi} = \bar{\Psi}M$ with $\bar{\Phi} = [\phi_1, \ldots, \phi_k]$ and $\bar{\Psi} = [\psi_1, \ldots, \psi_k]$. Let $\bar{\Lambda}_2 = \text{diag}(\lambda_1, \ldots, \lambda_k)$, the problem in Eq. (3) becomes

$$\min_{M \in S(k,k)} \text{off}(M^\top\bar{\Lambda}_2 M) + \mu\|F^\top\bar{\Phi} - G^\top\bar{\Psi}M\|_F^2 \tag{4}$$

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

After obtaining M, we use the eigenvectors in $\bar{\Phi}$ and the aligned eigenvectors $\hat{\Psi} = \bar{\Psi}M$ 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 $t = [t_1, \ldots, t_q]$
// Step 1: Eigendecomposition of Laplacian
$\mathcal{L}_1 \leftarrow I - D_1^{-\frac{1}{2}} A_1 D_1^{-\frac{1}{2}}$, $\mathcal{L}_2 \leftarrow I - D_2^{-\frac{1}{2}} A_2 D_2^{-\frac{1}{2}}$
$\Phi, \Lambda_1 \leftarrow \mathrm{eig}(\mathcal{L}_1)$
$\Psi, \Lambda_2 \leftarrow \mathrm{eig}(\mathcal{L}_2)$
// Step 2: Compute corresponding functions
for all t_i in t **do**
  $f_i \leftarrow \sum_{j=1}^{n} e^{-t_i \lambda_j} \phi_j \odot \phi_j$
  $g_i \leftarrow \sum_{j=1}^{n} e^{-t_i \lambda_j} \psi_j \odot \psi_j$
$F \leftarrow [f_1, \ldots, f_q]$
$G \leftarrow [g_1, \ldots, g_q]$
// Step 3: Base alignment
$\bar{\Phi} = [\phi_1, \ldots, \phi_k]$
$\bar{\Psi} = [\psi_1, \ldots, \psi_k]$
$M \leftarrow \min_{M \in S(k,k)} \mathrm{off}(M^\top \bar{\Lambda}_2 M) + \mu \|F^\top \bar{\Phi} - G \bar{\Psi} M\|_F^2$
$\hat{\Psi} = \bar{\Psi} M$
// Step 4: Calculate mapping matrix

$$\min_{[c_{11},\ldots,c_{kk}]^\top} \left\| \begin{bmatrix} \mathrm{diag}(g_1^\top \hat{\Psi}) \\ \vdots \\ \mathrm{diag}(g_q^\top \hat{\Psi}) \end{bmatrix} \begin{bmatrix} c_{11} \\ \vdots \\ c_{kk} \end{bmatrix} - \begin{bmatrix} \bar{\Phi}^\top f_1 \\ \vdots \\ \bar{\Phi}^\top f_q \end{bmatrix} \right\|_2^2$$

$C \leftarrow \mathrm{diag}([c_{11}, \ldots, c_{kk}]^\top)$
// Step 5: Matching by linear assignment
$N \leftarrow$ Matching of columns of $\bar{\Phi}^\top$ to those of $C\hat{\Psi}^\top$
return N

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 $\mathcal{L}_1, \mathcal{L}_2$ of the two graphs G_1 and G_2. Then compute the eigenvectors Φ, Ψ and eigenvalues Λ_1, Λ_2 by the eigendecomposition $\mathcal{L}_1 = \Phi \Lambda_1 \Phi^\top$ and $\mathcal{L}_2 = \Psi \Lambda_2 \Psi^\top$.

Step 2: Compute corresponding functions. In the second step, calculate the matrices of corresponding functions $F = [f_1, \ldots, f_q]$ and $G = [g_1, \ldots, g_q]$ as diagonals of the heat kernel at time steps $[t_1, \ldots, t_q]$ with $f_i = \sum_{j=1}^{n} e^{-t_i \lambda_j} \phi_j \odot \phi_j$ and g_i equivalently using Ψ.

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

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

$$\min_{[c_{11},\ldots,c_{kk}]^\top} \left\| \begin{bmatrix} \mathrm{diag}(g_1^\top \hat{\Psi}) \\ \vdots \\ \mathrm{diag}(g_q^\top \hat{\Psi}) \end{bmatrix} \begin{bmatrix} c_{11} \\ \vdots \\ c_{kk} \end{bmatrix} - \begin{bmatrix} \bar{\Phi}^\top f_1 \\ \vdots \\ \bar{\Phi}^\top f_q \end{bmatrix} \right\|_2^2 \tag{5}$$

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

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

Complexity analysis. The computation of the first k Laplacian eigenvectors takes $O(k \max\{|E_1|, |E_2|\})$ by fast methods for diagonally dominant matrices [30]. Base alignment needs $O(k^3)$ to solve the orthogonality constraint through trust-region methods. The least-squares method runs in $O(q\,k)$. The matching step by JV runs in $O(n^3)$. Overall, the $O(n^3)$ time factor is dominant.

Connection to Differential Geometry. Our work rests on the theory on Riemannian manifolds [17] and builds on the analogy between a graph's Laplacian and the continuous Laplace-Beltrami operator [55].

5 BOOSTING GRASP

Having introduced GRASP, we observe that most state-of-the-art methods for graph alignment leverage advances in graph representation learning [19], 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 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.

5.1 Modular Graph Alignment

GRASP belongs to a family of graph alignment algorithms that expose a *modular* structure [34]. Besides GRASP, the recently proposed CONE [7] and its predecessor REGAL [20] 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 [47]. For GRASP, we choose the spectral embeddings as described in Section 4.4. CONE uses NetMF [48] 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 each graph's node 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 4.5. 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 [22].

Monolithic Alignment Algorithms. In this context it is worth pointing out that two state-of-the-art graph alignment algorithms, S-GWL [61], and LREA [41] 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.

