Soft Graph Matching: Submodular Relaxation and Lovász Extension

Aritra Konar
Dept. of Electrical and Computer Engineering
University of Virginia
Charlottesville, VA, USA
aritra@virginia.edu

Nicholas D. Sidiropoulos
Dept. of Electrical and Computer Engineering
University of Virginia
Charlottesville, VA, USA
nikos@virginia.edu

***Abstract*—Graph matching aims to align a pair of graphs by minimizing their edge disagreements. As the problem is NP–hard in the worst-case, various methods have been proposed for approximately solving the problem. One popular approach is to relax the combinatorial problem to a continuous formulation, whose solution represents a soft correspondence between the vertex-sets of the graphs. Previous work has primarily motivated such soft matching formulations as an intermediate step towards obtaining hard correspondences. In this paper, we depart from this viewpoint and provide an alternate motivation for soft matching as a means of identifying classes of topologically-invariant subgraphs, which cannot be revealed by hard correspondences. Drawing on this observation, we consider the family of doubly-stochastic relaxations for graph matching and propose a new convex relaxation for the problem. We establish that the objective function of our formulation can be interpreted as the tightest convex relaxation of the combinatorial quadratic graph matching objective function in a certain sense, and describe an efficient first-order algorithm for computing its solution. Through experiments conducted on real-world data, we demonstrate the empirical effectiveness of the algorithm relative to the prevailing relaxations for graph matching.**

I. Introduction

Graph matching refers to the problem of computing a one-to-one correspondence mapping between the vertex sets of a pair of graphs such that the edge-disagreements induced by the mapping are minimized. The problem has diverse applications ranging from bio-informatics, to computer vision, and data mining (see [1], [2] and references therein). However, the problem is combinatorial in nature and is known to be notoriously difficult to solve, being equivalent to the NP–hard quadratic assignment problem [3] in the worst-case.

In light of the above fact, approximation algorithms are widely used with the goal of computing high-quality albeit sub-optimal solutions for the problem. A commonly used technique is to employ a relaxation of the combinatorial constraints and solve the resulting continuous optimization problem to obtain a soft-correspondence matrix, whose entries indicate the likelihood of assigning a pair of vertices to each other. In prior work, such "soft" formulations of graph matching have primarily been used as a means to the end of obtaining a hard correspondence mapping, usually by performing a final "hardening" step by solving a linear assignment problem with the soft-correspondence matrix used as the objective function.

Supported by the National Science Foundation under grant IIS 1908070.

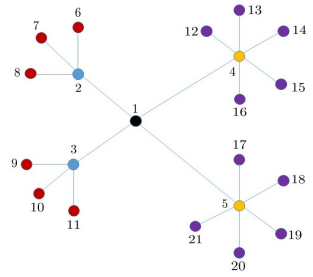


Fig. 1: Invariance classes in a graph $\mathcal{G}_A$. Swapping a pair of vertices of the same color (together with their descendants, if any) does not affect the topology of the graph. Note that a one-to-one correspondence is incapable of revealing the invariance classes.

In this paper, we motivate soft matching from a different perspective. Our starting point is the fact that real-world graphs are often comprised of topologically invariant subgraphs such as cliques and star graphs [4]. Consequently, performing hard alignment may not always be meaningful, as it does not convey any information regarding the classes of topologically-invariant vertices. For example, consider the graph depicted in Figure 1, where each different coloring of the vertices depicts an invariance class. Note that exchanging labels between any pair of vertices belonging to the same class, together with their descendants (if any), does not affect the topology of the graph. In such a case, given the graph and one of its isomorphic forms, computing a one-to-one correspondence between their vertex sets is not well motivated. Consequently, here it is of greater importance to identify the different invariance classes in the graphs, and it is for this task that we consider the problem of designing an appropriate soft matching formulation of graph matching. More specifically, our main contributions are as follows:

1) We consider the family of doubly-stochastic (DS) relaxations for graph matching (which employ doubly stochastic matrices as soft correspondences), and introduce a new non-smooth, convex DS formulation. Leveraging a recent result [5] which shows that the quadratic objective

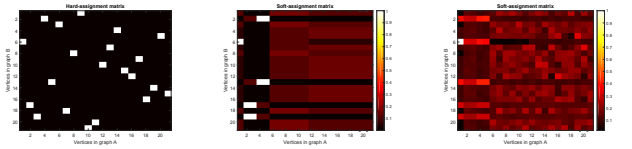
2374-8486/20/$31.00 ©2020 IEEE
DOI 10.1109/ICDM50108.2020.00035

function in graph matching is a submodular function, we establish that our soft matching formulation can be viewed as the tightest convex relaxation of graph matching, within the family of DS relaxations.

2) We provide an intuitive explanation of why our formulation constitutes a meaningful relaxation of graph matching. We apply the insights gained to improve the potency of our formulation for aligning unweighted graphs by proposing practical one-hop neighborhood based edge-weighting strategies.
3) Although our soft matching formulation is a convex problem, we show that it possesses non-Euclidean geometry, which constitutes a major roadblock in efficiently computing its solution using the standard toolkit of convex optimization. By judiciously adapting to the geometry of the problem, we show that a non-Euclidean projected subgradient algorithm of Nesterov [6] can be employed to efficiently solve the problem. We also describe a stochastic approximation variant of the algorithm, which endows it with the ability to scale up to larger problem instances.

In order to put the scope of our contributions into context, we now discuss prior and related work on soft matching relaxations. Apart from the family of DS relaxations, other techniques include orthogonal relaxation, semidefinite relaxation [7], and integer linear programming relaxations [8]. The first method is not known to perform well in practice and is seldomly used, while the latter two methods stem from expensive reformulations of graph matching that entail the introduction of a large number of variables and constraints, and are limited to small instances. These techniques serve as means to ultimately producing a hard alignment. Note that this mode of use is not our primary motivation in this paper. The class of DS relaxations, on the other hand, is naturally suited for our interpretation of soft matching, which is the reason why we consider it in this paper. Prevailing DS relaxations include convex and non-convex quadratic programming formulations. We show that our proposed convex DS relaxation is the tightest convex relaxation in this family (in a certain sense) and propose an algorithm for efficiently solving it that compares very favorably to state-of-the-art baselines for solving the other DS relaxations [9], [10] on real-world graphs. As a preview of our main results, we used our convex formulation and the prevailing convex formulation for performing soft matching between the graph $\mathcal{G}_A$ in Figure 1 and an isomorphic variant $\mathcal{G}_B$. We display heatmaps of the correspondence matrices (normalized by the largest entry) obtained as solutions to both problems in Figure 2. Clearly, the solution of our formulation does a much better job at delineating the invariance classes in the graph.

Finally, we point out that a recent paper [5] also exploits the submodularity of the graph matching objective function to develop an iterative approximation algorithm. However, the algorithm is combinatorial and aims to compute hard alignments, whereas in our work we utilize the submodularity property in a very different way to develop a principled convex relaxation for soft matching.



(a) Permutation relating $\mathcal{G}_A$ and $\mathcal{G}_B$. (b) Optimal solution of our formulation. (c) Optimal solution of prevailing formulation.

Fig. 2: Visualizing invariance classes in $\mathcal{G}_A$ (shown in Figure 1) and its isomorphic counterpart $\mathcal{G}_B$ via soft correspondences. Lighter (resp. darker) colors indicate higher (resp. smaller) correspondence values.

