# Graph Matching via the lens of Supermodularity

Aritra Konar, *Member, IEEE,* and Nicholas D. Sidiropoulos, *Fellow, IEEE*.

**Abstract**—Graph matching, the problem of aligning a pair of graphs so as to minimize their edge disagreements, has received widespread attention owing to its broad spectrum of applications in data science. As the problem is NP–hard in the worst-case, a variety of approximation algorithms have been proposed for obtaining high quality, suboptimal solutions. In this paper, we approach the task of designing an efficient polynomial-time approximation algorithm for graph matching from a previously unconsidered perspective. Our key result is that graph matching can be formulated as maximizing a monotone, supermodular set function subject to matroid intersection constraints. We leverage this fact to apply a discrete optimization variant of the minorization-maximization algorithm which exploits supermodularity of the objective function to iteratively construct and maximize a sequence of global lower bounds on the objective. At each step, we solve a maximum weight matching problem in a bipartite graph. Differing from prior approaches, the algorithm exploits the combinatorial structure inherent in the problem to generate a sequence of iterates featuring monotonically non-decreasing objective value while always adhering to the combinatorial matching constraints. Experiments on real-world data demonstrate the empirical effectiveness of the algorithm relative to the prevailing state-of-the-art.

**Index Terms**—Graph matching, discrete optimization, supermodularity, matroid intersection, minorization-maximization, weighted bipartite matching

## 1 Introduction

Aligning a pair of graphs constitutes a fundamental problem in combinatorial optimization where the goal is to compute an injective mapping between the vertex sets of the graphs that minimizes edge disagreements. Applications of the problem abound – in computer vision, graph matching has been applied for object matching [1], shape retrieval [2], and action recognition [3]. In bioinformatics, graph matching is used in studying protein-protein interactions across different biological species for inferring protein functionality [4]–[6]. Meanwhile, in data mining, graph matching has been used in ontology alignment for determining the similarities amongst different representations of a database [7], and in network de-anonymization for identifying anonymous individuals by aligning the topology of different social networks [8].

Despite its myriad applications, a longstanding challenge lies in the fact that the problem is notoriously difficult to solve. The classical graph isomorphism problem, which is a special case of graph matching, is presently known to admit an exact solution in quasi-polynomial-time [9] (i.e., in time $2^{O((\log n)^c)}$, where $n$ is the number of vertices and $c > 0$ is a constant). However, the approach is not computationally attractive for even moderate problem instances. On the other hand, the general case of graph matching is equivalent to the NP–hard quadratic assignment problem (QAP) [10], which is also NP–hard to approximate within a factor of $2^{(\log n)^{1-\epsilon}}$ of the optimal solution [11] (for some small $\epsilon > 0$).

Notwithstanding such pessimistic theoretical results, owing to the practical importance of the problem, several approximation methods have been devised for graph matching with the goal of computing high-quality, albeit suboptimal solutions. These include linearization methods [12], which cast the QAP as a mixed integer linear program (MILP). The downside of such a reformulation is that it introduces a large number of new variables and linear constraints, which poses a practical challenge in solving the resulting MILP efficiently. Another broad class of approximation methods apply various relaxations of the combinatorial constraints of the QAP, resulting in convex and non-convex quadratic programming relaxations [13], [14], Lagrangian-based relaxation [15], and semidefinite relaxation [16]. In general, these methods are computationally intensive and the obtained solution is not guaranteed to be feasible for the original QAP. Consequently, a post-processing discretization step often has to be performed on the result in order to obtain a feasible solution, which may incur additional complexity. These considerations limit the application of such methods to small problem instances.

More scalable approximation methods for graph matching include message passing [17], and spectral methods [4]–[6], [18]. While such methods are attractive for their low complexity, this often comes that the expense of not being theoretically well principled – message passing is prone to suffer from oscillations, whereas several spectral methods resort to dropping the combinatorial constraints of the QAP entirely and compute a matching based on the spectral content of the graph adjacency matrices. A notable exception amongst such methods is the spectral alignment algorithm of [18], which possesses certain optimality guarantees for matching random Erdos-Renyi graphs. However, real world graphs are not known to obey the Erdos-Renyi model, and it is not clear whether these performance guarantees extend beyond this synthetic graph model.

While the problem has been considered from various aspects, very few of the aforementioned approaches aim to directly tackle its combinatorial form. Indeed, a salient feature of almost all methods is that they involve a form of relaxation of the matching constraints; the prevailing wisdom being that their combinatorial nature is the key

• *A. Konar and N. D. Sidiropoulos are with the Department of Electrical and Computer Engineering, University of Virginia, Charlottesville, VA, 22904. Partially supported by NSF IIS-1908070.*
*E-mail: (aritra,nikos)@virginia.edu*

component which makes solving the QAP a difficult proposition.

Is there an alternative approach that offers a principled way of efficiently exploiting the combinatorial structure inherent in the problem? In this paper, we provide an affirmative answer to the above question. Our key result, which we formally establish, is that graph matching can be equivalently formulated as maximizing a *supermodular* function subject to the intersection of a pair of matroids. Supermodular functions constitute a special class of discrete functions that are particularly notable for featuring an improving returns property, while also exhibiting several other interesting properties analogous to both convex and concave functions [24]. While problems involving such functions are known to arise in many areas of machine learning (see [24] and references therein), to the best of our knowledge, this marks the first time that such a property has been established for graph matching.

It is worth noting that our result does not make the problem any easier to solve, but it does lead to an alternative approximation approach that iteratively approximates the quadratic objective function in the combinatorial domain while preserving the constraints. This is done by leveraging the particular "convex" property that every supermodular function possesses a (discrete) subgradient [20], which enables us to obtain a global modular lower bound on the objective function about any given correspondence mapping. The attribute is further exploited within a discrete optimization variant of the minorization-maximization (MM) algorithm proposed in [20], which iteratively maximizes a sequence of global modular lower bounds on the quadratic objective function subject to the matroid intersection constraints. At every step of the algorithm, we are required to solve a maximum-weight bipartite matching problem, which can be optimally solved in polynomial-time.

To summarize, the algorithm exploits the supermodular form of the objective function of the QAP to perform iterative maximization by solving a series of weighted bipartite matching subproblems. The sequence of iterates generated exhibit monotonically non-decreasing objective while maintaining feasibility. We also demonstrate ways of reducing the computational footprint of the algorithm by obviating the need to compute expensive Kronecker products of the graph adjacency matrices and allowing the matching subproblems to be solved approximately. Experiments on real-world graphs are provided to demonstrate the favorable performance of the proposed approach relative to the prevailing state-of-the-art.

The conference version of this work appeared at *ICDM 2019* [21] as a short paper. Relative to [21], the present journal version adds the proofs of all technical results, additional analytical and algorithmic details, and more detailed experiments.

## 2 Overview of Supermodularity and Matroids

Given a collection of $n$ items labeled $\mathcal{V} = [n] := \{1, \cdots, n\}$, a *set function* $f : 2^{\mathcal{V}} \to \mathbb{R}$ assigns a real value to any subset $\mathcal{S} \subseteq \mathcal{V}$. Notable amongst set functions is the class of supermodular functions [22]–[24], which are defined as follows:

**Definition 1. [Supermodularity]** A set function $f(.)$ is said to be *supermodular* if and only if for all subsets $\mathcal{A}, \mathcal{B} \subseteq \mathcal{V}$, it holds that

$$f(\mathcal{A}) + f(\mathcal{B}) \leq f(\mathcal{A} \cup \mathcal{B}) + f(\mathcal{A} \cap \mathcal{B}) \tag{1}$$

The above definition can be equivalently, and more conveniently, restated as:

**Definition 2.** For all $\mathcal{A} \subseteq \mathcal{B} \subseteq \mathcal{V} \setminus \{v\}$, it holds that

$$f(\mathcal{A} \cup \{v\}) - f(\mathcal{A}) \leq f(\mathcal{B} \cup \{v\}) - f(\mathcal{B}) \tag{2}$$

That is, for such functions, given subsets $\mathcal{A} \subseteq \mathcal{B} \subseteq \mathcal{V} \setminus \{v\}$, the marginal improvement obtained by adding an element $v$ to the smaller set $\mathcal{A}$ is never exceeds that obtained by adding $v$ to the superset $\mathcal{B}$. Simply stated, (2) asserts that supermodular functions, exhibit an improving returns property, i.e., it always helps to "collect" more items.

**Definition 4. [Modularity]** A set function $g(.)$ is said to be *modular* if and only if there exists a vector $\mathbf{g} \in \mathbb{R}^n$ for all subsets $\mathcal{S} \subseteq \mathcal{V}$ such that $g(\mathcal{S}) = \mathbf{g}^T \mathbb{1}_{\mathcal{S}} = \sum_{e \in \mathcal{S}} g(e)$. Note that such a function is both submodular and supermodular.

**Definition 5. [Monotonicity]** A set function $f(.)$ is said to be *monotone* if $f(\mathcal{A}) \leq f(\mathcal{B})$ for all $\mathcal{A} \subseteq \mathcal{B} \subseteq \mathcal{V}$. Moreover, if $f(.)$ is also supermodular then monotonicity is equivalent to the condition $f(\{v\}) - f(\emptyset) \geq 0, \forall\, v \in \mathcal{V}$.