5.2 PageRank (PR)

We develop an embedding based on PageRank [46]. PR is real-valued vector that represents the importance of each node, defined as $\mathbf{r}_j = \sum_{(i,j)\in E} \alpha \frac{\mathbf{r}_i}{d_i} + (1-\alpha)\mathbf{p}$, where $\alpha \in [0,1]$ is the damping factor and $p_i = 1/n$ for each i. We generate corresponding functions sampling PR at different α values. 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.

5.3 Iterative Closest Point

The Iterative Closest Point (ICP) method [6] 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 X and iteratively transforming the source cloud P. The algorithm starts with an initial alignment Y between the points of each cloud and proceeds transforming the

source points with a mapping matrix that minimizes the least-squared difference between P and the alignment Y. This matrix is obtained by solving the orthogonal Procrustes problem (OPP) [52]. 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 ϵ. Time complexity is dominated by the closest point algorithm, which is $O(n \log n)$ using a k-d tree [6]. Algorithm 3 illustrates the method.

Algorithm 3 Iterative Closest Point

1: **repeat**
2: $k = k + 1$.
3: Compute k-th alignment matrix $Y_k = ClosestPoint(P_k, X)$.
4: Find mapping matrix $C_k = OPP(P_k, Y_k)$.
5: Rotate P_k via C_k to obtain $P_{k+1} = C_k P_k$.
6: **until** $MSE(P_k, Y_k) - MSE(P_{k+1}, Y_{k+1}) < \epsilon$

We apply ICP in GRASP to refine the initial node alignment (mapping) matrix C_0 that we obtain as in Section 4.3.

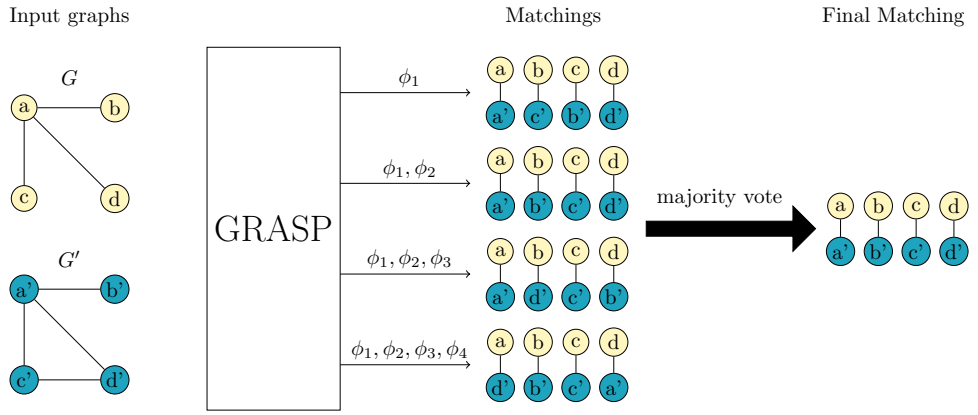


Fig. 3. An example of the voting heuristic in action; GRASP runs four times, adding one eigenvector in each iteration, from ϕ_1 to ϕ_4. We omit eigenvectors $\psi_1, \ldots, \psi_4$ for the sake of readability. Each set of eigenvectors returns a matching from nodes a, b, c, d to a', b', c', d'; on node a the match a–a' is selected by 3 out of 4 eigenvector sets and chosen as the output match.

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.

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 4.3) 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 [13] 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 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.

6 EXPERIMENTS

In this section, we present our thorough experimental study. Table 2 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 [20, 31]; 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 [20, 31]; 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	$\|V\|$	$\|E\|$	Network type	Ground-truth	Bipartite
Arenas Email [33]	1 133	5 451	communication	✘	✘
Facebook-ego [35]	4 039	88 234	social	✘	✘
CA-AstroPh [35]	17 903	197 031	collaboration	✘	✘
Hamsterer [33]	2 000	16 097	social	✘	✘
PPI [6]	3 852	38 705	biological	✘	✘
Voles [11]	712	2391	proximity	✔	✘
MultiMagna [59]	1004	8323	biological	✔	✘
HighSchool [16]	327	5818	proximity	✔	✘
plantpolrobertson [50]	1 884	15 255	biological	✘	✔
chowiki [33]	195	352	co-authorship	✘	✔
tpiwiktionary [33]	861	2 079	co-authorship	✘	✔

Table 2. Datasets used in our evaluation, $|V|$ number of nodes, $|E|$ number of vertices.

Baselines. We compare against the following established baselines for *unrestriced* graph alignment.

- **REGAL** [20]: 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)** [41]: A spectral method that yields one-to-one matchings via the minimization of edge mismatches.

- **CONE** [7]: 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** [61]: A monolithic alignment method that jointly computes embeddings and alignments using optimal transport. S-GWL is a scalable variant of GWL [62].
- **GRAMPA** [14]: A monolithic alignment method that calculates an alignment based on a similarity matrix of embeddings derived from eigenvectors of the adjacency matrix.

All codes[4] are in Python. We used available[5] implementations for REGAL, CONE, and S-GWL and re-implemented GRAMPA and LREA. 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: $acc = \frac{|A_{cor}|}{|A_{gt}|}$.