II. Preliminaries

Consider a pair of undirected graphs $\mathcal{G}_A, \mathcal{G}_B$ on n vertices with vertex sets $\mathcal{V}_A = \mathcal{V}_B = [n] := \{1, \cdots, n\}$, and edge sets $\mathcal{E}_A, \mathcal{E}_B \subseteq [n] \times [n]$. Graph matching aims to compute a bijective mapping between the vertex sets of the two graphs which induces minimal edge disagreements. Formally, the problem can be expressed as

$$\min_{\mathbf{P} \in \mathbb{P}^n} \frac{1}{2} \|\mathbf{B} - \mathbf{P}\mathbf{A}\mathbf{P}^T\|_F^2, \tag{1}$$

where $\mathbf{A}, \mathbf{B} \in \mathbb{S}^n$ denote the (positively weighted) adjacency matrices of the graphs $\mathcal{G}_A$ and $\mathcal{G}_B$ respectively, and $\mathbb{P}^n$ denotes the set of $n \times n$ permutation matrices. Note that (1) admits the following equivalent representation

$$\min_{\mathbf{P} \in \mathbb{P}^n} \frac{1}{2} \|\mathbf{B}\mathbf{P} - \mathbf{P}\mathbf{A}\|_F^2, \tag{2}$$

by virtue of the unitary property of the set of matrices $\mathbb{P}^n$. Expanding the cost function of (2) and applying the same property produces another equivalent form

$$\min_{\mathbf{P} \in \mathbb{P}^n} \frac{\|\mathbf{A}\|_F^2 + \|\mathbf{B}\|_F^2}{2} - \langle \mathbf{B}\mathbf{P}, \mathbf{P}\mathbf{A} \rangle. \tag{3}$$

Since graph matching is a NP–hard optimization problem, in practice one often resorts to relaxing the "hard" combinatorial matching constraints $\mathbb{P}^n$ to the set of doubly-stochastic matrices

$$\mathbb{B}^n := \{\mathbf{S} \in \mathbb{R}_+^{n \times n} : \mathbf{S}\mathbf{1}_n = \mathbf{1}_n, \mathbf{S}^T\mathbf{1}_n = \mathbf{1}_n\}, \tag{4}$$

which constitutes the convex hull of $\mathbb{P}^n$ and represents "soft" matching constraints. However, adopting such a measure breaks the equivalence amongst the problems (1)–(3) and produces two distinct relaxations. This raises the following question: which relaxation should one employ in practice? This question is considered next.

III. Relating graph matching relaxations

Given a doubly-stochastic matrix $\mathbf{S} \in \mathbb{B}^n$, define the pair of functions

$$f(\mathbf{S}) := \frac{1}{2} \|\mathbf{B}\mathbf{S} - \mathbf{S}\mathbf{A}\|_F^2 \tag{5a}$$

$$g(\mathbf{S}) := \frac{\|\mathbf{A}\|_F^2 + \|\mathbf{B}\|_F^2}{2} - \langle \mathbf{B}\mathbf{S}, \mathbf{S}\mathbf{A} \rangle. \tag{5b}$$

Minimizing each of these functions over the set $\mathbb{B}^n$ yields a distinct relaxation for the original problem (1). To be precise,

$$(\mathbf{R1}) \ \min_{\mathbf{S}\in\mathbb{B}^n} f(\mathbf{S}) \tag{6}$$

is a relaxation of (2), and is a convex quadratic programming (QP) problem, as the cost function is the composition of a convex function with a linear map, and the set $\mathbb{B}^n$ is a bounded polyhedron. Meanwhile,

$$(\mathbf{R2}) \ \min_{\mathbf{S}\in\mathbb{B}^n} g(\mathbf{S}) \tag{7}$$

corresponds to a relaxation of (3) and is a *non-convex* QP problem. While these relaxed problems are not equivalent, their optimal values yield lower bounds on the optimal value of (1), which is NP–hard to determine in general. However, the question of which of the above relaxations yields the tighter lower bound on the optimal value of (1) has not been previously settled. A partial answer to this question has been provided in [11], which establishes that for a specific family of random graphs, the optimal value of (**R2**) coincides with that of (1) almost surely, whereas the optimal value of (**R1**) does not. We now extend this result [1] to the case of general graphs. To this end, we will require the following useful result, which is extracted from [12, Lemma 6].

Lemma 1. *Given any matrix $\mathbf{W} \in \mathbb{R}^{n\times n}$, for every $\mathbf{S} \in \mathbb{B}^n$, we have*

$$\|\mathbf{WS}\|_F^2 \leq \|\mathbf{W}\|_F^2, \text{ and } \|\mathbf{SW}\|_F^2 \leq \|\mathbf{W}\|_F^2.$$

Let OPT(**R1**) and OPT(**R2**) denote the optimal values of their respective problems. With Lemma 1 in hand, we can establish the following result.

Theorem 1. *For any pair of undirected graphs $\mathcal{G}_A, \mathcal{G}_B$, it holds that*

$$\text{OPT}(\textbf{\textit{R1}}) \leq \text{OPT}(\textbf{\textit{R2}})$$

Proof. It suffices to show that

$$f(\mathbf{S}) \leq g(\mathbf{S}), \forall\, \mathbf{S} \in \mathbb{B}^n. \tag{8}$$

To begin with, note that $f(\mathbf{S})$ can be expressed as

$$f(\mathbf{S}) = \frac{\|\mathbf{SA}\|_F^2 + \|\mathbf{BS}\|_F^2}{2} - \langle \mathbf{BS}, \mathbf{SA}\rangle, \tag{9}$$

from which it is evident that $f(\mathbf{S})$ and $g(\mathbf{S})$ share a common last term. Hence, the condition (8) reduces to

$$\|\mathbf{SA}\|_F^2 + \|\mathbf{BS}\|_F^2 \leq \|\mathbf{A}\|_F^2 + \|\mathbf{B}\|_F^2, \forall\, \mathbf{S} \in \mathbb{B}^n. \tag{10}$$

According to Lemma 1, for any $\mathbf{S} \in \mathbb{B}^n$, we always have

$$\|\mathbf{SA}\|_F^2 \leq \|\mathbf{A}\|_F^2, \text{ and } \|\mathbf{BS}\|_F^2 \leq \|\mathbf{B}\|_F^2 \tag{11}$$

[1]The work of [11] proves a stronger result, namely that for a class of random graph models, solving (**R2**) actually *recovers* the optimal solution of (1) with probability 1, while solving (**R1**) does not. Here, our focus is not on extending the recoverability result, but rather quantifying the relationship between the two relaxations in terms of the optimal values that they yield for the general graph matching problem.

Simply summing the pair of inequalities then yields the desired claim. □

While Theorem 1 establishes that (**R2**) is guaranteed to be a tighter relaxation compared to (**R1**), it entails solving a non-convex QP problem, which is NP–hard in general.

IV. A NEW CONVEX RELAXATION FOR GRAPH MATCHING

In this section, we introduce a new convex relaxation for the graph matching problem (1). Our starting point is the observation that the inner product $\langle \mathbf{BP}, \mathbf{PA}\rangle$ appearing in the cost function of (3) can be equivalently expressed in quadratic form as follows

$$\begin{aligned}\langle \mathbf{BP}, \mathbf{PA}\rangle &= \text{Trace}(\mathbf{P}^T\mathbf{BPA}) &\text{(12a)}\\ &= \text{vec}(\mathbf{P})^T\text{vec}(\mathbf{BPA}) &\text{(12b)}\\ &= \text{vec}(\mathbf{P})^T(\mathbf{A}\otimes\mathbf{B})\text{vec}(\mathbf{P}), &\text{(12c)}\end{aligned}$$

where we have utilized the property

$$\text{vec}(\mathbf{CXD}) = (\mathbf{D}^T \otimes \mathbf{C})\text{vec}(\mathbf{X})$$

of the vec(.) operator, and the fact that $\mathbf{A}$ is symmetric.