**Definition 6. [Matroid]** A *matroid* [25] is an ordered pair $(\mathcal{V}, \mathcal{I})$ consisting of a finite collection of items $\mathcal{V}$ and a family of subsets of $\mathcal{V}$ called independent sets $\mathcal{I}$, which satisfy the following axioms

(A1) $\emptyset \in \mathcal{I}$
(A2) If $\mathcal{B} \in \mathcal{I}$ and $\mathcal{A} \subseteq \mathcal{B}$, then $\mathcal{A} \in \mathcal{I}$
(A3) If $\mathcal{A}, \mathcal{B} \in \mathcal{I}$ and $|\mathcal{A}| < |\mathcal{B}|$, then there exists an element $v \in \mathcal{B} \setminus \mathcal{A}$ such that $\mathcal{A} \cup \{v\} \in \mathcal{I}$

Matroids can be viewed as an abstraction of the notion of linear independence in linear algebra, and are useful for modeling constraints in combinatorial optimization problems. A specific example of interest is a *partition matroid*, where we are given a partition of $\mathcal{V}$ into $m$ pair-wise disjoint subsets $\{\mathcal{G}_i\}_{i=1}^m$ such that $\mathcal{V} = \cup_{i=1}^m \mathcal{G}_i$ with $\mathcal{G}_i \cap \mathcal{G}_j = \emptyset, \forall\, i \neq j, i, j \in [m]$. Then, the collection of subsets $\mathcal{I} = \{\mathcal{A} \subseteq \mathcal{V} : |\mathcal{A} \cap \mathcal{G}_i| \leq k_i, \forall\, i \in [m]\}$ satisfies axioms (A1)-(A3).

## 3 Problem Statement

Consider a simple, weighted, undirected graph $\mathcal{G}_A = (\mathcal{V}_A, \mathcal{E}_A, w_A)$ on $n_A$ vertices with vertex-set $\mathcal{V}_A = [n_A] := \{1, \cdots, n_A\}$, edge-set $\mathcal{E}_A \subset [n_A] \times [n_A]$ and a non-negative weight function $w_A : \mathcal{E}_A \to \mathbb{R}_+$ defined on the edges. Similarly, we define $\mathcal{G}_B = (\mathcal{V}_B, \mathcal{E}_B, w_B)$ to be a weighted, undirected graph on $n_B$ vertices. Without loss of generality, we assume that $n_A \leq n_B$. Let $\mathbf{A} \in \mathbb{R}^{n_A \times n_A}$ and $\mathbf{B} \in \mathbb{R}^{n_B \times n_B}$ denote the (symmetric) weighted adjacency matrices of $\mathcal{G}_A$ and $\mathcal{G}_B$ respectively. The objective of the graph matching problem is to compute an injective mapping from the vertex-set $\mathcal{V}_A$ to $\mathcal{V}_B$ which minimizes the number of (weighted) edge disagreements between graphs $\mathcal{G}_A$ and $\mathcal{G}_B$. Formally, the problem can be stated as follows: given $\mathbf{A}, \mathbf{B}$, we seek to find an assignment matrix $\mathbf{P} \in \{0,1\}^{n_B \times n_A}$ with entries

$$P(j,i) = \begin{cases} 1, \text{if vertex } i \in \mathcal{V}_A \text{ is assigned to vertex } j \in \mathcal{V}_B \\ 0, \text{otherwise} \end{cases} \tag{3}$$

that solves the optimization problem

$$\min_{\mathbf{P} \in \mathcal{P}}. \|\mathbf{B} - \mathbf{P}\mathbf{A}\mathbf{P}^T\|_F^2 \tag{4}$$

where the feasible set $\mathcal{P}$ of injective mappings is compactly expressed as

$$\mathcal{P} := \{\mathbf{P} \in \{0,1\}^{n_B \times n_A} \mid \mathbf{P}^T\mathbf{P} = \mathbf{I}\} \tag{5}$$

for $n_A < n_B$. This corresponds to the subgraph matching problem, where we seek to select a subgraph of $\mathcal{G}_B$ which has minimal edge disagreements with $\mathcal{G}_A$. On the other hand, when $n_A = n_B$, the feasible set is described by the set of all $n_A \times n_A$ permutation matrices

$$\mathcal{P} := \{\mathbf{P} \in \{0,1\}^{n_A \times n_A} \mid \mathbf{P}^T\mathbf{1} = \mathbf{1}, \mathbf{P}\mathbf{1} = \mathbf{1}\}. \tag{6}$$

In this case, the vertex mapping induced by $\mathbf{P}$ is bijective, and (4) corresponds to the classical graph matching problem. With some abuse of notation, we use $\mathcal{P}$ to denote the feasible set for the general case of graph matching (with $n_A \leq n_B$).

For our purposes, it will be convenient to reformulate problem (4) in maximization form. Towards this end, note that the cost function of (4) can be expressed as

$$\begin{aligned} \|\mathbf{B} - \mathbf{P}\mathbf{A}\mathbf{P}^T\|_F^2 &= \|\mathbf{B}\|_F^2 + \|\mathbf{P}\mathbf{A}\mathbf{P}^T\|_F^2 - 2\text{Trace}(\mathbf{B}\mathbf{P}\mathbf{A}\mathbf{P}^T) \\ &= \|\mathbf{B}\|_F^2 + \|\mathbf{A}\|_F^2 - 2\text{Trace}(\mathbf{B}\mathbf{P}\mathbf{A}\mathbf{P}^T) \end{aligned} \tag{7}$$

where the second equality follows from the fact that $\mathbf{P}$ is always column-orthonormal. Hence, the first two terms in the above summand can be dropped to yield the problem

$$\max_{\mathbf{P} \in \mathcal{P}}. \text{Trace}(\mathbf{B}\mathbf{P}\mathbf{A}\mathbf{P}^T) \tag{8}$$

Utilizing the cyclic property of the trace operator, the linearization property $\text{vec}(\mathbf{A}\mathbf{X}\mathbf{B}) = (\mathbf{B}^T \otimes \mathbf{A})\text{vec}(\mathbf{X})$ of the vec(.) operator, and the fact that $\mathbf{A}$ is symmetric, the objective function of (8) can be further expressed as

$$\begin{aligned} \text{Trace}(\mathbf{B}\mathbf{P}\mathbf{A}\mathbf{P}^T) &= \text{Trace}(\mathbf{P}^T\mathbf{B}\mathbf{P}\mathbf{A}) \\ &= \text{vec}(\mathbf{P})^T\text{vec}(\mathbf{B}\mathbf{P}\mathbf{A}) \\ &= \text{vec}(\mathbf{P})^T(\mathbf{A} \otimes \mathbf{B})\text{vec}(\mathbf{P}) \end{aligned} \tag{9}$$

where the symbol $\otimes$ denotes the Kronecker product. Defining the $n := n_A n_B$ dimensional vector $\mathbf{x} = \text{vec}(\mathbf{P})$ and the $n \times n$ matrix $\mathbf{H} = \mathbf{A} \otimes \mathbf{B}$, we can equivalently express the objective function of (8) as a quadratic function $\mathbf{x}^T\mathbf{H}\mathbf{x}$.

In order to express the feasible set $\mathcal{P}$ in terms of the vector $\mathbf{x}$, we proceed as follows: first, we define a complete bipartite graph $\mathcal{G}_C = (\mathcal{V}_C, \mathcal{E}_C)$ between the vertices of $\mathcal{G}_A$ and $\mathcal{G}_B$, with vertex-set $\mathcal{V}_C := \mathcal{V}_A \cup \mathcal{V}_B$ (on $n_A + n_B$ vertices) and edge-set $\mathcal{E}_C := [n_A] \times [n_B]$. Using this construction, any injective mapping $\mathbf{P} \in \mathcal{P}$ from $\mathcal{V}_A$ to $\mathcal{V}_B$ can be equivalently viewed as a subset of edges of $\mathcal{E}_C$ which form a matching of maximum cardinality in the complete bipartite graph $\mathcal{G}_C$, i.e., the vector $\mathbf{x} \in \{0,1\}^n$ corresponds to the index vector of a maximum-cardinality matching in $\mathcal{G}_C$. Note that when $n_A < n_B$, the matching is not perfect (it is not possible to "cover" all vertices in $\mathcal{V}_B$). Meanwhile, for $n_A = n_B$, we obtain a perfect matching, which also corresponds to a permutation mapping (since $\mathcal{G}_C$ is complete). Formally, the set of all index vectors of matchings in $\mathcal{G}_C$ can be represented by the set

$$\begin{aligned} \mathcal{M} := \Big\{ \mathbf{x} \in \{0,1\}^n \mid &\sum_{j:(i,j) \in \mathcal{E}_C} x_{ij} \leq 1, \\ &\sum_{i:(i,j) \in \mathcal{E}_C} x_{ij} \leq 1, \forall\, (i,j) \in \mathcal{E}_C \Big\} \end{aligned} \tag{10}$$

Hence, we can equivalently express problem (8) in vector form as

$$\max_{\mathbf{x} \in \mathcal{M}}. \mathbf{x}^T\mathbf{H}\mathbf{x} \tag{11}$$

which corresponds to computing the maximum-cardinality matching in $\mathcal{G}_C$ that maximizes the number of weighted edge overlaps between $\mathcal{G}_A$ and $\mathcal{G}_B$.

In its general form, graph matching is equivalent to the quadratic assignment problem, which is known to be NP–hard in the worst-case. Roughly speaking, this implies that there is no polynomial-time algorithm which can optimally solve every problem instance, unless P=NP. Consequently, a considerable amount of effort has been invested in designing theoretically efficient and practically effective polynomial-time approximation algorithms. In the forthcoming sections, we outline our proposed approach.