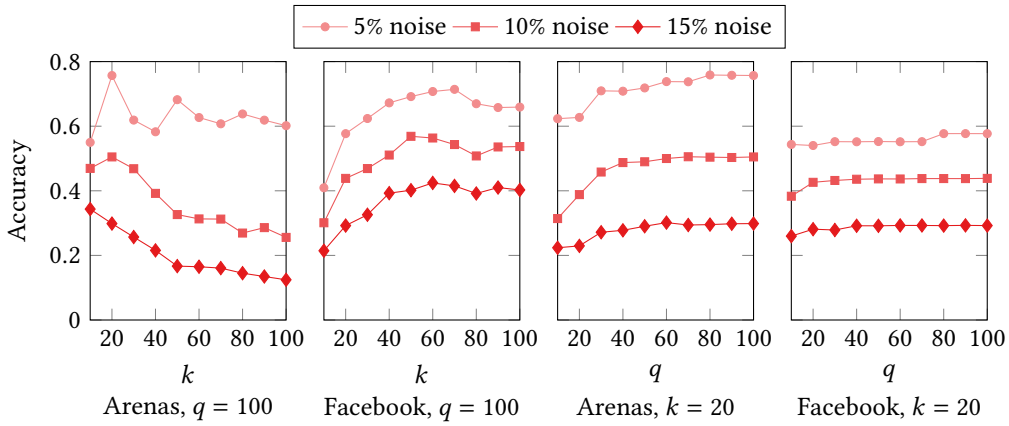


Fig. 4. Accuracy on the Facebook and Arenas graphs for different numbers of corresponding functions q and different numbers of eigenvectors k for noise levels 5%, 10% and 15%.

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 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 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

[4]Our codes are available at https://github.com/AU-DIS/GRASP.
[5]https://github.com/GemsLab/REGAL, https://github.com/GemsLab/CONE-Align, https://github.com/HongtengXu/s-gwl.

rightmost charts in Figure 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 $q = 100$.

6.2 Selecting components

Here, we further exploit the modularity of GRASP (Section 5.1), aiming to select best-performing modular components. For the sake of a fair comparison, we also examine whether CONE [7] may benefit from a different choice of embedding other than the default NetMF [48].

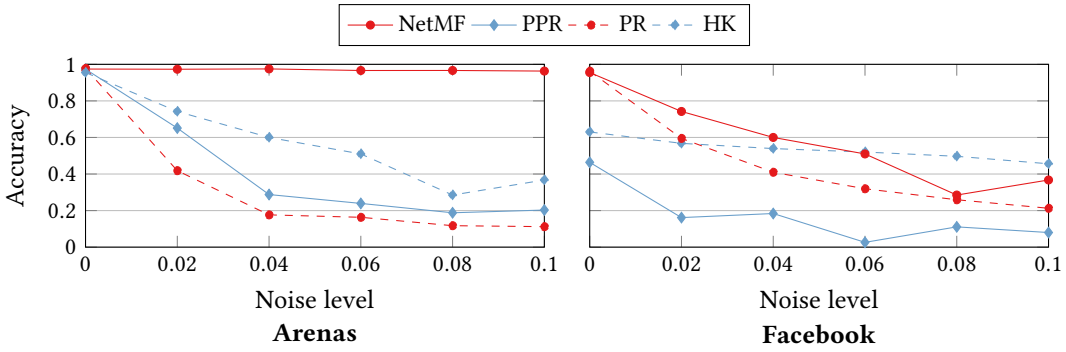


Fig. 5. Accuracy of CONE-Align with different embeddings on the Arenas and Facebook.

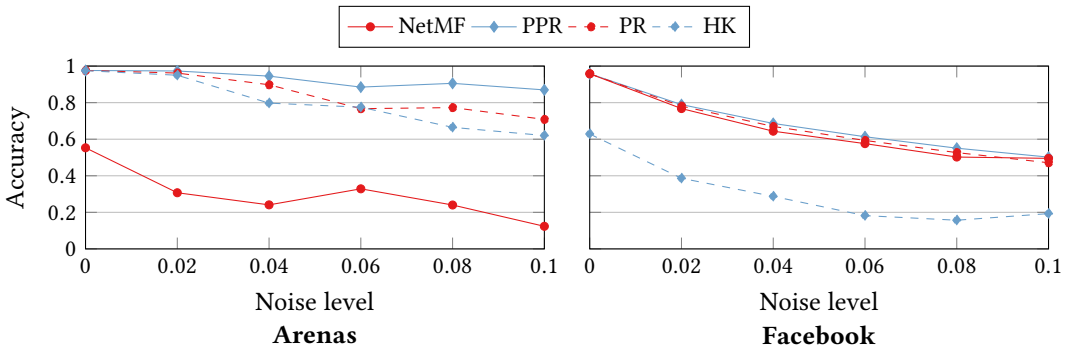


Fig. 6. Accuracy of GRASP (ICP, voting and JV) with PPR, PR, HK and NetMF embeddings.

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* $H_t = \Phi e^{-t\Lambda}\Phi^\top = \sum_{j=1}^{n}$ for varying t. Figure 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 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.

Boosting Align. Next, we investigate the impact of the enhancements in Section 5 on GRASP-PPR. Figure 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.

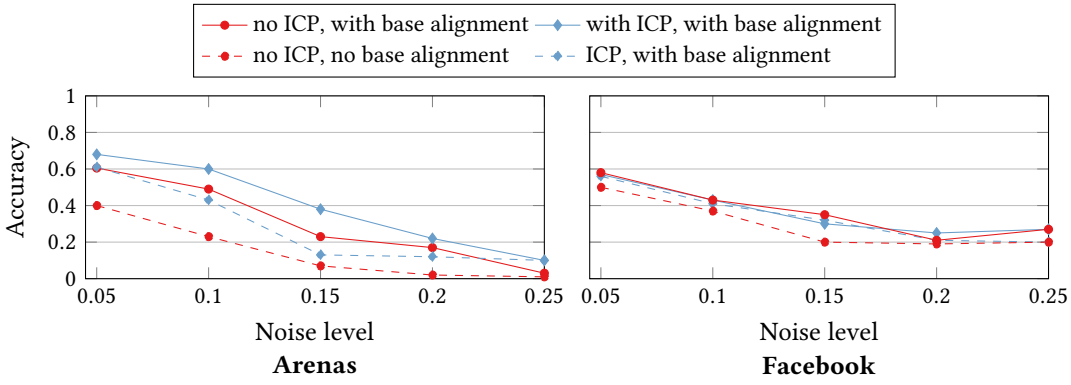


Fig. 7. Accuracy of GRASP-PPR with ICP and BA.