Let $m = n^2$, and define the m-dimensional binary vector $\mathbf{x} := \text{vec}(\mathbf{P})$ and the $m \times m$ matrix $\mathbf{H} := \mathbf{A} \otimes \mathbf{B}$. Then, (12c) can be expressed in quadratic form as

$$\langle \mathbf{BP}, \mathbf{PA}\rangle = \mathbf{x}^T\mathbf{Hx}. \tag{13}$$

Observe that $\mathbf{H}$ can be viewed as the adjacency matrix of a graph $\mathcal{G}_H$ on m vertices with vertex set $\mathcal{V}_H := \mathcal{V}_A \times \mathcal{V}_B$; i.e., a vertex $u \in \mathcal{V}_H$ corresponds to a pair of vertices (i, j), with $i \in \mathcal{V}_A$ and $j \in \mathcal{V}_B$. An edge exists between a pair of vertices $u = (i, j)$ and $v = (k, l)$ in $\mathcal{G}_H$ if and only if $(i, k) \in \mathcal{E}_A$ and $(j, l) \in \mathcal{E}_B$. Note that the weight of such an edge is $w_{(u,v)} := A_{(i,k)}B_{(j,l)}$. The edge set of $\mathcal{G}_H$ is denoted as $\mathcal{E}_H \subseteq [m] \times [m]$. Let $\mathbf{d}_H \in \mathbb{R}^m$ represent the (weighted) degree vector of the vertices of $\mathcal{G}_H$. This can be expressed as

$$\begin{aligned}\mathbf{d}_H = \mathbf{H}\mathbf{1}_m &= (\mathbf{A}\otimes\mathbf{B})(\mathbf{1}_n \otimes \mathbf{1}_n)\\ &= (\mathbf{A}\mathbf{1}_n)\otimes(\mathbf{B}\mathbf{1}_n) = \mathbf{d}_A \otimes \mathbf{d}_B,\end{aligned} \tag{14}$$

where we have invoked the mixed-product rule for Kronecker products, and $\mathbf{d}_A, \mathbf{d}_B$ are the (weighted) degree vectors of $\mathcal{G}_A$ and $\mathcal{G}_B$ respectively. The graph Laplacian associated with $\mathcal{G}_H$ is then given by $\mathbf{L}_H := \text{diag}(\mathbf{d}_H) - \mathbf{H}$. For any vector $\mathbf{z} \in \mathbb{R}^m$, the Laplacian quadratic form associated with $\mathbf{L}_H$ can be written as

$$\mathbf{z}^T\mathbf{L}_H\mathbf{z} = \sum_{(u,v)\in\mathcal{E}_H} w_{(u,v)}(z_{(u)} - z_{(v)})^2. \tag{15}$$

With these definitions in place, from (15), we obtain the following equivalent expressions

$$\begin{aligned}\langle \mathbf{BP}, \mathbf{PA}\rangle &= \mathbf{x}^T(\text{diag}(\mathbf{d}_H) - \mathbf{L}_H)\mathbf{x} &\text{(16a)}\\ &= \mathbf{d}_H^T\mathbf{x} - \sum_{(u,v)\in\mathcal{E}_H} w_{(u,v)}|x_{(u)} - x_{(v)}| &\text{(16b)}\end{aligned}$$

where the equalities stem from the elementary facts that $\alpha \in \{0, 1\} \Leftrightarrow \alpha = \alpha^2$ and $\beta \in \{-1, 0, 1\} \Leftrightarrow |\beta| = \beta^2$.

On substituting (16b) in place of $\langle \mathbf{BP}, \mathbf{PA} \rangle$ in the cost function of (3), we obtain an additional equivalent reformulation of (1). However, when the "hard" combinatorial set $\mathbb{P}^n$ is relaxed to the "soft" set $\mathbb{B}^n$, the equivalence amongst these formulations is again lost and we obtain the following distinct relaxation

$$(\mathbf{R3}) \min_{\mathbf{s}\in\mathbb{B}^n} \left\{ h_4(\mathbf{s}) := \sum_{(u,v)\in\mathcal{E}_H} w_{(u,v)} |s_{(u)} - s_{(v)}| - \mathbf{d}_H^T \mathbf{s} + c \right\} \tag{17}$$

where we have employed the definitions $\mathbf{s} := \text{vec}(\mathbf{S})$, $c := (\|\mathbf{a}\|_2^2 + \|\mathbf{b}\|_2^2)/2$, $\mathbf{a} := \text{vec}(\mathbf{A})$, and $\mathbf{b} := \text{vec}(\mathbf{B})$. To the best of our knowledge, such a relaxation for graph matching has not been considered previously. Note that (**R3**) is a *non-smooth*, convex optimization problem, being the sum of an affine function and a graph total-variation (TV) regularization term.

We now establish that the above relaxation possesses an important property. To begin with, note that the bijective mapping defined by any permutation matrix $\mathbf{P} \in \mathbb{P}^n$ can be equivalently represented as a perfect matching in a complete bipartite graph $\mathcal{G}_C := (\mathcal{L}_C, \mathcal{R}_C, \mathcal{E}_C)$ with $\mathcal{L}_C := \mathcal{V}_A$ and $\mathcal{R}_C := \mathcal{V}_B$ denoting the vertices on the "left" and "right" hand sides of the bipartition respectively, and $\mathcal{E}_C := [n] \times [n]$ denoting the edge set. Let $\mathcal{M} \subset \mathcal{E}_C$ denote the set of perfect matchings in $\mathcal{G}_C$. Then, every $\mathbf{x} = \text{vec}(\mathbf{P})$ corresponds to the indicator vector of a perfect matching $\mathcal{S} \in \mathcal{M}$ in $\mathcal{G}_C$, i.e.,

$$\mathbf{x} = \mathbb{1}_{\mathcal{S}}, \mathcal{S} \text{ is a perfect matching in } \mathcal{G}_C \tag{18}$$

With this observation, (3) can be equivalently reformulated as a subset selection problem of the form

$$\min_{\mathcal{S}\in\mathcal{M}} \left\{ F(\mathcal{S}) := -\mathbb{1}_{\mathcal{S}}^T \mathbf{H} \mathbb{1}_{\mathcal{S}} \right\} \tag{19}$$

Next, we will utilize the following key result regarding the set function $F : 2^{\mathcal{E}_C} \to \mathbb{R}$.

Lemma 2 (Proposition 1, [5])**.** *$F(\mathcal{S})$ is a monotone non-increasing, submodular function.*

Submodular functions [13] are notable for exhibiting a notion of diminishing returns, i.e., for all $\mathcal{A} \subseteq \mathcal{B} \subseteq \mathcal{E}_C$ and all $e \in \mathcal{E}_C \setminus \mathcal{B}$, it holds that

$$f(\mathcal{A} \cup \{e\}) - f(\mathcal{A}) \geq f(\mathcal{B} \cup \{e\}) - f(\mathcal{B}). \tag{20}$$

A remarkable property of submodular functions is that every such function possesses a convex extension [14] that extends its domain from the vertices of the unit hypercube $\{0,1\}^m$ to the entire hypercube $[0,1]^m$, widely known as the Lovász extension. The Lovász extension $f_L : \mathbb{R}^m \to [0,1]^m$ of a submodular function F takes the form of the support function

$$f_L(\mathbf{x}) = \max_{\mathbf{g}\in\mathcal{B}_F} \mathbf{g}^T \mathbf{x} \tag{21}$$

of the set

$$\mathcal{B}_F := \{\mathbf{y} \in \mathbb{R}^m : y(\mathcal{E}_C) = F(\mathcal{E}_C); y(\mathcal{S}) \leq F(\mathcal{S}), \forall\, S \subseteq \mathcal{E}_C\}, \tag{22}$$

which is the *base polytope* associated with F. We point out that the Lovász extension f_L is convex if and only if F is submodular. Additionally, it also admits the following useful interpretation, which is extracted from [15, Section 5.1].

Lemma 3. *Given a subodular function F, define the function*

$$w(\mathbf{x}) := \begin{cases} F(\mathcal{S}), & \forall\, \mathbf{x} = \mathbb{1}_{\mathcal{S}}, \mathcal{S} \subseteq 2^{[m]} \\ +\infty, & \forall\, \mathbf{x} \neq \{0,1\}^m \end{cases} \tag{23}$$

Then, the Fenchel biconjugate of w is exactly the Lovász extension of F.

In other words, the Lovász extension is the convex closure, or the greatest convex under-estimator of the submodular function F (in a certain sense) on the domain $[0,1]^m$.