## 4 A New Perspective on Graph Matching

We begin by equivalently reformulating the discrete optimization problem (11) as a subset selection problem. Since each vector $\mathbf{x} \in \mathcal{M}$ is the index vector of a matching set in $\mathcal{G}_C$, it can be equivalently expressed as $\mathbf{x} = \mathbb{1}_{\mathcal{S}}$, i.e., as the indicator vector of a subset of edges $\mathcal{S} \subset \mathcal{E}_C$ which corresponds to a matching in $\mathcal{G}_C$. In order to describe the matching set $\mathcal{M}$ in terms of set notation, we first define the following families of subsets of the edges $\mathcal{E}_C$

$$\mathcal{I}_A = \{\mathcal{S} \subset \mathcal{E}_C, |\mathcal{S} \cap \delta(i)| \leq 1, \forall\, i \in \mathcal{V}_A\}, \tag{12a}$$

$$\mathcal{I}_B = \{\mathcal{S} \subset \mathcal{E}_C, |\mathcal{S} \cap \delta(j)| \leq 1, \forall\, j \in \mathcal{V}_B\} \tag{12b}$$

where $\delta(i) := \{i\} \times [n_B]$ denotes the set of edges incident to vertex $j \in \mathcal{V}_A$, and $\delta(j) := [n_A] \times \{j\}$ denotes the set of edges incident to vertex $j \in \mathcal{V}_B$. The set $\mathcal{I}_A$ enforces the condition that each vertex $i \in \mathcal{V}_A$ has at most one edge incident to a vertex $j \in \mathcal{V}_B$, while $\mathcal{I}_B$ similarly ensures that every vertex $j \in \mathcal{V}_B$ has at most one edge incident to a vertex $i \in \mathcal{V}_A$. Note that any set $\mathcal{S} \in \mathcal{M}$ must satisfy both (12a) and (12b) in order to be a valid matching in $\mathcal{G}_C$. Hence, we can express $\mathcal{M}$ as the intersection of the families of subsets $\mathcal{I}_A$ and $\mathcal{I}_B$, i.e., as $\mathcal{M} = \mathcal{I}_A \cap \mathcal{I}_B$. Furthermore, by construction, the sets $\{\delta(i)\}_{i=1}^{n_A}$ and $\{\delta(j)\}_{j=1}^{n_B}$ in (12a) and (12b) are pairwise disjoint and correspond to a partition of $\mathcal{E}_C$, i.e., $\mathcal{E}_C = \cup_{i=1}^{n_A} \delta(i) = \cup_{j=1}^{n_B} \delta(j)$. From this description, it is clear that both $\mathcal{I}_A$ and $\mathcal{I}_B$ can be viewed as the respective independent sets of a pair of partition matroids $(\mathcal{E}_C, \mathcal{I}_A)$ and $(\mathcal{E}_C, \mathcal{I}_B)$ defined on the edges of $\mathcal{G}_C$. We conclude that $\mathcal{M}$ corresponds to the intersection of a pair of partition matroids.

In set notation, problem (11) can then be equivalently expressed as maximizing a quadratic function of the indicator vector of a set subject to matroid intersection constraints:

$$\max_{\mathcal{S}\in\mathcal{I}_A\cap\mathcal{I}_B} \left\{ f(\mathcal{S}) := \mathbb{1}_{\mathcal{S}}^T \mathbf{H} \mathbb{1}_{\mathcal{S}} \right\} \tag{13}$$

We now make the following crucial observation regarding the objective function $f(\mathcal{S})$.

**Proposition 1.** *$f(\mathcal{S})$ is a non-negative, monotone, supermodular function.*

*Proof.* In order to show that $f(\mathcal{S})$ is supermodular, note that the matrix $\mathbf{H}$ has non-negative off-diagonal entries by construction, which is both a necessary and sufficient condition for a quadratic function to be supermodular [24, Proposition 6.3]. Furthermore, by choice of $\mathcal{S} = \emptyset$, we have $f(\emptyset) = 0$. The non-negativity of $f(\mathcal{S})$ then follows from the fact that the matrix $\mathbf{H}$ has non-negative entries, and $f(\mathcal{S})$ for $\mathcal{S} \neq \emptyset$ corresponds to summing up the entries of the submatrix $\mathbf{H}[\mathcal{S}, \mathcal{S}] \in \mathbb{R}^{|\mathcal{S}|\times|\mathcal{S}|}$ with rows and columns indexed by $\mathcal{S}$. Finally, since $f(\mathcal{S})$ is supermodular, monotonicity is equivalent to the condition $f(\{e\}) - f(\emptyset) \geq 0, \forall\ e \in \mathcal{E}_C$. In this case, we have $f(\{e\}) - f(\emptyset) = h_{ee} = 0$, since the diagonal entries of $\mathbf{H}$ are zero, from which the result readily follows. □

The implication of the above result is that graph matching corresponds to maximizing a non-negative, monotone supermodular function subject to matroid intersection constraints. To the best of our knowledge, this marks the first time that the problem has been viewed through the lens of supermodular optimization. While this observation does not change the fact that the problem is NP–hard in its general form, it does offer a way to design a polynomial-time approximation algorithm which naturally respects the combinatorial nature of the problem, as we explain in the next section.

## 5 A Successive Approximation Algorithm for Graph Matching

Our approximation approach is centered around exploiting the specific combinatorial structure of (13), i.e., the fact that the objective function is non-negative, monotone, supermodular and that the constraints correspond to the intersection of a pair of matroids. For the purpose of doing so, we utilize a discrete optimization analogue of the minorization-maximization (MM) algorithm proposed in [20], which we now describe.

### 5.1 Overview:

The algorithm starts from an initial matching set $\mathcal{S}_0 \in \mathcal{M}$ and then proceeds in the following iterative fashion:

(1) **Minorization:** At each step $k \in \mathbb{N}$, a modular function $m^{(k)}(\mathcal{S})$ is constructed by approximating the objective function $f(\mathcal{S})$ about the current solution set $\mathcal{S}_k$ in a manner such that the following twin properties are satisfied

$$f(\mathcal{S}) \geq m^{(k)}(\mathcal{S}), \forall\, \mathcal{S} \subseteq \mathcal{E}_C, \text{ and } f(\mathcal{S}_k) = m^{(k)}(\mathcal{S}_k). \tag{14}$$

Hence, the function $m^{(k)}(\mathcal{S})$ constitutes a global modular lower bound of $f(\mathcal{S})$ which is tight about the current solution set $\mathcal{S} = \mathcal{S}_k$.

(2) **Maximization:** Next, the obtained modular lower bound is maximized subject to the matching constraints $\mathcal{M}$. This corresponds to solving a maximum-weight matching problem in a complete bipartite graph, and hence, can be solved optimally in polynomial-time. The solution obtained from this subproblem is then set to be the solution set $\mathcal{S}_{k+1}$ in the next iteration.

Overall, the algorithm successively maximizes a sequence of global lower bounds on the objective function, while preserving the feasibility of the generated iterates $\{\mathcal{S}_k\}_{k\geq 0}$, i.e., we always have $\mathcal{S}_k \in \mathcal{M}, \forall\ k \in \mathbb{N}$. This is an important distinction between our approach and prevailing approximation methods for graph matching, which often resort to relaxing the matching constraint set. Furthermore, steps (1) and (2) also ensure that the iterate sequence $\{\mathcal{S}_k\}_{k\geq 0}$ features monotonically non-decreasing objective value.

### 5.2 Formal Description:

The key ingredient required to carry out each step of the algorithm is the construction of the global modular lower bound. Towards this end, we make use of the following powerful fact [20]: since $f(.)$ is supermodular, for every set $\mathcal{X} \subseteq \mathcal{E}_C$, it possesses a non-empty subdifferential set $\partial f(\mathcal{X})$ (similar to convex functions), which can be formally defined as

$$\partial f(\mathcal{X}) = \{\mathbf{g} \in \mathbb{R}^n \mid f(\mathcal{Y}) \geq f(\mathcal{X}) + g(\mathcal{Y}) - g(\mathcal{X}), \forall\ \mathcal{Y} \subseteq \mathcal{E}_C\} \tag{15}$$

where $g(.) = \mathbf{g}^T \mathbb{1}_{(.)}$ denotes a modular function. The set $\partial f(\mathcal{X})$ is described by the intersection of at most $2^n$ non-redundant linear inequalities in $\mathbf{g}$, and is thus a polyhedron. A subgradient $\mathbf{g} \in \partial f(\mathcal{X})$ can be used to define a modular function of the form

$$m_{\mathcal{X}}(\mathcal{Y}) := f(\mathcal{X}) + g(\mathcal{Y}) - g(\mathcal{X}), \tag{16}$$

which, by construction, satisfies the properties

$$m_{\mathcal{X}}(\mathcal{X}) = f(\mathcal{X}), \tag{17a}$$

$$m_{\mathcal{X}}(\mathcal{Y}) \leq f(\mathcal{Y}), \forall\ \mathcal{Y} \subseteq \mathcal{E}_C \tag{17b}$$

Thus, each subgradient $\mathbf{g} \in \partial f(\mathcal{X})$ defines a tight modular global lower bound of the objective function $f(.)$, which is required for use at each step of the MM algorithm. As shown in [20], the following specific choices of vectors define valid subgradients of $\partial_f(\mathcal{X})$

$$\hat{g}(j) = \begin{cases} f(\mathcal{E}_C) - f(\mathcal{E}_C \setminus \{j\}), & \forall\ j \in \mathcal{X} \\ f(\mathcal{X} \cup \{j\}) - f(\mathcal{X}), & \forall\ j \notin \mathcal{X} \end{cases} \tag{18a}$$

$$\check{g}(j) = \begin{cases} f(\mathcal{X}) - f(\mathcal{X} \setminus \{j\}), & \forall\ j \in \mathcal{X} \\ f(\{j\}) - f(\emptyset), & \forall\ j \notin \mathcal{X} \end{cases} \tag{18b}$$

Since $f(.)$ is monotone, it follows that the subgradients defined above have non-negative entries.

Having formally defined how to construct a global modular lower bound of the objective function $f(\mathcal{S})$ about any set $\mathcal{S} \subseteq \mathcal{E}_C$, we are ready to complete our description of the

MM algorithm. After initialization from an arbitrary matching set $\mathcal{S}_0 \in \mathcal{M}$, at every subsequent iteration $k \in \mathbb{N}$, the objective function $f(\mathcal{S})$ is replaced by its modular approximation about the current iterate $\mathcal{S}_k \in \mathcal{M}$ constructed using a subgradient $\mathbf{g} \in \partial f(\mathcal{S}_k)$. This results in the following optimization subproblem

$$\max_{\mathcal{S}\in\mathcal{I}_A\cap\mathcal{I}_B} \left\{ m^{(k)}(\mathcal{S}) := f(\mathcal{S}_k) + g(\mathcal{S}) - g(\mathcal{S}_k) \right\} \tag{19}$$

which is equivalent to maximizing a global lower bound on $f(\mathcal{S})$. Upon dropping constants, the above problem becomes

$$\max_{\mathcal{S}\in\mathcal{I}_A\cap\mathcal{I}_B} \left\{ g(\mathcal{S}) := \mathbf{g}^T \mathbb{1}_{\mathcal{S}} \right\} \tag{20}$$

which corresponds to solving a maximum-weight matching problem in the complete bipartite graph $\mathcal{G}_C$, and can be solved optimally in polynomial-time. For example, in the case $n_A = n_B$, (20) reduces to a linear assignment problem which can be solved via augmenting-path based methods, e.g., the Hungarian algorithm [26] or the empirically superior Jonker-Volgenant algorithm [27]. Meanwhile, for the subgraph matching case with $n_A < n_B$, recognizing that (20) is a linear programming problem with a particular combinatorial structure, the Network Simplex algorithm [28] can be employed for solving it. The solution of (20) is then set to be the subsequent iterate $\mathcal{S}_{k+1}$.