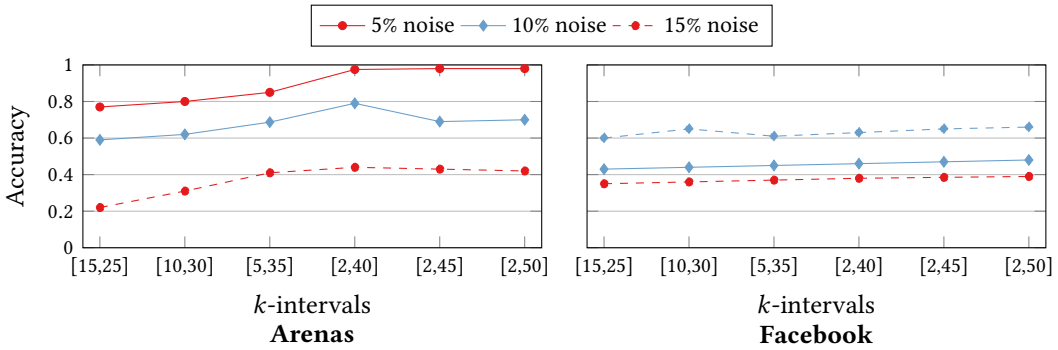


Fig. 8. Accuracy of GRASP-PPR with ICP using the SortGreedy voting procedure with different intervals of k

Figure 8 showcases the performance of voting with different amounts of k, using SortGreedy (SG) [13] both for node assignment while collecting votes and for the final assignment based on collected votes; other, more computationally demanding choices brought negligible improvements.

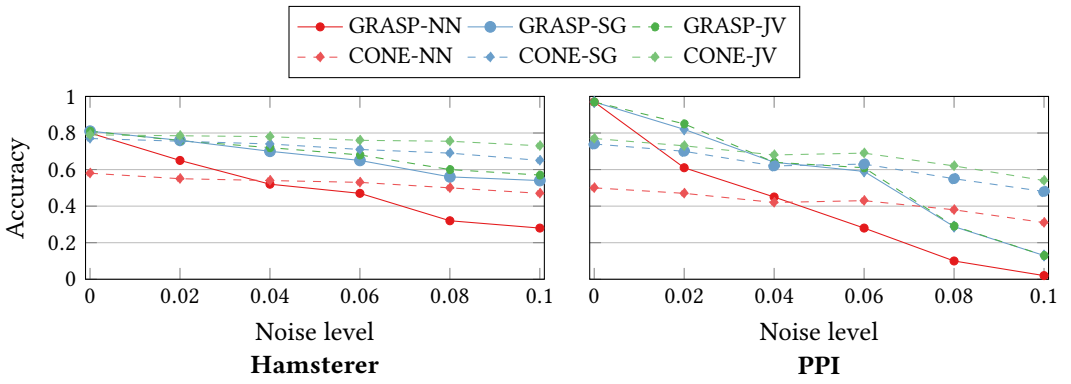


Fig. 9. Accuracy with different assignment algorithms.

Boosting Assign. We also examine the effect of the linear assignment algorithm in more detail. We try out nearest-neighbor (NN), SortGreedy (SG) [13] and Jonker-Volgenant (JV) on the Hamsterer and PPI data. Figure 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.

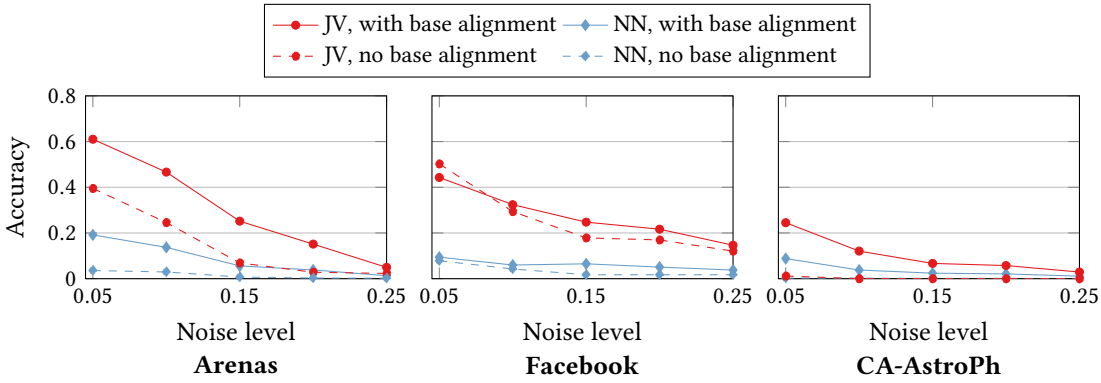


Fig. 10. Accuracy of nearest neighbor and JV matching algorithms with and without base alignment.

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) and (ii) the usage of base alignment (Section 4.5). Figure 10 shows that both the JV linear assignment 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.

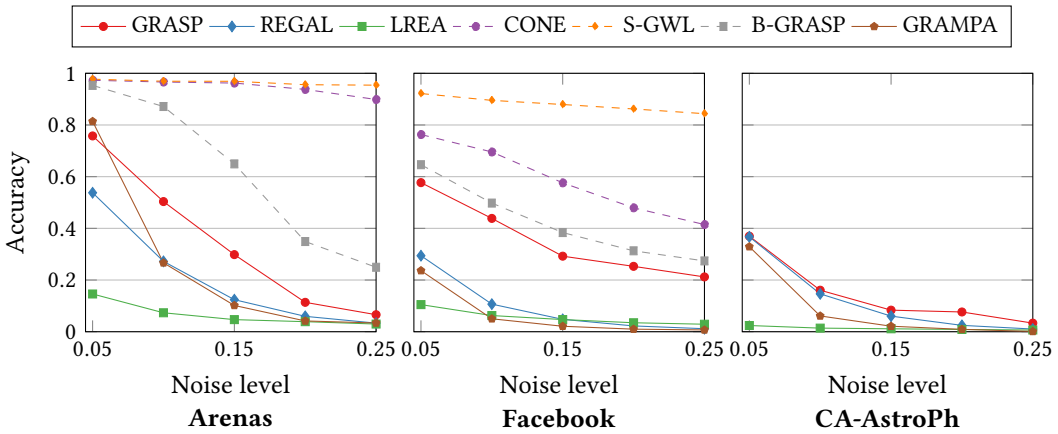


Fig. 11. Accuracy comparison under synthetic noise.