While the Lovász extension enjoys these important properties, it does not admit an analytical form in general, owing to the (potentially) exponential number of inequalities that characterize the base polytope. However, it can be evaluated at any point $\mathbf{x} \in [0,1]^m$ in log-linear time by utilizing Edmond's greedy algorithm [16]. In contrast, we now establish that for graph matching, the Lovász extension does admit an explicit form.

Theorem 2. *The Lovász extension of the submodular function $F(\mathcal{S}) = -\mathbb{1}_{\mathcal{S}}^T \mathbf{H} \mathbb{1}_{\mathcal{S}}$ is given by*

$$f_L(\mathbf{x}) = \sum_{(u,v)\in\mathcal{E}_H} w_{(u,v)} |x_{(u)} - x_{(v)}| - \mathbf{d}_H^T \mathbf{x}.$$

Proof. We decompose the function $F(\mathcal{S})$ as

$$\begin{aligned} F(\mathcal{S}) &= -\mathbb{1}_{\mathcal{S}}^T (\text{diag}(\mathbf{d}_H) - \mathbf{L}_H) \mathbb{1}_{\mathcal{S}} \\ &= \mathbb{1}_{\mathcal{S}}^T \mathbf{L}_H \mathbb{1}_{\mathcal{S}} - \mathbf{d}_H^T \mathbb{1}_{\mathcal{S}} \\ &= G(\mathcal{S}) + H(\mathcal{S}) \end{aligned} \tag{24}$$

where $G(\mathcal{S}) := \mathbb{1}_{\mathcal{S}}^T \mathbf{L}_H \mathbb{1}_{\mathcal{S}}$ corresponds to the graph-cut function, and is submodular, and $H(\mathcal{S}) := -\mathbf{d}_H^T \mathbb{1}_{\mathcal{S}}$ is a modular (and hence, also submodular) function.

Since the Lovász extension of the sum equals the sum of Lovász extensions [14], it suffices to compute the Lovász extensions g_L and h_L of G and H respectively. It is known [15, p. 165] that the Lovász extension of $G(\mathcal{S})$ is the total-variation function

$$g_L(\mathbf{x}) = \sum_{(u,v)\in\mathcal{E}_H} w_{(u,v)} |x_{(u)} - x_{(v)}|. \tag{25}$$

On the other hand, $H(\mathcal{S})$ is a modular function, whose Lovász extension is simply $h_L(\mathbf{x}) = -\mathbf{d}_H^T \mathbf{x}$.

□

From the above result, we see that the convex problem (**R3**) corresponds to minimizing the Lovász extension of the submodular function $F(.)$ over the set $\mathbb{B}^n$. As a consequence of this observation and Lemma 3, it follows that (**R3**) is the closest convex problem of (1), in a certain sense. At this point, we do not have any information regarding the relationship between the optimal solutions of (**R2**) and (**R3**). However, (**R3**) possesses the advantage of being convex and thus can

be solved optimally in polynomial-time for *every* instance, in contrast to (**R2**), which does not admit a polynomial-time solution in general.

V. Interpretation of L-Relaxation

Using the fact that the Lovász extension for graph matching can be computed explicitly, we provide a simple interpretation of the convex relaxation (**R3**) (from hereon referred to as the *L-relaxation*). To begin with, consider the linear programming (LP) problem

$$\max_{\mathbf{s}\in\mathbb{B}^n} \mathbf{d}_H^T\mathbf{s} \Leftrightarrow \max_{\mathbf{S}\in\mathbb{B}^n} \text{Trace}(\mathbf{d}_A\mathbf{d}_B^T\mathbf{S}) \tag{26}$$

where the equivalence stems from the fact that

$$\begin{aligned}\mathbf{d}_H^T\mathbf{s} = (\mathbf{d}_A \otimes \mathbf{d}_B)^T\text{vec}(\mathbf{S}) &= \text{vec}(\mathbf{d}_B\mathbf{d}_A^T)^T\text{vec}(\mathbf{S}) \\ &= \text{Trace}(\mathbf{d}_A\mathbf{d}_B^T\mathbf{S}).\end{aligned} \tag{27}$$

Note that (26) corresponds to (**R3**) without the total-variation (TV) regularization term. Furthermore, the above LP problem is equivalent to its discrete counterpart

$$\max_{\mathbf{P}\in\mathbb{P}^n} \text{Trace}(\mathbf{d}_A\mathbf{d}_B^T\mathbf{P}) \tag{28}$$

since the set of doubly stochastic matrices $\mathbb{S}^n$ forms the convex-hull of $\mathbb{P}^n$. Consequently, the solution of the relaxed LP problem (26) is guaranteed to be integral and optimal for (28). In turn, (28) is equivalent to

$$\min_{\mathbf{P}\in\mathbb{P}^n} \|\mathbf{d}_B - \mathbf{P}\mathbf{d}_A\|_2^2. \tag{29}$$

By the above chain of equivalences, it follows that the solution of (26) computes the permutation that best aligns the (weighted) degree vectors of the input graphs $\mathcal{G}_A$ and $\mathcal{G}_B$, which is intuitively pleasing, as one expects vertices with similar degrees to correspond to each other.

On adding the TV regularization term to the objective function in (26), we obtain the L-relaxation (in maximization form)

$$\max_{\mathbf{s}\in\mathbb{B}^n}\left\{\mathbf{d}_H^T\mathbf{s} - \sum_{(u,v)\in\mathcal{E}_H} w_{(u,v)}|s_{(u)} - s_{(v)}|\right\} \tag{30}$$

which is *not* equivalent to its discrete counterpart

$$\max_{\mathbf{x}\in\mathbb{P}^n}\left\{\mathbf{d}_H^T\mathbf{x} - \mathbf{x}^T\mathbf{L}_H\mathbf{x}\right\} \tag{31}$$

in general. That being said, solving the relaxed problem (30) still constitutes a meaningful measure of matching, as we now explain.

Note that the the non-negativity of the TV term ensures that the optimal solution of (30) is no larger than that of the LP problem (26), and hence, is a tighter relaxation of (31). This can be attributed to the following reason – (26) seeks to pair-up vertices "individually" on the basis of their (weighted) degrees alone, but does not take into consideration the alignment induced for the one-hop neighbors of each aligned vertex-pair, which is precisely what the TV regularization term in (30) takes into account. To be more specific, the linear term in the objective function of (30) can be expressed as

$$\mathbf{d}_H^T\mathbf{s} = \sum_{u\in\mathcal{V}_H} [d_H]_{(u)}s_{(u)} = \sum_{(i,j)\in\mathcal{V}_A\times\mathcal{V}_B} [d_A]_{(i)}[d_B]_{(j)}S_{(i,j)} \tag{32}$$

where $u := (i-1)n+j$ for a given vertex pair $(i,j) \in \mathcal{V}_A\times\mathcal{V}_B$. Since $\mathbf{d}_A, \mathbf{d}_B, \mathbf{s} \geq \mathbf{0}$ and we are performing maximization, it is evident that a larger "soft" correspondence value $S_{(i,j)}$ is assigned to those pairs of vertices which have large (weighted) degrees and are similar to each other. Meanwhile, for an edge $(u,v) \in \mathcal{E}_H$ (comprising vertex pairs $u = (i,j), v = (k,l)$ in $\mathcal{G}_H$) with weight $w_{(u,v)} = A_{(i,k)}B_{(j,l)}$, each component $w_{(u,v)}|s_{(u)} - s_{(v)}|$ in the TV regularizer enforces smoothness between $S_{(i,j)}$ and $S_{(k,l)}$ (in the absolute value sense), i.e., for a given pair of vertices with a large correspondence score $S_{(i,j)}$, the TV regularizer aims to ensure that the one-hop neighborhoods of the vertex pairs in each graph also have a high correspondence score, with the highest emphasis placed on those vertex- neighbor pairs that induce the largest weighted edge-overlap. This is again intuitively pleasing, since the overall objective of the graph matching problem is to find a correspondence mapping between the vertex-sets of the graphs that maximizes the total weighted edge overlap.