Overall, the MM algorithm exploits the supermodularity of the objective function of the graph matching problem to perform approximate maximization by iteratively solving a sequence of maximum-weight bipartite matching problems. The iterates $\{\mathcal{S}_k\}_{k\geq 0}$ generated by the algorithm always satisfy the matching constraints $\mathcal{M}$, while the sequence exhibits monotonically non-decreasing objective value. The latter property can be established via the following chain of inequalities

$$f(\mathcal{S}_{k+1}) \geq m^{(k)}(\mathcal{S}_{k+1}) \geq m^{(k)}(\mathcal{S}_k) = f(\mathcal{S}_k) \tag{21}$$

where the first inequality is due to the lower bound property of $m^{(k)}(\mathcal{S}_{k+1})$, the second inequality stems from the optimality of $\mathcal{S}_{k+1}$, while the last equality is due to the tightness of approximation at $\mathcal{S} = \mathcal{S}_k$. As the matching set $\mathcal{M}$ is finite, it follows that the algorithm attains convergence in terms of the objective function. Pseudo-code for the algorithm is provided below.

**Algorithm 1: Minorization Maximization**

```
1 Input: $f(.)$ and an initial matching set $\mathcal{S}_0 \in \mathcal{M}$
2 Output: An approximate solution $\hat{\mathcal{S}}$
3 Initialize: $k \leftarrow 0$
4 repeat
5     Construct subgradient $\mathbf{g} \in \partial f(\mathcal{S}_k)$
6     Perform update $\mathcal{S}_{k+1} \in \arg\max_{\mathcal{S}\in\mathcal{I}_A\cap\mathcal{I}_B} g(\mathcal{S})$
7     $k \leftarrow k+1$
8 until objective function has converged
9 Return: $\hat{\mathcal{S}} \leftarrow \mathcal{S}^{(k)}$
```

By design, the algorithm is parameter-tuning free, requiring only the specification of the initial matching set $\mathcal{S}_0 \in \mathcal{M}$. This could be the output of another algorithm, whose solution we wish to refine further, or a carefully designed initialization for the given problem instance.

## 5.3 Computational Aspects and Interpretation:

The complexity of the MM algorithm is determined by the per-iteration cost of computing a subgradient and then solving a maximum weight bipartite matching problem. Regarding construction of the subgradients, on first glance at (18), it appears that we have to instantiate and store the Kronecker product matrix $\mathbf{H} = \mathbf{A} \otimes \mathbf{B}$, which can prove to be expensive both in terms of computational effort and storage space. Fortunately, this is not the case for both choice of subgradients. Indeed, it can be shown (see Appendix A for the complete derivation) that for a given matching set $\mathcal{S} \in \mathcal{M}$, the subgradient expressions reduce to

$$\hat{g}(j) = \begin{cases} 2\deg_B(\pi(i))\deg_A(i), & \forall\, j \in \mathcal{S} \\ 2\mathbf{b}_{\pi(i)}^T \mathbf{P}\mathbf{a}_i, & \forall\, j \notin \mathcal{S} \end{cases} \tag{22a}$$

$$\check{g}(j) = \begin{cases} 2\mathbf{b}_{\pi(i)}^T \mathbf{P}\mathbf{a}_i, & \forall\, j \in \mathcal{S} \\ 0, & \forall\, j \notin \mathcal{S} \end{cases} \tag{22b}$$

where $\mathbf{P} = \text{unvec}(\mathbb{1}_{\mathcal{S}})$ denotes the matricized matching index vector (i.e., the correspondence matrix), with non-zero entries $P(\pi(i), i) = 1$ if and only if vertex $i \in [n_A]$ is assigned to vertex $\pi(i) \in [n_B]$, or equivalently, edge $j = (i-1)n_A + \pi(i)$ connecting the vertex-pair $(i, \pi(i))$ in $\mathcal{G}_C$ is included in the matching set $\mathcal{S}$. Meanwhile, $P(\bar{\pi}(i), i) = 0$ signifies that vertex $i \in [n_A]$ does not correspond to $\bar{\pi}(i) \in [n_B]$, or that the edge $j = (i-1)n_A + \bar{\pi}(i)$ connecting the vertex-pair $(i, \bar{\pi}(i))$ in $\mathcal{G}_C$ is not included in $\mathcal{S}$. Finally, $\deg_A(i)$ and $\deg_B(\pi(i))$ denote the degrees of vertices $i \in [n_A]$ and $\pi(i) \in [n_B]$ respectively[1], while $\mathbf{a}_i \in \mathbb{R}^{n_A}$ and $\mathbf{b}_{\pi(i)} \in \mathbb{R}^{n_B}$ represent the respective columns of the adjacency matrices $\mathbf{A}$ and $\mathbf{B}$.

The expressions (22) reveal that the subgradients can be computed with only modest effort. For example, $\check{\mathbf{g}}$ is a sparse vector with at most $n_A$ non-zero elements, each of which can be evaluated via an inner product. While $\hat{\mathbf{g}}$ is not guaranteed to be sparse in general, note that forming the entries $\{\hat{g}(j)\}_{j\in\mathcal{S}}$ entails trivial computation, whereas each of the remaining entries $j \notin \mathcal{S}$ can be formed by simply evaluating an inner-product.

More importantly, the expressions (22) also provide insight regarding the nature of our successive approximation algorithm. Let us first consider the case where the subgradient type $\check{\mathbf{g}}$ is used at each step. We seek an interpretation of the entries $\{\check{g}(j)\}_{j\in\mathcal{S}_k}$, as the remaining entries are zeros. Notice that for a given assignment matrix $\mathbf{P} \in \mathcal{P}$, each of the terms $\mathbf{b}_{\pi(i)}^T \mathbf{P}\mathbf{a}_i, \forall\, i \in [n_A]$ is a measure of similarity or overlap between the one-hop neighborhoods of the aligned vertex pair $(i, \pi(i)) \in [n_A] \times [n_B]$. Hence, given the current solution $\mathcal{S}_k$ at iteration $k \in \mathbb{N}$, the non-zero entries $\{g(j)\}_{j\in\mathcal{S}_k}$ of $\mathbf{g}$ reflect the degree of neighborhood overlap induced by $\mathbf{P}_k = \text{unvec}(\mathbb{1}_{\mathcal{S}_k})$. This is intuitively pleasing, as it indicates that aligned vertex pairs with similar neighborhoods are assigned the largest weights in the maximum-

1. For a weighted graph, the degree of a vertex is taken to be the sum of weighted edges incident on that vertex

weight bipartite matching problem, and thus, are likely to be picked in the matching set $\mathcal{S}_{k+1}$ as well.

We now consider the case where the subgradient type $\hat{\mathbf{g}}$ is used at each step. It is evident that the entries of $\{\hat{g}(j)\}_{j\in\mathcal{S}_k}$ which are assigned large weights correspond to those pairs of aligned vertices $(i,\pi(i))\in[n_A]\times[n_B]$ which have similar degrees, with the largest values assigned for those aligned vertex-pairs which have high degree. This is again intuitively pleasing, since a pair of vertices with similar degrees are more likely to share a correspondence compared to a pair with dissimilar degrees. Meanwhile, regarding the entries $\{\hat{g}(j)\}_{j\notin\mathcal{S}_k}$, we can provide the following interpretation: for a pair of non-aligned vertices $(i,\bar{\pi}(i))$, note that $\mathbf{b}_{\bar{\pi}(i)}^T\mathbf{P}_k\mathbf{a}_i$ measures the similarity or overlap between the neighborhoods $\mathbf{a}_i$ and $\mathbf{b}_{\bar{\pi}(i)}$. It stands to reason that for a high quality solution $\mathcal{S}_k$, the neighborhood overlap for any pair of non-aligned vertices should be small [2]. However, if the opposite is true for any vertex pair $(i,\bar{\pi}(i))$ (i.e., the entry $\hat{g}(j)$ is assigned a large weight), then it indicates that $(i,\bar{\pi}(i))$ are likely to be a good match, and suggests that the edge $j=(i-1)n_A+\bar{\pi}(i)$ should be selected in the matching set in the subsequent step.

We now focus on the computational complexity of the maximum-weight bipartite matching problem. For the case of $n_A=n_B$, the worst-case complexity incurred in solving a linear assignment problem is $O(n_A^3)$ [26], [27], whereas for $n_A<n_B$, the Network-Simplex algorithm incurs worst-case complexity of $O(n_An_B(n_A+n_B)\log(n_A+n_B))$ in solving the matching problem [28]. While these are polynomial-time results, the super-cubic run-time of these algorithms (in terms of the problem dimension) has unfavorable implications in terms of their scalability to large problem instances.

To facilitate the scalability of the MM algorithm, instead of exactly solving each bipartite matching problem (20), we can opt for *approximate* maximization. One option is to use a simple greedy algorithm, which has a substantially improved run-time complexity of $O(n_An_B\log(n_An_B))$, and is guaranteed to output a solution which is no worse than $50\%$ of the optimal objective value of (20). For the case of $n_A=n_B$, we advocate using a more sophisticated alternative: Sinkhorn's matrix balancing algorithm [29], which features computationally lightweight updates and can compute a high-quality approximate solution of (20) at low complexity. The details of this technique are relegated to Appendix B.