6.3 Comparison under synthetic noise

We compare GRASP and B-GRASP to three scalable methods, namely REGAL [20], GRAMPA [14] and LREA [41], as well as two methods we have found to be non-scalable, namely CONE [7] and S-GWL [61], on real-world data with synthetically generated noise. Figure 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.

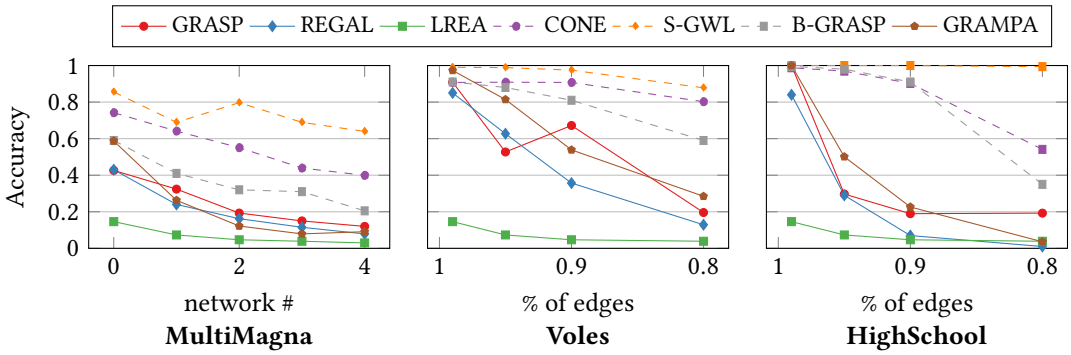


Fig. 12. Accuracy comparison under real noise.

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 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 (cf. Figure 14 in Section 6.6 regarding scalability), which we observed with synthetic noise, transfers to real-world alignment problems with moderate noise.

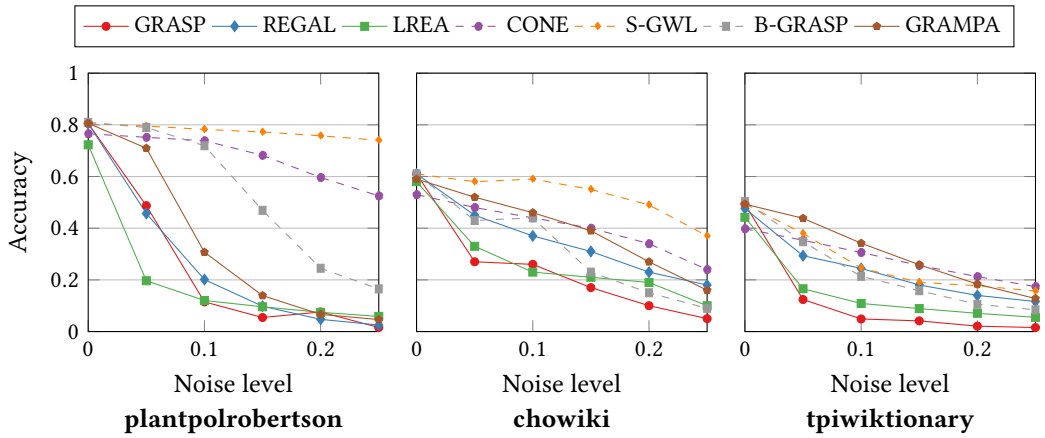


Fig. 13. Accuracy comparison on bipartite graphs.

6.5 Bipartite Graphs

We now experiment with the three bipartite graphs; Figure 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.

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 [42] with a degree distribution following a standard normal distribution. Figure 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.

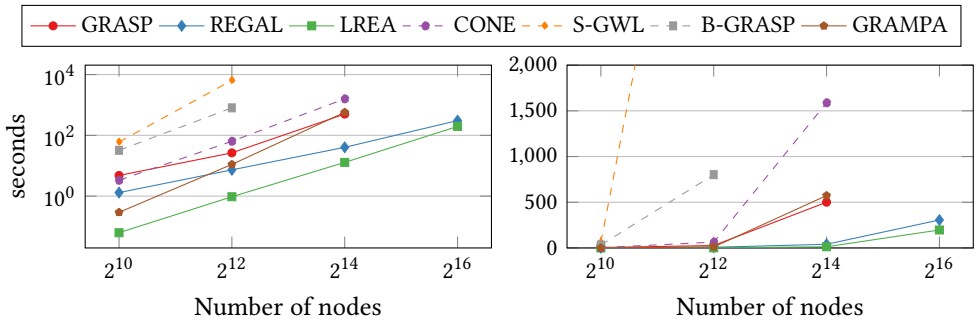


Fig. 14. Runtime vs. number of nodes on four random graphs of increasing size.

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 15 shows the runtime partitioned across the three 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.