Weighting edges in the L-relaxation: Note that for unweighted, undirected graphs, the weights $\{w_{(u,v)}\}_{(u,v)\in\mathcal{E}_H}$ in the TV regularizer of the Lovász extension possess a nominal value of one, i.e., all edges are assigned equal importance for alignment. However, such a choice can be sub-optimal, as it is completely agnostic to the structure of the graphs $\mathcal{G}_A$ and $\mathcal{G}_B$. Consequently, we prescribe the following two ways of leveraging the structure of the graphs to design non-uniform weights $w_{(u,v)} \in (0,1]$ for each edge in a manner that reflects its importance in aligning a pair of vertices.

Scheme (A): Formally, for a vertex $i \in \mathcal{V}_A$, we define its in-neighborhood as $\bar{\mathcal{N}}(i) := i \cup \{j : (i,j) \in \mathcal{E}_A\}$, which comprises the vertex itself and its one-hop neighborhood. Given an (unweighted) edge $(i,j) \in \mathcal{E}_A$ with vertex in-neighborhoods $\bar{\mathcal{N}}(i)$ and $\bar{\mathcal{N}}(j)$ respectively, we assign it a weight

$$w_{(i,j)} = |\bar{\mathcal{N}}(i) \cap \bar{\mathcal{N}}(j)|/|\bar{\mathcal{N}}(i) \cup \bar{\mathcal{N}}(j)|. \tag{33}$$

Note that the above choice corresponds to the Jaccard similarity of the in-neighborhoods of the vertices connected by an edge. The reasoning behind such a choice is simple: if a pair of vertices connected by an edge have a large number of common one-hop neighbors (i.e., the edge participates in a large number of triangles), then the edge is assigned a higher weight. We also compute weights $w_{(k,l)}$ for the edges $(k,l) \in \mathcal{E}_B$ in a similar manner, and form the weights $w_{(u,v)} = w_{(i,j)}.w_{(k,l)}$, where $u = (i,k)$ and $v = (j,l)$ are vertices in $\mathcal{G}_H$. Thus, connected vertex pairs in $\mathcal{G}_A$ and $\mathcal{G}_B$ exhibiting large one-hop neighborhood overlap will induce higher edge-weights when paired up in $\mathcal{G}_H$, which in turn places greater emphasis on the correspondence values $S_{(i,k)}$ and $S_{(k,l)}$ being similar in the TV regularizer.

Scheme (B): If the pair of graphs being compared are near-bipartite, or admit a hierarchical decomposition of high degree hubs connected with multiple smaller degree spokes, then due to the scarcity of triangles, applying the above edge-weighting scheme is not a meaningful measure of an edge's importance. In such a case, we advocate assigning each edge with a weight that corresponds to the relative degree difference (RDD) of the vertices it connects, i.e., for each $(i,j) \in \mathcal{E}_A$, we set

$$w_{(i,j)} = \left(1 + 2\frac{|[d_A]_{(i)} - [d_A]_{(j)}|}{[d_A]_{(i)} + [d_A]_{(j)}}\right)^{-1} \tag{34}$$

Thus, higher the relative similarity of the degrees of the vertices connecting an edge, the greater is the strength of the edge. We compute edge weights for $\mathcal{G}_B$ in a similar manner, and form edge weights $\{w_{(u,v)}\}_{(u,v)\in\mathcal{E}_H}$ as described previously.

Finally, we point out that this freedom in selecting edge-weights is unique to the L-relaxation; an analogous strategy does not appear to carry over for the relaxations **(R1)** and **(R2)** (simply replacing the adjacencies by the weighted adjacencies does not result in an analogous formulation to **(R3)**). In hindsight, this additional flexibility appears to stem from the particular form of the Lovász relaxation for the graph matching problem.

VI. Algorithms

In this section, we investigate efficient means of solving the L-relaxation (30). We point out that (30) can be equivalently reformulated as a LP problem, albeit at the expense of introducing an additional $|\mathcal{E}_H|$ variables and $2|\mathcal{E}_H|$ linear inequality constraints. In particular, for unweighted graphs [2], we have

$$\begin{aligned}|\mathcal{E}_H| &= (1/2)\mathbf{1}_m^T\mathbf{H}\mathbf{1}_m = (1/2)(\mathbf{1}_n \otimes \mathbf{1}_n)^T(\mathbf{A}\otimes\mathbf{B})(\mathbf{1}_n\otimes\mathbf{1}_n)\\ &= (1/2)(\mathbf{1}_n\otimes\mathbf{1}_n)^T(\mathbf{d}_A\otimes\mathbf{d}_B) = 2|\mathcal{E}_A||\mathcal{E}_B|\end{aligned}$$

which entails introducing an additional $O(n^4)$ variables in the worst-case scenario. Due to the prohibitive computational complexity stemming from such a reformulation, we explore alternative algorithmic approaches that are capable of intelligently exploiting the structure of the problem.

To that end, note that the Lovász extension $f_L(\mathbf{s})$ is non-differentiable, which suggests using a Euclidean projected subgradient algorithm for solving (30). The algorithm requires initialization from a point $\mathbf{s}^0 \in \mathbb{B}^n$ and then proceeds in the following iterative fashion

$$\mathbf{s}^{k+1} = \arg\min_{\mathbf{s}\in\mathbb{B}^n}\left\{(\mathbf{g}^k)^T\mathbf{s} + \frac{1}{\eta_k}\|\mathbf{s}-\mathbf{s}^k\|_2^2\right\}, \forall\, k\in\mathbb{N} \tag{36}$$

where $\mathbf{g}^k \in \partial f_L(\mathbf{s}^k)$ denotes a subgradient of the Lovász extension $f_L(\mathbf{s})$ at the current iterate $\mathbf{s} = \mathbf{s}^k$ and $\eta_k > 0$ is a step-size. Assuming that the subgradients of $f_L(\mathbf{s})$ are bounded, i.e., there exists a $G > 0$ such that

$$\|\mathbf{g}\|_2 \le G, \forall\, g \in \partial f_L(\mathbf{s}), \forall\, \mathbf{s}\in\mathbb{B}^n,$$

[2]The same result is also true for weighted graphs, with $\mathbf{A}$ and $\mathbf{B}$ taken to be the support of the non-zero edges.

using the step-size rule $\eta_k = O(1/(G\sqrt{k}))$ suffices to guarantee convergence to the optimal objective value of (30) at a sublinear-rate of $O(G/\sqrt{k})$ [17, Theorem 3.2]. However, such an algorithm is not appropriate here due to the following reasons.

(a) At first glance, it appears that the convergence rate is independent of the problem dimension m. Unfortunately, this is only true for problems with Euclidean structure whereas in our case, the Lovász extension possesses non-Euclidean structure. In order to see this, we re-express $f_L(\mathbf{s})$ as

$$\begin{aligned}f_L(\mathbf{s}) &= -\mathbf{d}_H^T\mathbf{s} + \sum_{(u,v)\in\mathcal{E}_H} w_{(u,v)}|(\mathbf{e}_{(u)} - \mathbf{e}_{(v)})^T\mathbf{s}|\\ &= -\mathbf{d}_H^T\mathbf{s} + \|\mathbf{W}\mathbf{F}^T\mathbf{s}\|_1\end{aligned} \tag{37}$$

where $\mathbf{W} := \mathrm{diag}(\mathbf{w})$ is a diagonal matrix of the edge weights $\{w_{(u,v)}\}_{(u,v)\in\mathcal{E}_H}$ stacked in a vector $\mathbf{w}$, and $\mathbf{F} \in \{-1,0,+1\}^{m\times|\mathcal{E}_H|}$ is the directed vertex-edge incidence matrix of $\mathcal{G}_H$. Note that a column of $\mathbf{F}$ corresponds to an edge $(u,v)\in\mathcal{E}_H$, and is of the form $\mathbf{f}_{(u,v)} := \mathbf{e}_{(u)} - \mathbf{e}_{(v)}$. From elementary convex analysis [17], it follows that the subdifferential set of $f_L(\mathbf{s})$ at a point $\mathbf{s}\in\mathbb{B}^n$ is given by

$$\begin{aligned}\partial f_L(\mathbf{s}) &:= -\mathbf{d}_H + \mathbf{F}\mathbf{W}\partial\|\mathbf{s}\|_1,\\ &= \{-\mathbf{d}_H + \mathbf{F}\mathbf{W}\mathbf{z}, \|\mathbf{z}\|_\infty \le 1\}.\end{aligned} \tag{38}$$