While resorting to approximate maximization of the bipartite matching subproblems within the MM framework reduces the per-iteration complexity, we point out that the iterates generated in this case are not guaranteed to exhibit monotonically non-decreasing objective value as the second inequality in (21) is not guaranteed to hold here.

### 5.4 Incorporating Prior Information

In certain cases, additional side information may be available in the form of a non-negative prior alignment matrix $\mathbf{W}\in\mathbb{R}_+^{n_B\times n_A}$ whose $(j,i)-$th entry denotes the degree of similarity between vertices $j\in\mathcal{V}_B$ and $i\in\mathcal{V}_A$. This information can be incorporated into the problem setup for improving the quality of the matching by adopting the following formulation

$$\min_{\mathbf{P}\in\mathcal{P}}\ \|\mathbf{B}-\mathbf{P}\mathbf{A}\mathbf{P}^T\|_F^2+(\lambda/2)\|\mathbf{P}-\mathbf{W}\|_F^2 \tag{23}$$

where $\lambda>0$ is a regularization parameter that effects a trade-off between adhering to the prior information and minimizing the edge disagreements. In terms of the vector $\mathbf{x}=\text{vec}(\mathbf{P})$, problem (23) can be equivalently reformulated as

$$\max_{\mathcal{S}\in\mathcal{I}_A\cap\mathcal{I}_B}\ \left\{h(\mathcal{S}):=\mathbb{1}_{\mathcal{S}}^T\mathbf{H}\mathbb{1}_{\mathcal{S}}+\lambda\mathbf{w}^T\mathbb{1}_{\mathcal{S}}\right\} \tag{24}$$

where $\mathbf{w}:=\text{vec}(\mathbf{W})$. In this case, the objective function $h(\mathcal{S})$ is also monotone, supermodular, being the sum of a supermodular function and a modular function. Hence, the MM framework can also be applied here for approximate maximization of (24). At each step of the algorithm, it suffices to only compute a modular approximation of the quadratic term $\mathbb{1}_{\mathcal{S}}^T\mathbf{H}\mathbb{1}_{\mathcal{S}}$ at the current solution set $\mathcal{S}_k$ via a subgradient $\mathbf{g}\in\partial f(\mathcal{S}_k)$ while leaving the modular term $\mathbf{w}^T\mathbb{1}_{\mathcal{S}}$ unchanged. This results in the following optimization subproblem

$$\max_{\mathcal{S}\in\mathcal{I}_A\cap\mathcal{I}_B}\ \left\{m^{(k)}(\mathcal{S}):=f(\mathcal{S}_k)+g(\mathcal{S})-g(\mathcal{S}_k)+\lambda\mathbf{w}^T\mathbb{1}_{\mathcal{S}}\right\} \tag{25}$$

which is again equivalent to maximizing a global lower bound on the objective function $h(\mathcal{S})$ at each step. The above problem can be simplified to the following form

$$\max_{\mathcal{S}\in\mathcal{I}_A\cap\mathcal{I}_B}\ \left\{u(\mathcal{S}):=(\mathbf{g}+\lambda\mathbf{w})^T\mathbb{1}_{\mathcal{S}}\right\} \tag{26}$$

which again corresponds to a maximum-weight bipartite matching problem. Depending on the scale of the problem, we can either choose to solve each subproblem (26) exactly or inexactly via the methods discussed previously.

## 6 Experimental Evaluation

In this section, we compare the performance of our proposed approximation algorithms against the prevailing start-of-the-art on real world graphs.

### 6.1 Baselines

We employed the following algorithms as performance benchmarks:

- **Umeyama's Method [30]:** A classical spectral method which uses the eigenvectors of both adjacency matrices $\mathbf{A}$ and $\mathbf{B}$ to construct a similarity matrix that is then used to perform max-weight bipartite matching to obtain the final alignment. Owing to the requirement of computing two full eigen-decompositions, for large graphs, the method is prone to suffering from high computational complexity.
- **IsoRank [4]:** A spectral method which considers the regularized formulation (24) and applies random walks with restarts to compute the PageRank eigenvector of the normalized Kronecker product graph $\mathbf{A}\otimes\mathbf{B}$ (with the prior alignment vector $\mathbf{w}$ serving

2. The working hypothesis here is that there is some measure of dissimilarity in the neighborhood vectors, i.e., the graphs $\mathcal{G}_A$ and $\mathcal{G}_B$ are *not* regular.

as the "teleportation" vector), followed by bipartite matching to obtain the final alignment.

- **Eigen-Align (EA) [18]:** A recent spectral method that uses the principal eigen-vectors of $\mathbf{A}$ and $\mathbf{B}$ to construct a rank-1 similarity matrix that is then used to perform bipartite matching to obtain the final alignment. As pointed out in [31], the matching problem admits a closed form solution in this case. The method is known to work well for aligning random Erdos-Renyi graphs [18], and enjoys low computational complexity.
- **Feature Engineering (FE) [32]:** An adaptation of the NetSimile method proposed in [32] for constructing graph embeddings. Every vertex of $\mathcal{G}_A$ and $\mathcal{G}_B$ is described by a feature vector with six attributes: its degree, average degree of its one-hop neighbors, clustering coefficient [33], average clustering coefficient of its one-hop neighbors, number of edges in its egonet[3], and number of outgoing edges from its egonet. Thereafter, a similarity matrix $\mathbf{S} \in \mathbb{R}^{n_B \times n_A}$ is created using Euclidean distances, i.e., the matrix has entries $S(j,i) = \|\mathbf{y}_j - \mathbf{x}_i\|_2^2$, where $\mathbf{x}_i \in \mathbb{R}^6$ (resp. $\mathbf{y}_j \in \mathbb{R}^6$) is defined as the feature vector corresponding to vertex $i \in \mathcal{V}_A$ (resp. $j \in \mathcal{V}_B$). A bipartite matching step is used to extract the final alignment from the similarity matrix. We point out that while the embeddings learned via NetSimile have found prior use in tasks like clustering, anomaly detection and visualization [32], to the best of our knowledge, they have not been used for graph matching before our present work.

## 6.2 Implementation

All algorithms were implemented in Matlab in a manner that does not require explicit computation of the Kronecker product matrix $\mathbf{A} \otimes \mathbf{B}$ and our experiments were carried out on a Windows computer outfitted with a Intel(R) i7 CPU with 16 GB RAM memory. For the baselines, instead of solving the final bipartite matching problem exactly, which can be time consuming, we used the greedy matching algorithm to obtain an approximate alignment at low complexity. Regarding the implementation of our MM-based algorithms, we used the following settings.

- **Choice of initialization:** A key factor that influences the performance of our algorithm is the choice of the initial matching set $\mathcal{S}^{(0)} \in \mathcal{M}$. As it is a non-trivial task to construct a good initialization for general instances, in our experiments, we use our algorithm to refine the output of the baselines. More specifically, we use the alignment computed by feature engineering to initialize our algorithm. This choice is determined by the fact that FE performs well (empirically) on many real world graphs at reasonable complexity, whereas the alignments determined by the other baselines (IsoRank, Eigen-Align) are either of poor quality, or do not offer substantial improvement (Umeyama) when used as initialization for our method.

3. The egonet of a vertex is defined as the subgraph induced by the vertex and its single-hop neighbors.

TABLE 1: Summary of network statistics: the number of vertices ($n$), the number of edges ($m$), the largest degree ($d_{\max}$), and network type.

| Network | $n$ | $m$ | $d_{\max}$ | Type |
|---|---|---|---|---|
| C.ELEGANS | 453 | 2,053 | 237 | interactome |
| A.THALIANA | 2,082 | 4,145 | 124 | interactome |
| STANFORD-CS | 2,759 | 10,270 | 303 | Web graph |
| JAPANESE | 3,177 | 7,998 | 725 | Word adjacency |
| CA-GRQC | 5,242 | 14,490 | 81 | co-authorship |
| PGP | 10,680 | 24,316 | 205 | social |
| AS-OREGON | 11,174 | 23,409 | 2,389 | router |
| AS-CAIDA | 26,475 | 53,381 | 2,628 | router |

- **Choice of subgradient:** We opted to use the subgradient $\check{\mathbf{g}}$ (with entries defined in (22b)) in our MM algorithms. This choice was based on the facts that it can be computed in $O(n_A)$ time and can be stored using at most $O(n_A)$ non-zero entries, which is *linear* in the size of the input graph $\mathcal{G}_A$. Additionally, it also delivered superior empirical performance compared to using $\hat{\mathbf{g}}$.
- **Choice of formulation:** In our experiments, we consider the regularized version (23) of the graph matching problem, where the prior node-alignment matrix $\mathbf{W}$ is set to be the similarity matrix obtained via feature engineering. We set the parameter $\lambda = 1e-4$ in our experiments.
- **Choice of inner solver:** Note that our algorithm is required to solve a max-weight bipartite matching problem at each step, which constitutes the main computational bottleneck. For exact inner approximation, we use the algorithm of [27], whereas for inexact approximation, we resort to using the Sinkhorn matrix balancing algorithm. The regularization parameter $\epsilon$ was chosen via trial and error, and the number of matrix balancing steps was limited to $5$. In this case, we perform a greedy matching step on the output of the Sinkhorn algorithm to obtain the final alignment at each iteration.

## 6.3 Datasets

We used real-world datasets drawn from different application domains (see Table 1) to test and compare the performance of the methods. These include (i) protein-protein interaction networks (C.ELEGANS [36] and A.THALIANA [6]), where the vertices are proteins and the edges correspond to their interactions, (ii) a web graph of the domain cs.stanford.edu [37], where the vertices are web pages, and the edges are symmetrized hyperlinks, (iii) a word adjacency network constructed from a Japanese text (JAPANESE) [38], (iv) a co-authorship network constructed from arXiv submissions in a scientific discipline (CA-GRQC [39]), where the vertices denote scientists, and the edges represent collaborations between co-authors of a publication, (v) a social network depicting the user interactions of the Pretty Good Privacy (PGP) algorithm [40], and (vi) a pair of internet router graphs (AS-OREGON [41] and AS-CAIDA [39]), where vertices are routers, and the edges represent the topology of the router network.