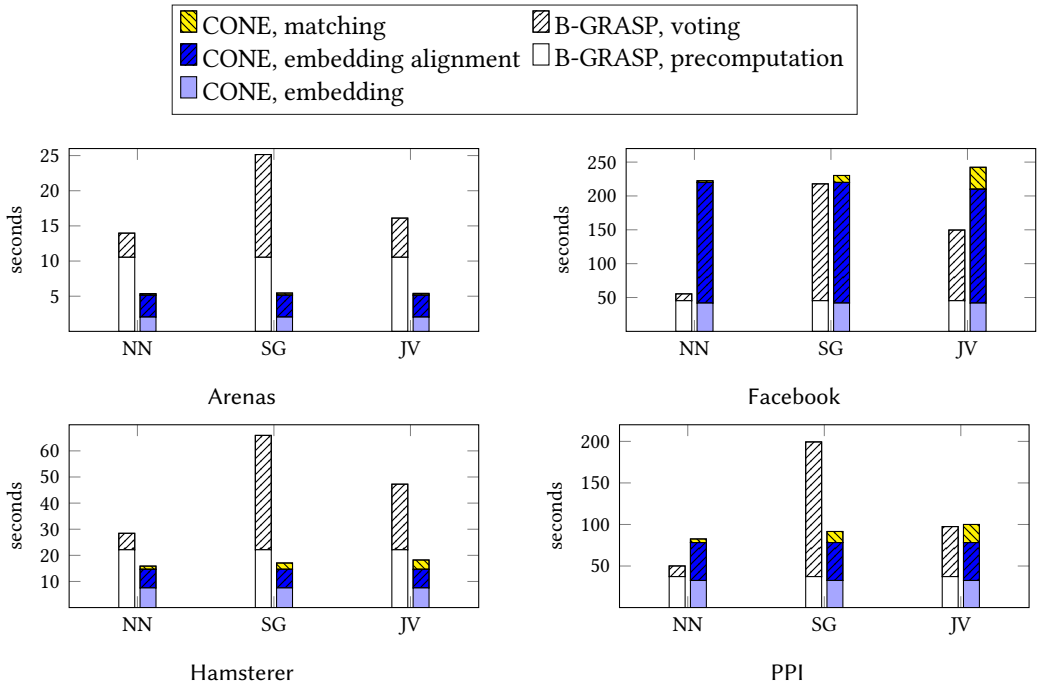


Fig. 15. Time for B-GRASP and CONE with NetMF on four datasets.

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 4.6) can be performed offline; REGAL also allows for precomputation of representations. Figure 16 shows the time to compute alignments after precomputation. Remarkably, GRASP outperforms both REGAL and LREA in the largest CA-AstroPh data. Figure 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.

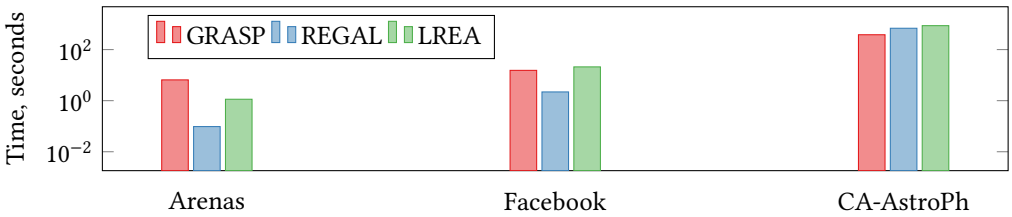


Fig. 16. Alignment time *excluding* precomputation time.

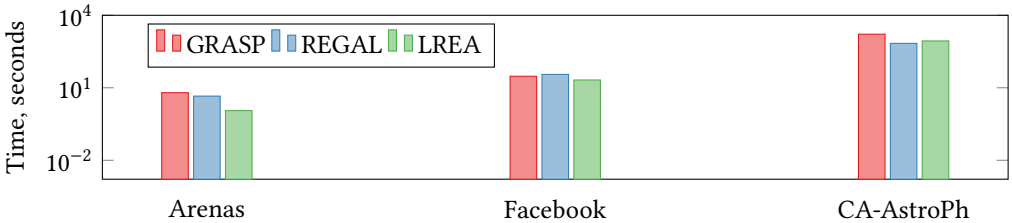


Fig. 17. Alignment time *including* precomputation time.

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 [43]. Table 3 lists the results.

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

Table 3. 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 γ 1.71, 2.34 and 3.35. Figure 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.

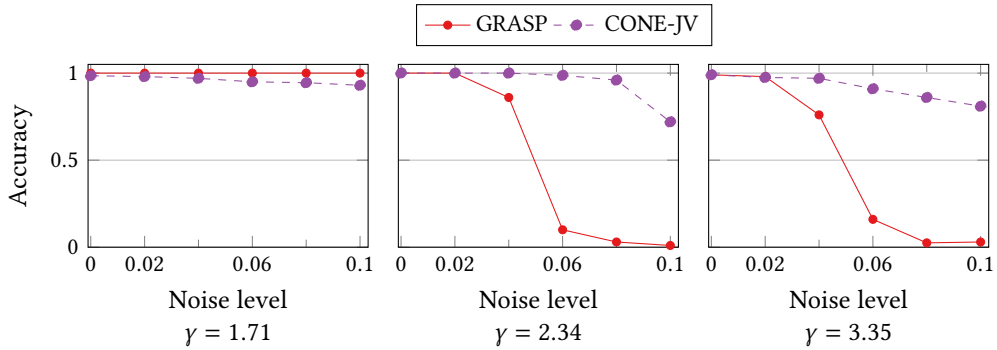


Fig. 18. Accuracy of GRASP with PPR and JV vs. CONE with JV assignment on generated power law graphs.

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, to the alignment of hierarchical graph summaries [24, 51], and towards flexible definitions of subgraph isomorphism, including the case of matching graphs with unequal numbers of nodes.

REFERENCES

[1] Mohammad Abdulkader Abdulrahim. 1998. *Parallel Algorithms for Labeled Graph Matching*. Ph. D. Dissertation. Colorado School of Mines, USA.

[2] Pierre-Antoine Absil, Christopher G. Baker, and Kyle A. Gallivan. 2007. Trust-Region Methods on Riemannian Manifolds. *Found. Comput. Math.* 7, 3 (2007), 303–330.