Next, we derive a bound on norm of the subgradients of $f_L(\mathbf{s})$. Given a subgradient $\mathbf{g}\in\partial f_L(\mathbf{s})$, we obtain the bound

$$\begin{aligned}\|\mathbf{g}\|_\infty &\le \|\mathbf{d}_H\|_\infty + \|\mathbf{F}\mathbf{W}\mathbf{z}\|_\infty\\ &\le d_{\max}(\mathbf{H}) + \|\mathbf{F}\|_\infty\|\mathbf{W}\|_\infty\|\mathbf{z}\|_\infty\\ &\le d_{\max}(\mathbf{H}) + d_{\max}(\mathbf{H})\max(\mathbf{w})\\ &= (1+\max(\mathbf{w}))d_{\max}(\mathbf{H})\\ &= (1+\max(\mathbf{w}))d_{\max}(\mathbf{A})d_{\max}(\mathbf{B}),\end{aligned} \tag{39}$$

where the operators $d_{\max}(.)$ and $\max(.)$ respectively denote the largest degree in a graph and the largest entry in a vector, and in the third step we have utilized the fact that $\|\mathbf{F}\|_\infty = d_{\max}(\mathbf{H})$. Since $\|\mathbf{g}\|_2 \le \sqrt{m}\|\mathbf{g}\|_\infty$, we obtain the result that the "size" of the subgradients (as measured in the standard Euclidean sense) is directly dependent on the problem dimension $n = \sqrt{m}$. Consequently, attempting to solve (30) via a standard subgradient method would result in a convergence rate $O(\sqrt{\frac{m}{k}}) = O(n/\sqrt{k})$ that is *dependent* on the problem dimension, which is undesirable for large problem instances.

(b) Additionally, performing the updates of the subgradient algorithm (36) requires computing Euclidean projections onto the constraint set $\mathbb{B}^n$, i.e., solving a problem of the form

$$\mathbf{s}^{k+1} = \arg\min_{\mathbf{s}\in\mathbb{B}^n}\|\mathbf{s} - (\mathbf{s}^k - \eta_k\mathbf{g}^k)\|_2^2 \tag{40}$$

at every step, which is non-trivial in general. Roughly speaking, this can be attributed to the fact that in high dimensions the polytope $\mathbb{B}^n$ looks "diamond-like" with a

large number of facets, which prevents efficient computation of the Euclidean projection.

In hindsight, these drawbacks stem from the inability of the standard projected subgradient algorithm (36) to adapt to the non-Euclidean geometry of the objective function and constraint set of (30). This motivates us to seek an alternative that is capable of addressing the aforementioned problems. We now argue that Nesterov's Dual Averaging algorithm (NDA) [6] is well-suited for such a task.

NDA is an iterative first-order algorithm that starts from an initial point $\mathbf{s}^0 \in \mathbb{B}^n$ and then proceeds as follows

$$\mathbf{s}^{k+1} = \arg\min_{\mathbf{s}\in\mathbb{B}^n} \left\{ (\bar{\mathbf{g}}^k)^T\mathbf{s} + \frac{1}{\eta_k}\Psi(\mathbf{s}) \right\}, \forall\, k \in \mathbb{N}. \tag{41}$$

The term $\bar{\mathbf{g}}^k = \sum_{t=0}^{k-1} \mathbf{g}^t$ is the sum of all past subgradients and $\Psi(\mathbf{s})$ is a "potential" function (to be specified shortly) that is strongly convex on $\mathbb{B}^n$. The update (41) admits the following interpretation: at each step we seek to minimize an averaged first-order approximation to the Lovász extension $f_L(.)$ while the potential function serves as regularization that curtails excessive variation in the generated iterates.

The potential function $\Psi(.)$ is typically chosen to be attuned to the non-Euclidean geometry of the problem. In our case, we select $\Psi(\mathbf{s})$ to be the negative entropy function, i.e.,

$$\Psi(\mathbf{s}) = \sum_{u\in\mathcal{V}_H} s_{(u)}(\log s_{(u)} - 1). \tag{42}$$

For this choice, we can establish the following properties of NDA for our problem.

Near-dimension independent convergence rate: Define the parameter $\beta = (1+\max(\mathbf{w}))d_{\max}(\mathbf{A})d_{\max}(\mathbf{B})$. Then, for the choice of step-size $\frac{1}{\beta}\sqrt{\frac{\log n}{2k}}$, NDA attains a convergence rate of $O(\sqrt{\frac{n\log n}{k}})$, which, for large problem instances, represents a significant improvement over the $O(n/\sqrt{k})$ convergence rate of the standard projected subgradient algorithm. The proof is omitted due to space limitations.

Near-linear time projections: The NDA update (41) can be further simplified to reveal a simple approach for computing non-Euclidean projections onto the polytope $\mathbb{B}^n$. Upon defining the vector $\mathbf{r}^k := \exp(-\eta_k\bar{\mathbf{g}}^k)$, we can rewrite the update (41) as follows

$$\begin{aligned}
\mathbf{s}^{k+1} &= \arg\min_{\mathbf{s}\in\mathbb{B}^n} \sum_{u\in\mathcal{V}_H} \left\{ -s_{(u)}\log r^k_{(u)} + s_{(u)}(\log s_{(u)} - 1) \right\}, \\
&= \arg\min_{\mathbf{s}\in\mathbb{B}^n} \sum_{u\in\mathcal{V}_H} \left\{ s_{(u)}\log\left(\frac{s_{(u)}}{r^k_{(u)}}\right) - s_{(u)} + r^k_{(u)} \right\}, \\
&= \arg\min_{\mathbf{s}\in\mathbb{B}^n} D_{\mathrm{KL}}(\mathbf{s}|\mathbf{r}^k), \forall\, k \in \mathbb{N}.
\end{aligned} \tag{43}$$

where $D_{\mathrm{KL}}(.|.)$ denotes the (unnormalized) Kullback-Leibler (KL) divergence between a pair of vectors. In other words, the above update corresponds to computing the KL projection of the exponentiated vector $\mathbf{r}^k$ onto the polytope $\mathbf{s} \in \mathbb{B}^n$. While the problem does not admit a closed-form solution, it can be efficiently solved using Sinkhorn's matrix balancing algorithm [18] in $O(m\log m)$ time [19], [20] (see Appendix A for details).

From the above observations, we conclude that NDA with negative entropy as the choice of the potential function $\Psi(.)$ can indeed adapt to the non-Euclidean geometry of the L-relaxation. In practice, we employ a variant of the NDA algorithm described in [6] which utilizes a weighted sum of the past subgradients and a more aggressive step-size strategy. The method is summarized in Algorithm 1, and enjoys the same theoretical properties of the standard NDA algorithm, but exhibits significantly better performance in practice.