### 6.4 Experimental Setup

We design our experiments following common practices in the literature on graph matching [18]. Given a dataset, we first perform a pre-processing step to select the largest (strongly) connected component and eliminate all weights and self-loops. If the original graph is directed, a subsequent symmetrization step is additionally performed. In other words, we obtain a simple, undirected, unweighted graph $\mathcal{G}_A$.

Next, we create a "noisy" graph $\mathcal{G}_{\bar{A}}$ by randomly adding new edges with probability $p_e$, i.e., we generate a random Erdos-Renyi graph with edge-density $p_e$ and adjacency matrix $\mathbf{Q}$ and then create $\mathcal{G}_{\bar{A}}$ with adjacency matrix

$$\bar{\mathbf{A}} = \mathbf{A} + (\mathbf{1} - \mathbf{A}) * \mathbf{Q}.$$

Here, the operator "$*$" denotes the Hadamard (element-wise) product. Finally, we generate the graph $\mathcal{G}_B$ as a noisy, permuted version of $\mathcal{G}_A$ with adjacency matrix

$$\mathbf{B} = \mathbf{P}_* \bar{\mathbf{A}} \mathbf{P}_*^T,$$

where $\mathbf{P}_*$ is a randomly generated permutation matrix whose non-zero entries represent ground-truth alignments. Note that while $\mathcal{G}_B$ has the same number of vertices as $\mathcal{G}_A$, its edge-set $\mathcal{E}_B$ is a superset of $\mathcal{E}_A$. Hence, our goal here is to correctly align $\mathcal{E}_A$ with its counterpart present in $\mathcal{E}_B$. While our problem setup guarantees that there exists such a alignment (i.e., we have $\mathbf{P}_* \in \arg\min_{\mathbf{P}\in\mathcal{P}} \|\mathbf{B} - \mathbf{PAP}^T\|_F^2$), the solution is not guaranteed to be unique in general owing to the presence of topologically-invariant subgraphs such as cliques and star graphs in real world graphs [34]. We varied the noise level $p_e$ such that the percentage of extra edges in $\mathcal{G}_B$ ranged from $1-10\%$ of the edges in $\mathcal{G}_A$. For each noise-level, we averaged our results over 30 Monte-Carlo trials.

As an illustrative example of our setup, we display the degree distributions of the graphs corresponding to the dataset A.THALIANA ($\mathcal{G}_A$) and a noisy, permuted counterpart $\mathcal{G}_B$ generated by randomly adding $10\%$ extra edges on a log-log plot in Figure 1. It can be noted that the degree distributions of both graphs (approximately) obey a power-law, with few vertices of high degree (i.e., the hubs), while the majority of vertices have small degree. The distribution of $\mathcal{G}_B$ is slightly shifted to the right relative to $\mathcal{G}_A$ owing to the extra edges. Furthermore, it is also evident that perturbing $\mathcal{G}_A$ does not change the identities of the hubs, which lie above the noise-level due to their high degree. Hence, one expects that a high-quality algorithm should at least be capable of aligning the hubs in both graphs. The main difficulty then lies in producing high-quality alignments for the lower degree nodes, which are at the noise-level. Since real-world graphs are known to exhibit such skewed degree distributions [35], the aforementioned observation applies broadly in general.

### 6.5 Evaluation Metrics

In order to evaluate the performance of the methods, we used the following metrics.

1) **Edge correctness:** the ratio of the number of edge overlaps induced by the algorithm and the number of edges in $\mathcal{G}_A$. Perfectly aligning the edge-set of $\mathcal{G}_A$ with its counterpart in $\mathcal{G}_B$ results in a correctness score of 1.
2) **Relative Degree Difference:** Given a pair of aligned vertices $(i, \pi(i)) \in [n_A] \times [n_B]$, we measure their (degree-based) similarity according to the relative degree difference (RDD) metric, which is formally defined as
$$\mathrm{rdd}(i, \pi(i)) = \left(1 + \frac{|\deg(i) - \deg(\pi(i))|}{(\deg(i) + \deg(\pi(i)))/2}\right)^{-1},$$
where $\deg(.)$ returns the degree of the corresponding vertex. The RDD metric assigns higher value to aligned pairs of vertices with similar (relative) degrees, and thus provides another measure of assessing the quality of a given correspondence mapping. Here, we use the metric to assess how effective a method is in aligning different categories of vertices, ranging from hubs to low-degree nodes.
3) **Runtime**

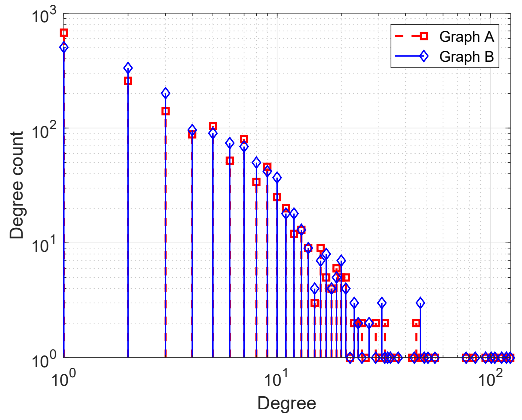

Fig. 1: The typical setup – degree distributions of A.THALANIA (Graph A) and a noisy, permuted counterpart (Graph B) with $10\%$ randomly added edges displayed as a log-log plot. Observe that the hubs in Graph A are largely unaffected by the perturbation, as opposed to the low-degree nodes.

### 6.6 Results and Discussion

We first study the number of iterations required by our algorithms to attain convergence in terms of the objective function when initialized using Feature Engineering (FE). As an illustrative example, we chose the dataset A.THALIANA, created a noisy permuted graph with approximately $10\%$ more edges, and ran both the exact and inexact versions of our methods. Figure 2 displays the evolution of the objective function (i.e., the number of edge overlaps) with iterations. As expected, solving each subproblem exactly results in larger relative improvement in objective value (approx. $5\%$) compared to inexact-MM. However, in terms of run-time, the inexact algorithm is around an order of magnitude faster on average ($2.5s$ vs $24s$). Most importantly, the algorithms converge in approximately 1 iteration, which means we can do away with additional expensive iterations. We made a similar observation across all datasets, and henceforth, we use only a single iteration to refine the output of FE.

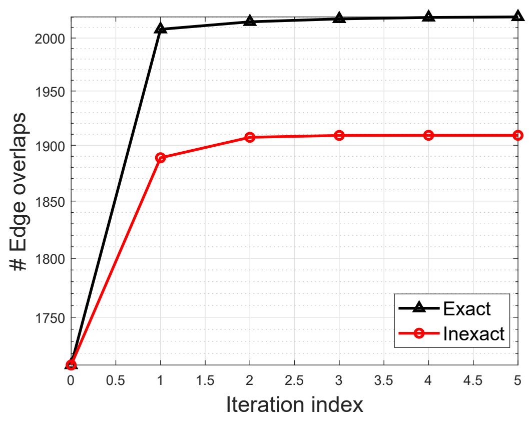

Fig. 2: Evolution of objective function with iterations on the A.THALANIA dataset, averaged over 30 trials. Observe that the knee in the curves occurs in the first iteration.

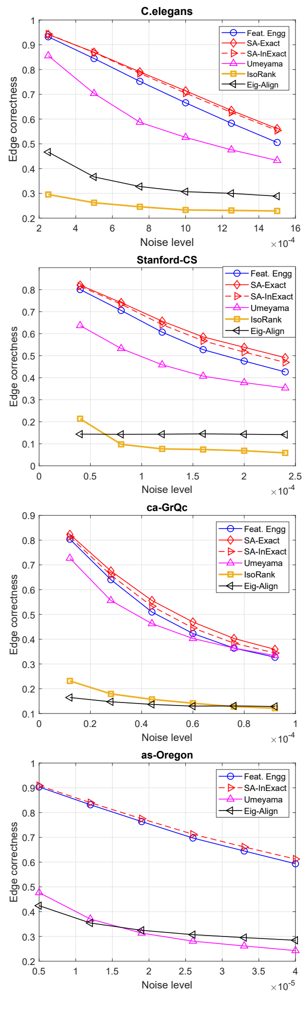

Fig. 3: Edge correctness vs. noise level across datasets. For each noise level, 30 different graphs $\mathcal{G}_B$ with a certain percentage of additional edges were generated. The range is varied to generate $1-10\%$ additional edges of the given graph $\mathcal{G}_A$. The higher the curve, the better.

As we will see, this single iteration can yield substantial improvement in performance with respect to the previously defined metrics.

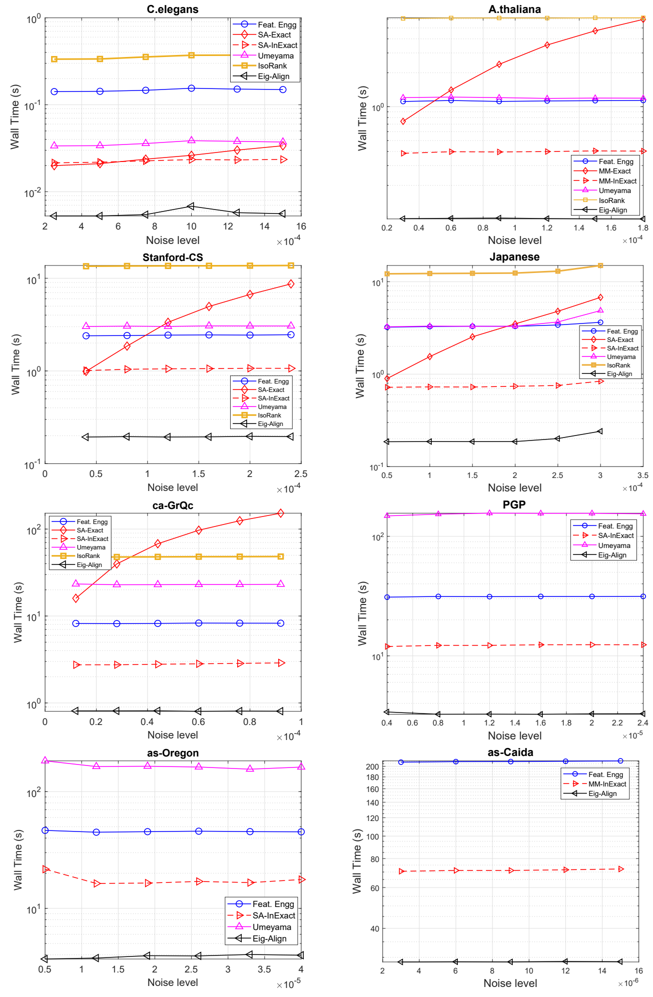

Fig. 4: Wall time (in seconds) vs. noise level across datasets. The lower the curve, the better.