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

[4] Mohsen Bayati, David F. Gleich, Amin Saberi, and Ying Wang. 2013. Message-Passing Algorithms for Sparse Network Alignment. *TKDD* 7, 1 (2013), 3:1–3:31.

[5] Mikhail Belkin and Partha Niyogi. 2006. Convergence of Laplacian Eigenmaps. In *NeurIPS*. 129–136.

[6] Paul J. Besl and Neil D. McKay. 1992. A Method for Registration of 3-D Shapes. *IEEE Trans. Pattern Anal. Mach. Intell.* 14, 2, 239–256.

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

[8] Xiaokai Chu, Xinxin Fan, Di Yao, Zhihua Zhu, Jianhui Huang, and Jingping Bi. 2019. Cross-Network Embedding for Multi-Network Alignment. In *The Web Conference*. 273–284.

[9] Fan R. K. Chung. 1997. *Spectral graph theory*. Vol. 92. American Mathematical Soc.

[10] David Cohen-Steiner, Weihao Kong, Christian Sohler, and Gregory Valiant. 2018. Approximating the Spectrum of a Graph. In *KDD*. 1263–1271.

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

[12] Tyler Derr, Hamid Karimi, Xiaorui Liu, Jiejun Xu, and Jiliang Tang. 2021. Deep Adversarial Network Alignment. In *CIKM*. 352–361.

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

[14] Zhou Fan, Cheng Mao, Yihong Wu, and Jiaming Xu. 2020. Spectral Graph Matching and Regularized Quadratic Relaxations: Algorithm and Theory. In *ICML (Proceedings of Machine Learning Research, Vol. 119)*. 2985–2995.

[15] Soheil Feizi, Gerald T. Quon, Mariana Recamonde Mendoza, Muriel Médard, Manolis Kellis, and Ali Jadbabaie. 2020. Spectral Alignment of Graphs. *IEEE Trans. Netw. Sci. Eng.* 7, 3 (2020), 1182–1197.

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

[17] Sylvestre Gallot, Dominique Hulin, and Jacques Lafontaine. 1990. *Riemannian geometry*. Springer.

[18] Ji Gao, Xiao Huang, and Jundong Li. 2021. Unsupervised Graph Alignment with Wasserstein Distance Discriminator. In *KDD*. 426–435.

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

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

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

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

[23] Paris A. Karakasis, Aritra Konar, and Nicholas D. Sidiropoulos. 2021. Joint Graph Embedding and Alignment with Spectral Pivot. In *KDD*. 851–859.

[24] Panagiotis Karras and Nikos Mamoulis. 2008. Hierarchical synopses with optimal error guarantees. *ACM Trans. Database Syst.* 33, 3 (2008), 18:1–18:53.

[25] Ehsan Kazemi, Seyed Hamed Hassani, and Matthias Grossglauser. 2015. Growing a Graph Matching from a Handful of Seeds. *Proc. VLDB Endow.* 8, 10 (2015), 1010–1021.

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

[27] David Knossow, Avinash Sharma, Diana Mateus, and Radu Horaud. 2009. Inexact Matching of Large and Sparse Graphs Using Laplacian Eigenvectors. In *Graph-Based Representations in Pattern Recognition, 7th IAPR-TC-15 International Workshop, GbRPR 2009 (Lecture Notes in Computer Science, Vol. 5534)*. Springer, 144–153.

[28] Johannes Kobler, Uwe Schöning, and Jacobo Torán. 2012. *The graph isomorphism problem: its structural complexity*. Springer Science & Business Media.

[29] Giorgos Kollias, Madan Sathe, Shahin Mohammadi, and Ananth Grama. 2013. A fast approach to global alignment of protein-protein interaction networks. *BMC Research Notes* 6, 1 (2013), 35.

[30] Ioannis Koutis, Alex Levin, and Richard Peng. 2016. Faster Spectral Sparsification and Numerical Algorithms for SDD Matrices. *ACM Trans. Algorithms* 12, 2 (2016), 17:1–17:16.

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

[32] Artiom Kovnatsky, Michael M. Bronstein, Alexander M. Bronstein, Klaus Glashoff, and Ron Kimmel. 2013. Coupled quasi-harmonic bases. *Comput. Graph. Forum* 32, 2 (2013), 439–448.

[33] Jérôme Kunegis. 2013. KONECT – The Koblenz Network Collection. In *WWW Companion*. 1343–1350.

[34] Alexander Frederiksen Kyster, Simon Daugaard Nielsen, Judith Hermanns, Davide Mottin, and Panagiotis Karras. 2021. Boosting Graph Alignment Algorithms. In *CIKM*. 3166–3170.

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

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

[37] Or Litany, Emanuele Rodolà, Alexander M. Bronstein, and Michael M. Bronstein. 2017. Fully Spectral Partial Shape Matching. *Comput. Graph. Forum* 36, 2 (2017), 247–258.