Algorithm 1: Nesterov's **W**eighted **D**ual **A**veraging

1 **Input:** $f_L(.)$, initial point $\mathbf{s}^0 \in \mathbb{B}^n$, step-size parameter $c > 0$, exit tolerance $\delta > 0$
2 **Output:** A solution $\hat{\mathbf{s}}$ to the L-relaxation
3 **Initialize:** $k \leftarrow 0, \hat{\gamma}_0 = 1, \bar{\mathbf{g}}^0 = \mathbf{0}$
4 **repeat**
5 Get subgradient $\mathbf{g}^k = -\mathbf{d}_H + \mathbf{F}\mathbf{W}\mathrm{sign}(\mathbf{W}\mathbf{F}^T\mathbf{s}^k)$
6 Set $\bar{\mathbf{g}}^{k+1} = \bar{\mathbf{g}}^k + \mathbf{g}^k/\|\mathbf{g}^k\|_\infty$, $\hat{\gamma}_{k+1} = \sum_{i=0}^{k} 1/\hat{\gamma}_i$
7 Set $\gamma_{k+1} = c\hat{\gamma}_{k+1}$
8 Compute $\mathbf{s}^{k+1} = \textsc{Sinkhorn}(-\gamma_{k+1}\bar{\mathbf{g}}^{k+1})$
9 Set $\bar{\mathbf{s}}^{k+1} = \frac{1}{k+1}\sum_{i=0}^{k}\mathbf{s}^i$
10 $k \leftarrow k+1$
11 **until** $f(\bar{\mathbf{s}}^{k-1}) - f(\bar{\mathbf{s}}^k) \le \delta$
12 **Return:** $\hat{\mathbf{s}} \leftarrow \mathbf{s}^k$

Scalability: We now discuss an important issue regarding the scalability of the proposed method. The key subroutine at the heart of Algorithm 1 is the computation of the subgradient of the TV term during each step of NDA. However, doing so seemingly requires first forming the vertex-edge incidence matrix $\mathbf{F}$ and the edge-weight vector $\mathbf{w}$ of $\mathcal{G}_H$, which in turn requires instantiating the Kronecker product $\mathbf{H} = \mathbf{A}\otimes\mathbf{B}$. Owing to the substantial space-time complexity incurred in performing such an operation, we investigate an alternative algorithmic approach which bypasses such a step altogether.

To that end, we note that the L-relaxation can be equivalently expressed as

$$\min_{\mathbf{s}\in\mathbb{B}^n} \left\{ \frac{1}{|\mathcal{E}_H|}\left(-\mathbf{d}_H^T\mathbf{s} + \sum_{(u,v)\in\mathcal{E}_H} w_{(u,v)}|s_{(u)} - s_{(v)}| \right) \right\} \tag{44}$$

In this form, the TV term corresponds to an average of a finite sum of "atomic" TV functions

$$f_{(u,v)}(\mathbf{s}) := w_{(u,v)}|s_{(u)} - s_{(v)}| = |w_{(u,v)}\mathbf{f}_{(u,v)}^T\mathbf{s}|;$$

one associated with each edge $(u,v) \in \mathcal{E}_H$. Hence, if we had the ability to sample edges uniformly at random from $\mathcal{E}_H$, we could consider forming a *stochastic* subgradient

$$\mathbf{g}_{(u,v)} := -(1/|\mathcal{E}_H|)\mathbf{d}_H + w_{(u,v)}\mathrm{sign}(w_{(u,v)}\mathbf{f}_{(u,v)}^T\mathbf{s})\mathbf{f}_{(u,v)} \tag{45}$$

and employ it in the NDA algorithm in place of the batch subgradient (see line 5, Algorithm 1). In addition to being computationally lightweight, the stochastic subgradient de-

fined above is an unbiased estimate of the batch subgradient, i.e., we have

$$\mathbb{E}_{p\sim(u,v)}[\mathbf{g}_{(u,v)}] \in \partial f_L\left(\frac{1}{|\mathcal{E}_H|}\mathbf{s}\right),$$

where the probability of sampling an edge $(u,v) \in \mathcal{E}_H$ is $1/|\mathcal{E}_H| = 1/(2|\mathcal{E}_A||\mathcal{E}_B|)$. Furthermore, it has been shown that using unbiased stochastic subgradients in place of batch subgradients in NDA results in a primal convergence rate of $O(1/\sqrt{k})$ (in expectation) [21]. Hence, the (expected) convergence rate of stochastic-NDA does not degrade compared to its batch version, while the per-iteration complexity is substantially improved owing to the simplicity of the update (45). Overall, using the stochastic variant of NDA enables us to affect a very favorable performance-complexity trade-off.

That being said, in order to sample edges uniformly at random from $\mathcal{E}_H$, it still appears necessary to form $\mathcal{G}_H$ in the first place by instantiating the Kronecker product $\mathbf{A}\otimes\mathbf{B}$. In order to obviate this need, we make the following key observation: the Kronecker product of a pair of edges $(i,j) \in \mathcal{E}_A$ and $(k,l) \in \mathcal{E}_B$ generates two distinct edges $((i,k),(j,l))$ and $((i,l),(j,k))$ in $\mathcal{E}_H$ (the factor 2 stems from the fact that the graphs $\mathcal{G}_A$ and $\mathcal{G}_B$ are undirected), and the full Kronecker product $\mathbf{A}\otimes\mathbf{B}$ generates all such $2|\mathcal{E}_A||\mathcal{E}_B|$ edges in $\mathcal{G}_H$. Exploiting the fact that each edge in $\mathcal{E}_H$ arises from the Kronecker product of a pair of unique edges in $\mathcal{E}_A$ and $\mathcal{E}_B$, we propose the following scheme for sampling edges from $\mathcal{E}_H$. First, we sample an edge $(i,j) \in \mathcal{E}_A$ uniformly at random with probability (w.p.) $1/|\mathcal{E}_A|$, and independently, we draw an edge $(k,l) \in \mathcal{E}_B$ uniformly at random w.p. $1/|\mathcal{E}_B|$. Then, we perform an independent coin-toss (with an unbiased coin) to decide whether to generate the edge $((i,k),(j,l))$ (heads), i.e., or $((i,l),(j,k))$ (tails). Let $C = \{0,1\}$ be a Bernoulli random variable that denotes the outcome of the coin-toss, with $C = 1$ denoting heads. Then, the prescribed sampling scheme generates edges in $\mathcal{G}_H$ with probability

$$\begin{aligned}\Pr(((i,k),(j,l)) \in \mathcal{E}_H) &= \Pr((i,j) \in \mathcal{E}_A).\Pr((k,l) \in \mathcal{E}_B).\\ &\quad \Pr(C=1)\\ &= 1/(2|\mathcal{E}_A||\mathcal{E}_B|),\end{aligned} \tag{46}$$

which is the desired sampling probability. This shows that no Kronecker products are required in order to sample edges uniformly at random from $\mathcal{E}_H$. Thus, the scheme provides an efficient means of constructing the stochastic subgradient (45).

Additionally, we can also construct a mini-batch stochastic subgradient by independently sampling $K < 2|\mathcal{E}_A||\mathcal{E}_B|$ edges (with replacement) from each of the graphs $\mathcal{G}_A$ and $\mathcal{G}_B$ followed by drawing K edges in $\mathcal{G}_H$ according to the scheme described above. Let $\mathcal{E}_\mathcal{K} \subset \mathcal{E}_H$ denote the subset of K sampled edges. Then, the resulting mini-batch stochastic subgradient takes the form

$$\mathbf{g}_m = -(1/|\mathcal{E}_H|)\mathbf{d}_H + \frac{1}{K}\bar{\mathbf{F}}_\mathcal{K}\bar{\mathbf{W}}_\mathcal{K}\text{sign}(\bar{\mathbf{W}}_\mathcal{K}\bar{\mathbf{F}}_\mathcal{K}^T\mathbf{s}), \tag{47}$$

where $\bar{\mathbf{F}}_\mathcal{K} \in \{-1,0,1\}^{m\times K}$ is a matrix with the vectors $\{\mathbf{f}_{(u,v)}\}_{k\in\mathcal{K}}$ as its columns and $\bar{\mathbf{W}}_\mathcal{K}$ is a diagonal matrix of the weights of the K sampled edges $\mathcal{K}$. Note that in expectation

$$\mathbb{E}_{p\sim(u,v)}(\mathbf{g}_m) \in \partial f_L\left(\frac{1}{|\mathcal{E}_H|}\mathbf{s}\right). \tag{48}$$

Hence, the mini-batch stochastic subgradient (47) is also an unbiased estimate of the batch subgradient, and with lower variance compared to (45). In our experiments, when applying the L-relaxation for large problem instances, we employ the above mini-batching scheme (i.e., we replace the subgradient by its mini-batch approximation (47) in (line 5, Algorithm 1)).

VII. Experiments

In this section, we test the performance of our algorithms for solving the L-relaxtion against the prevailing relaxation algorithms on real-world graphs.