The performance of the algorithms on the different datasets in terms of edge correctness and runtime are depicted in figures 3 and 4 respectively. With regard to edge correctness, IsoRank and Eigen-Align (EA) perform the worst in general. While Umeyama's spectral method exhibits better performance, it is outperformed by FE in almost all cases. On the first five datasets, a single iteration of both exact and inexact MM algorithms performed on the solution of FE can bring about substantial improvement in edge correctness, especially in the higher noise regime (ranging from $10\%$ on CA-GRQC to $18\%$ on A.THALIANA). However, with regard to scalability, exact-MM exhibits the worst performance, which is why it is omitted from the 3 largest datasets. A similar observation can be made regarding IsoRank as well. The increasing complexity of performing full EVDs in Umeyama's spectral method is also evident in the timing results across the datasets, with it being omitted from the largest dataset. On the other hand, EA is consistently the fastest, while inexact-MM comes in second, being $3-5$ times faster than FE on average. This showcases the benefit of using the combined Sinkhorn matrix balancing and greedy matching strategy, as it brings about significant reduction in complexity compared to exact-MM, while preserving performance in terms of edge correctness. Finally, it can be observed that on the last two datasets AS-OREGON and AS-CAIDA, inexact-MM does not significantly improve the edge correctness compared to FE.

In order to obtain a better understanding of the performance of our MM-based algorithms with respect to FE, we perform the following experiment. Recall our working hypothesis that under the aforementioned experimental setup, a good algorithm should be capable of aligning the hubs (which are few) in real graphs with power-law degree distributions, but aligning the low degree vertices (which are many) is a more difficult proposition. From Figure 5, since FE performs the best overall amongst the baselines, we tested its efficacy in aligning different "categories" of vertices (according to their degrees) on each dataset. More specifically, given a dataset, we assigned each vertex to one of $4-5$ categories, depending on how large its degree is relative to the largest degree $d_{\max}$. These degree categories are represented in the form of a pie chart in the left column of Figure 3. As expected, the majority of the vertices in our datasets lie in the "bottom" category. For each such category, we computed the average RDD score of all vertices across all Monte-Carlo trials in $\mathcal{G}_A$ and their matched counterparts in $\mathcal{G}_B$ via the correspondence mapping determined by FE. We also repeated this procedure to compute average RDD scores of both MM algorithms of all vertices in each category. Finally, we display the improvement obtained by our methods relative to FE in terms of the average RDD scores over each category in the right hand column of Figure 3 for the largest noise level (which adds approximately $10\%$ additional edges), with the leftmost category (labeled as (A)) representing the hubs and the rightmost category denoting the low-degree vertices. What we observe is that for the hubs, our algorithms in general do not improve the alignment significantly over FE, which indicates that FE does indeed succeed in finding high quality alignments for the high degree vertices. Note that our methods do not bring about a degradation in the quality of these alignments. Instead, using the alignments for the hubs as "anchors", it can be seen that the MM methods focus on improving the alignment quality of the lower degree vertices, as evidenced by the relative improvement in RDD scores. This suggests that the improvement in edge correctness brought about by our methods stems from providing better quality alignments for the smaller vertices, which is the more difficult task compared to aligning hubs – and in certain cases, also the more interesting, e.g., when trying to match small footprint groups, or rare drug interactions. We point out that improving the alignments for the small vertices is not guaranteed to substantially improve the edge correctness. For example, the AS-OREGON and AS-CAIDA graphs possesses a very skewed degree distribution, with $92\%$ of the vertices having degrees smaller than $5$. In these cases, beyond aligning the hubs, it is difficult to improve the edge correctness on the basis of topology alone. On a final note, a small drawback of using the inexact-MM algorithm is that in certain cases, it can slightly degrade the average quality of alignment for vertices of certain degrees. This motivates exploring additional ways of improving the quality-complexity trade-off in approximately solving each matching subproblem in our MM framework, which we leave for future work.

## 7 CONCLUSION

In this paper, we presented a new result on graph matching regarding its formulation as maximizing a monotone, supermodular function subject to matroid intersection constraints. Adopting this view led to employing a successive approximation algorithm which utilizes the discrete subgradients of the objective function to perform iterative maximization by solving a sequence of maximum-weight bipartite matching problems. The algorithm naturally respects the combinatorial constraints of the problem, does not require computing expensive Kronecker products, and can retain its performance at reduced complexity even when the subproblems are solved approximately. The effectiveness of the approach was validated on real networks, which demonstrated its superior performance relative to the prevailing state-of-the-art. As an unanticipated but welcome bonus, we discovered that an adaptation of the NetSimile "handcrafted" features originally proposed in [32] for constructing graph embeddings can be effectively used for graph matching. This serves as a good initialization for our approach, and also outperforms many of the established graph matching baselines in our experiments with real graphs. The combination of this FE method with our discrete optimization approach gives the best results in all cases we tried.

## REFERENCES

[1] A. C. Berg, T. L. Berg, and J. Malik, "Shape matching and object recognition using low distortion correspondences," *Proc. Intl. Conf. Comp. Vision* , pp. 26–33, June 2005.

[2] A. Egozi, Y. Keller, and H. Guterman, "Improving shape retrieval by spectral matching and meta similarity," *IEEE Trans. Image Process.*, vol. 19, no. 5, pp. 1319–1327, May 2010.

[3] W. Brendel and S. Todorovic, "Learning spatio-temporal graphs of human activities", *Proc. Proc. Intl. Conf. Comp. Vision*, pp. 778–785, Nov. 2011.

[4] R. Singh, J. Xu, and B. Berger, "Global alignment of multiple protein interaction networks with application to functional orthology detection," in *Proc. Nation. Academ. of Scien.*, vol. 105, no. 35, pp. 763—768, 2008.

[5] M. Zaslavskiy, F. Bach, and J.-P. Vert, "Global alignment of protein–protein interaction networks by graph matching methods," *Bioinformatics*, vol. 25, no. 12, pp. 1259–1267, 2009.

[6] R. Patro and C. Kingsford, "Global network alignment using multiscale spectral signatures," *Bioinformatics*, vol. 28, no. 23, pp. 3105–3114, 2012.

[7] S. Lacoste-Julien, B. Taskar, D. Klein, and M. I. Jordan, "Word alignment via quadratic assignment," in *Proc. NAACL*, pp. 112–119, 2006.

[8] A. Narayanan and V. Shmatikov, "De-anonymizing social networks," in *Proc. IEEE Symp. Secur. and Privacy*, pp. 173—187, 2009.

[9] L. Babai, "Graph isomorphism in quasi-polynomial time", *Proc. ACM Symp. Theory of Comput.*, pp. 684—697, June 2016.

[10] T. C. Koopmans, and M. Beckmann, "Assignment problems and the location of economic activities", *Econometrica*, pp. 53–76, 1957.

[11] S. Sahni, and T. Gonzalez, "P-complete approximation problems", *J. of ACM*, vol. 23, no. 3, pp.555–565, July 1976.

[12] H. D. Sherali, and W. P. Adams, "*A reformulation-linearization technique for solving discrete and continuous nonconvex problems*", Vol. 31. Springer Science & Business Media, 2013.

[13] K. M. Anstreicher, and N. W. Brixius, "Solving quadratic assignment problems using convex quadratic programming relaxations," *Optimiz. Meth. and Soft.*, vol. 16, no. 1-4, pp. 49—68, 2001.

[14] J. T. Vogelstein, J. M. Conroy, V. Lyzinski, L. J. Podrazik, S. G. Kratzer, E. T. Harley, D. E. Fishkind, R. J. Vogelstein, and C. E. Priebe, "Fast approximate quadratic programming for graph matching," *PLOS one*, vol. 10, no. 4, 2015.