[38] Li Liu, William K Cheung, Xin Li, and Lejian Liao. 2016. Aligning Users across Social Networks Using Network Embedding.. In *IJCAI*. 1774–1780.
[39] Eric Malmi, Aristides Gionis, and Evimaria Terzi. 2017. Active network alignment: a matching-based approach. In *CIKM*. 1687–1696.
[40] Tong Man, Huawei Shen, Shenghua Liu, Xiaolong Jin, and Xueqi Cheng. 2016. Predict Anchor Links across Social Networks via an Embedding Approach. In *IJCAI*. IJCAI/AAAI Press, 1823–1829.
[41] Huda Nassar, Nate Veldt, Shahin Mohammadi, Ananth Grama, and David F. Gleich. 2018. Low Rank Spectral Network Alignment. In *WWW*. 619–628.
[42] Mark E. J. Newman. 2003. The Structure and Function of Complex Networks. *SIAM Rev.* 45, 2 (2003), 167–256.
[43] Mark E. J. Newman. 2005. Power Laws, Pareto Distributions and Zipf's Law. *Contemporary physics* 46, 5 (2005), 323–351.
[44] Sadegh Nobari, Panagiotis Karras, HweeHwa Pang, and Stéphane Bressan. 2014. L-opacity: Linkage-Aware Graph Anonymization. In *EDBT*. 583–594.
[45] Maks Ovsjanikov, Mirela Ben-Chen, Justin Solomon, Adrian Butscher, and Leonidas J. Guibas. 2012. Functional maps: a flexible representation of maps between shapes. *ACM Trans. Graph.* 31, 4 (2012), 30:1–30:11.
[46] Lawrence Page, Sergey Brin, Rajeev Motwani, and Terry Winograd. 1999. *The PageRank citation ranking: Bringing order to the web.* Technical Report. Stanford InfoLab.
[47] Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. 2014. Deepwalk: Online learning of social representations. In *KDD*. 701–710.
[48] Jiezhong Qiu, Yuxiao Dong, Hao Ma, Jian Li, Kuansan Wang, and Jie Tang. 2018. Network embedding as matrix factorization: Unifying deepwalk, line, pte, and node2vec. In *WSDM*. 459–467.
[49] Leonardo F. R. Ribeiro, Pedro H. P. Saverese, and Daniel R. Figueiredo. 2017. *struc2vec*: Learning Node Representations from Structural Identity. In *KDD*. 385–394.
[50] Charles Robertson. 1928. *Flowers and insects: lists of visitors to four hundred and fifty-three flowers.* Carlinville, Ill, n.p,.
[51] Tara Safavi, Caleb Belth, Lukas Faber, Davide Mottin, Emmanuel Müller, and Danai Koutra. 2019. Personalized Knowledge Graph Summarization: From the Cloud to Your Pocket. In *ICDM*. 528–537.
[52] Peter H Schönemann. 1966. A generalized solution of the orthogonal procrustes problem. *Psychometrika* 31, 1 (1966), 1–10.
[53] Jianbo Shi and Jitendra Malik. 2000. Normalized Cuts and Image Segmentation. *IEEE Trans. Pattern Anal. Mach. Intell.* 22, 8 (2000), 888–905.
[54] Rohit Singh, Jinbo Xu, and Bonnie Berger. 2008. Global alignment of multiple protein interaction networks with application to functional orthology detection. *Proc. Natl. Acad. Sci. USA* 105, 35 (2008), 12763–12768.
[55] Anton Tsitsulin, Davide Mottin, Panagiotis Karras, Alexander M. Bronstein, and Emmanuel Müller. 2018. NetLSD: Hearing the Shape of a Graph. In *KDD*. 2347–2356.
[56] Anton Tsitsulin, Davide Mottin, Panagiotis Karras, and Emmanuel Müller. 2018. VERSE: Versatile graph embeddings from similarity measures. In *The Web Conference*. 539–548.
[57] Anton Tsitsulin, Marina Munkhoeva, Davide Mottin, Panagiotis Karras, Ivan V. Oseledets, and Emmanuel Müller. 2021. FREDE: Anytime Graph Embeddings. *Proc. VLDB Endow.* 14, 6 (2021), 1102–1110.
[58] Shinj Umeyama. 1988. An eigendecomposition approach to weighted graph matching problems. *IEEE Trans. Pattern Anal. Mach. Intell.* 10, 5 (1988), 695–703.
[59] Vipin Vijayan and Tijana Milenković. 2018. Multiple Network Alignment via MultiMAGNA++. *IEEE ACM Trans. Comput. Biol. Bioinform.* 15, 5 (2018), 1669–1682.
[60] Hao Xiong, Junchi Yan, and Li Pan. 2021. Contrastive Multi-View Multiplex Network Embedding with Applications to Robust Network Alignment. In *KDD*. 1913–1923.
[61] Hongteng Xu, Dixin Luo, and Lawrence Carin. 2019. Scalable Gromov-Wasserstein Learning for Graph Partitioning and Matching. In *NeurIPS*. 3046–3056.
[62] Hongteng Xu, Dixin Luo, Hongyuan Zha, and Lawrence Carin. 2019. Gromov-Wasserstein Learning for Graph Matching and Node Embedding. In *ICML (Proceedings of Machine Learning Research, Vol. 97)*. 6932–6941.
[63] Mingqiang Xue, Panagiotis Karras, Chedy Raïssi, Panos Kalnis, and Hung Keng Pung. 2012. Delineating social network data anonymization via random edge perturbation. In *CIKM*. 475–484.
[64] Yuchen Yan, Si Zhang, and Hanghang Tong. 2021. BRIGHT: A Bridging Algorithm for Network Alignment. In *The Web Conference*. 3907–3917.
[65] Lyudmila Yartseva and Matthias Grossglauser. 2013. On the performance of percolation graph matching. In *ACM Conference on Online Social Networks*. 119–130.
[66] Abdurrahman Yasar and Ümit V Çatalyürek. 2018. An iterative global structure-assisted labeled network aligner. In *KDD*. 2614–2623.
[67] Si Zhang and Hanghang Tong. 2016. FINAL: Fast Attributed Network Alignment. In *KDD*. 1345–1354.

[68] Si Zhang, Hanghang Tong, Long Jin, Yinglong Xia, and Yunsong Guo. 2021. Balancing Consistency and Disparity in Network Alignment. In *KDD*. 2212–2222.

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

[70] Qinghai Zhou, Liangyue Li, Xintao Wu, Nan Cao, Lei Ying, and Hanghang Tong. 2021. Attent: Active Attributed Network Alignment. In *The Web Conference*. 3896–3906.

[71] Yang Zhou, Zeru Zhang, Sixing Wu, Victor S. Sheng, Xiaoying Han, Zijie Zhang, and Ruoming Jin. 2021. Robust Network Alignment via Attack Signal Scaling and Adversarial Perturbation Elimination. In *The Web Conference*. 3884–3895.