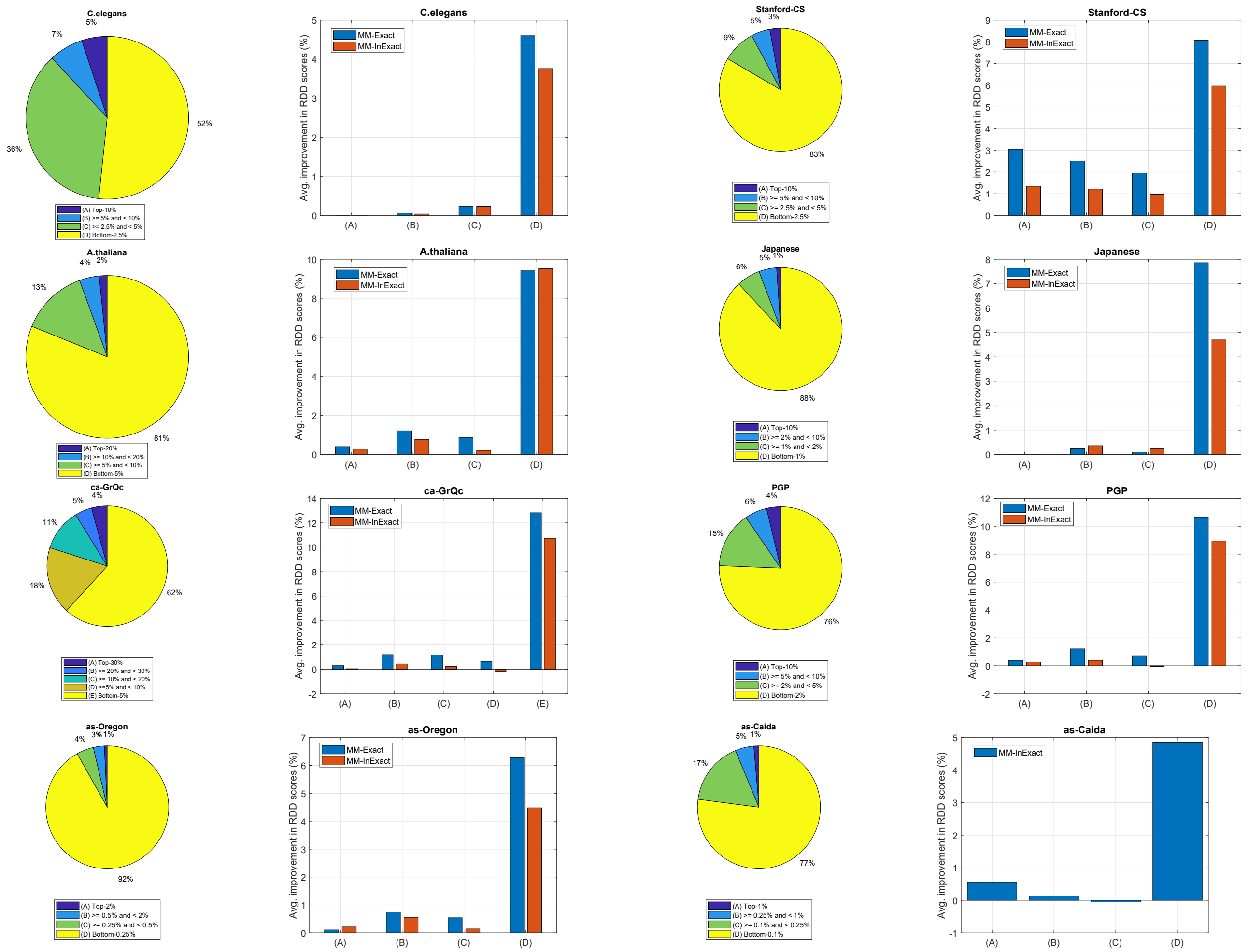

Fig. 5: **Left column:** Distribution of vertices according to their size with respect to the largest degree vertex in each dataset. **Right column:** Relative improvement in average RDD scores over FE for each category of vertices - category (A) represents the top vertices by degree, while the rightmost category represents the smallest vertices by degree.

[15] G. W. Klau, "A new graph-based method for pairwise global network alignment," *BMC Bioinformat.*, vol. 10, no. Suppl 1, p. S59, 2009.

[16] J. Peng, H. Mittelmann, and X. Li, "A new relaxation framework for quadratic assignment problems based on matrix splitting," *Mathem. Program. Computat.*, vol. 2, no. 1, pp. 59—77, 2010.

[17] M. Bayati, D. F. Gleich, A. Saberi, and Y. Wang, "Message-passing algorithms for sparse network alignment," *ACM Trans. Knowl. Discov. Data*, vol. 7, no. 1, p. 3, 2013.

[18] S. Feizi, G. Quon, M. Recamonde-Mendoza, M. Medard, M. Kellis, and A. Jadbabaie, "Spectral alignment of graphs," *IEEE Trans. Netw. Science*, April 2019.

[19] F. Bach, "Learning with submodular functions: A convex optimization perspective," *Found. Trends in Mach. Learn.*, vol. 6, no. 2-3, pp. 145–373, Dec. 2013.

[20] W. Bai, and J. Bilmes, "Greed is still good: Maximizing monotone submodular + supermodular functions", *Proc. Intl. Conf. Mach. Learn.*, pp. 314-323, Jul. 2018.

[21] A. Konar, and N. D. Sidiropoulos, "Iterative graph alignment via supermodular approximation", *Proc. of IEEE Intl. Conf. Data Mining*, Beijing, China, Nov. 2019.

[22] L. Lovasz, "Submodular functions and convexity", in *Mathematical Programming – The State of the Art*, pp. 235–257, Springer Berlin Heidelberg, 1983.

[23] S. Fujishige, "*Submodular functions and optimization*", 2nd edition, Annals of Disc. Math., vol. 58, 2005.

[24] F. Bach, "Learning with submodular functions: A convex optimization perspective," *Found. Trends in Mach. Learn.*, vol. 6, no. 2-3, pp. 145–373, Dec. 2013.

[25] J. Oxley, "*Matroid Theory*," Oxford University Press, 2011.

[26] H. W. Kuhn, "The Hungarian method for the assignment problem", *Nav. Res. Logist. Quart.*, no. 1–2, pp. 83-97, 1955.

[27] R. Jonker and A. Volgenant, "A shortest augmenting path algorithm for dense and sparse linear assignment problems", *Computing*, vol. 38, no. 4, pp. 325–340, Dec. 1987.

[28] J. B. Orlin, "A polynomial time primal network simplex algorithm for minimum cost flows", *Mathem. Progamm.* vol. 78, no. 2, pp. 109–129, Aug. 1997.

[29] R. Sinkhorn, "Diagonal equivalence to matrices with prescribed row and column sums", *The Amer. Math. Monthly*, vol. 74, no. 2, pp. 402–402, 1967.

[30] S. Umeyama, "An eigen-decomposition approach to weighted graph matching problems", *IEEE Trans. Pattern Analys. and Mach. Intell.*, vol. 10, no. 5, pp. 695—703, Sept. 1988.

[31] X. Liu, and S.-H. Teng, "Maximum bipartite matchings with low rank data," *Theor. Comput. Sci.*, vol. 621, pp. 82–91, Mar. 2016.

[32] M. Berlingerio, D. Koutra, T. E.-Rad, and C. Faloutsos, "Netsimile: A scalable approach to size-independent network similarity", in *Proc. IEEE/ACM Intl. Conf. Adv. Social Netw. Analys. and Mining*, pp. 1439–1440, Aug. 2013.

[33] D. J. Watts, and S. H. Strogatz, "Collective dynamics of 'small-world' networks," *Nature*, vol. 393, no. 6684, pp. 440–442, 1998.

[34] Y. Lim, U. Kang, and C. Faloutsos, "Slashburn: Graph compression and mining beyond caveman communities," *IEEE Trans. on Knowl. and Data Eng.*", vol. 26, no. 12, pp. 3077–3089, Apr. 2014.

[35] M. Faloutsos, P. Faloutsos, and C. Faloutsos, "On power-law relationships of the internet topology", *ACM SIGCOMM Comput. Commun. Rev.*, vol. 29, no. 4, pp. 251–262, Aug. 1999.

[36] J. Kunegis, “KONECT - The Koblenz Network Collection”, in *Proc. ACM Int. Conf. on World Wide Web*, pp. 1343–1350, May 2013.
[37] F. Bonchi, P. Esfandiar, D. F. Gleich, C. Greif, and L. V. Lakshmanan, “Fast matrix computations for pairwise and columnwise commute times and Katz scores,” *Intern. Mathem.*, vol. 8, no. 1, pp. 73–112, Mar. 2012.
[38] R. Milo, S. Itzkovitz, N. Kashtan, R. Levitt , S. Shen-Orr, I. Ayzenshtat, M. Sheffer, and U. Alon, “Superfamilies of evolved and designed networks”, *Science*, vol. 303, no. 5663, pp. 1538–42, Mar. 2004.
[39] J. Leskovec, J. Kleinberg, and C. Faloutsos, “Graph evolution: Densification and shrinking diameters”, *ACM Trans. Knowl. Discov. Data*, vol. 1, no. 1, pp. 2, Mar. 2007.
[40] M. Boguna, R. Pastor-Satorras, A. Diaz-Guilera, and A. Arenas, “Models of social networks based on social distance attachment,” *Phys. Rev. E*, vol. 70, no. 5, pp. 056122, Nov. 2004.
[41] J. Leskovec, and A. Krevl, SNAP Datasets: Stanford Large Network Dataset Collection, 2015. Available at http://snap.stanford.edu/data.

**Aritra Konar** (M’17) received the B.Tech. degree in Electronics and Communications Engineering from West Bengal University of Technology, West Bengal, India, and the M.S. and Ph.D. degrees in Electrical Engineering from the University of Minnesota, Minneapolis, USA, in 2011, 2014, and 2017 respectively. He is currently a Postdoctoral Associate in the Department of ECE, University of Virginia, VA, USA. His research interests include graph mining, nonlinear optimization and data analytics.

**Nicholas D. Sidiropoulos** (F’09) earned the Diploma in Electrical Engineering from Aristotle University of Thessaloniki, Greece, and M.S. and Ph.D. degrees in Electrical Engineering from the University of Maryland at College Park, in 1988, 1990 and 1992, respectively. He has served on the faculty of the University of Virginia, University of Minnesota, and the Technical University of Crete, Greece, prior to his current appointment as Louis T. Rader Professor and Chair of ECE at UVA. From 2015 to 2017 he was an ADC Chair Professor at the University of Minnesota. His research interests are in signal processing, communications, optimization, tensor decomposition, and factor analysis, with applications in machine learning and communications. He received the NSF/CAREER award in 1998, the IEEE Signal Processing Society (SPS) Best Paper Award in 2001, 2007, and 2011, served as IEEE SPS Distinguished Lecturer (2008-2009), and as Vice President - Membership of IEEE SPS (2017-2019). He received the 2010 IEEE Signal Processing Society Meritorious Service Award, and the 2013 Distinguished Alumni Award from the University of Maryland, Dept. of ECE. He is a Fellow of IEEE (2009) and a Fellow of EURASIP (2014).