Baselines: We employ the classical Frank-Wolfe (FW) algorithm [22] for solving the convex QP problem **(R1)** and the non-convex QP problem **(R2)**. We designate these algorithms as **FW-C** and **FW-NC**, respectively. At a high level, both algorithms successively linearize their respective quadratic cost functions to solve a sequence of linear assignment problems. Since **(R1)** is convex, employing the step-size sequence $\eta_k = 2/(k+2)$, guarantees a primal convergence rate (to the optimal value of **(R1)**) of $O(1/k)$ [23]. On the other hand, for **FW-NC** we employ an exact line-search procedure to select the step-size, (see details in [10]). As the problem is non-convex, the algorithm is only guaranteed to converge to a stationary point of **(R2)** [10]. In practice, the solution of **(R1)** obtained via FW-C is used as initialization for FW-NC [11]. A drawback of employing the Frank-Wolfe framework is the high complexity incurred in solving a linear assignment problem at each step, which is $O(n^3)$ [24] in the worst-case.

Specific details regarding implementation of the algorithms are given below.

L-relaxation: For all datasets in this section (which are unweighted graphs), we added edge-weights based on the Jaccard similarity of the vertex in-neighborhoods, as it yielded better empirical performance compared to the RDD metric. However, for the synthetic example depicted in Figure 1, which has no triangles, we used the RDD-based weighting scheme. In order to solve the L-relaxation, we employed Nesterov's Weighted Dual Averaging algorithm (**NWDA**) described in Algorithm 1. For small datasets where instantiating and storing the Kronecker product matrix $\mathbf{H} = \mathbf{A}\otimes\mathbf{B}$ incurs a light computational and memory footprint, we use the exact algorithm (without stochastic approximation). In these cases, we ran the algorithm for a maximum of 2000 iterations, and set the step-size parameter $c = 0.175$ (determined via trial-and-error). On the other hand, for larger datasets where instantiating $\mathbf{H}$ is not possible, we utilized the stochastic variant of **NWDA**, with the size of the mini-batch stochastic subgradient K set to be $\min\{|\mathcal{E}_A|,|\mathcal{E}_B|\}$, and the step-size parameter $c = 0.25$. The maximum number of iterations was again set to be 2000. The Sinkhorn matrix balancing sub-routine used within the

NWDA algorithm (for both the exact and stochastic variants) was run until an exit tolerance of $\delta = 1e-4$ was achieved, or a maximum of 250 iterations were reached.

Franke-Wolfe based relaxations: For **FW-C**, we set a maximum iteration counter of 200, as it enjoys a primal convergence rate that is an order of magnitude greater compared to **NWDA**. Meanwhile, the **FW-NC** algorithm is run for only $1-2$ iterations as a means of further refining the soft-correspondence matrices obtained by both **NWDA** and **FW-C**. We used the Jonker-Volgenant algorithm [25] as a sub-routine for solving the linear assignment sub-problems arising in **FW-C** and **FW-NC**, as it is empirically faster compared to the classical Hungarian algorithm [24].

Experimental setup: Given a graph $\mathcal{G}_A$, we first retrieve the largest connected component and then remove all weights and self-loops. What remains is a simple, unweighted, undirected graph. Next, we randomly perturb $\mathcal{G}_A$ by adding new edges with probability p_e to create a "noisy" counterpart graph $\mathcal{G}_{\bar{A}}$, i.e., after generating a random Erdos-Renyi graph with edge-density p_e and adjacency matrix $\mathbf{Q}$, we create the graph $\mathcal{G}_{\bar{A}}$ with adjacency matrix $\bar{\mathbf{A}} = \mathbf{A} + (\mathbf{1}_{n\times n} - \mathbf{A}) * \mathbf{Q}$ (the operator "$*$" denotes the Hadamard (element-wise) product). In the final step, we generate the graph $\mathcal{G}_B$ with adjacency matrix $\mathbf{B} = \mathbf{P}_* \bar{\mathbf{A}} \mathbf{P}_*^T$, where $\mathbf{P}_*$ is a randomly generated permutation matrix. Note that $\mathcal{G}_B$ corresponds to a noisy, permuted version of $\mathcal{G}_A$ whose edge-set $\mathcal{E}_B$ is a superset of $\mathcal{E}_A$. Thus, our objective is to find a permutation that correctly aligns $\mathcal{E}_A$ with its counterpart in $\mathcal{E}_B$. To make the setup challenging, we set the noise level p_e to generate, on average, 20% extra edges in $\mathcal{G}_B$, i.e., we set $p_e = 0.2|\mathcal{E}_A|/\binom{n}{2}$. It is common [11] to assess the performance of soft assignment algorithms by "hardening" their output via linear assignment to obtain a permutation matrix and employing the edge-correctness metric, which is the ratio of the number of edge overlaps induced by the resulting hard assignment and the number of edges in $\mathcal{G}_A$ (the maximum possible number of edges that can be overlapped). The higher the metric, the better the performance of the algorithm, with a maximum possible value of 1 (indicating perfect edge alignment) and a minimum value of 0. Note however, per our earlier discussion, that soft assignment gives a lot more information regarding partial isomorphisms in the graph. We report our results after averaging over 30 Monte-Carlo realizations of $\mathcal{G}_B$.

Datasets: The list of datasets used together with a summary of their descriptions can be found in Table I. The datasets were retrieved from standard repositories [26], [27], with each dataset corresponding to an undirected graph representing relationships amongst different entities. As the first 3 datasets are small in size, we explicitly compute the Kronecker product matrix $\mathbf{H}$ and apply the exact **NWDA** algorithm for solving the L-relaxation. For the remaining datasets, the computational and memory footprint of instantiating $\mathbf{H}$ is too large and we resort to the stochastic **NWDA** algorithm. Additionally, we add an extra baseline for comparison: Umeyama's classical spectral alignment algorithm [28], which utilizes the eigenvectors of the graphs $\mathcal{G}_A$ and $\mathcal{G}_B$ to directly compute a hard alignment.

TABLE I: Summary of networks

Graph	Vertices	Edges	Type
TERRORIST	64	243	Human interaction
LES MISERABLES	77	254	Literary co-occurence
POLBOOKS	105	441	Co-purchase
POLBLOGS	1,224	16,717	Social
CHAMELEON	2,277	31,356	Hyperlinks
FLIGHTS	2,939	15,677	Transportation
JAPANESEBOOK	3,177	7,988	Word adjacency

TABLE II: Average edge-correctness (with standard deviation)

Graph	NWDA	FW-C	SPECTRAL
TERRORIST	$\mathbf{0.59 \pm 0.05}$	0.42 ± 0.05	0.39 ± 0.06
LES MISERABLES	$\mathbf{0.57 \pm 0.04}$	0.32 ± 0.05	0.40 ± 0.05
POLBOOKS	$\mathbf{0.31 \pm 0.04}$	0.18 ± 0.02	0.21 ± 0.03

TABLE III: Average edge-correctness (with standard deviation)

Graph	FW-NC + NWDA	FW-NC + FW-C
TERRORIST	$\mathbf{0.84 \pm 0.03}$	0.70 ± 0.05
LES MISERABLES	$\mathbf{0.87 \pm 0.04}$	0.77 ± 0.05
POLBOOKS	$\mathbf{0.79 \pm 0.09}$	0.28 ± 0.05

Results: For the first 3 datasets, we report the performance of all methods in terms of average edge-correctness (with standard deviation) in Tables II and III, with the former showcasing the efficacy of the spectral alignment algorithm, **NWDA** and **FW-C** in computing a hard alignment, while the latter displays the results when the soft-correspondences computed by **NWDA** and **FW-C** are each refined by 2 steps of **FW-NC** followed by a final hard alignment step. It is evident that both convex relaxation algorithms perform well relative to the spectral method in the presence of high noise, even though they are not directly geared towards hard alignment. In particular, the solution computed by NWDA for the L-relaxation is the best among all methods across all datasets. Furthermore, from Table III, it can be seen that the when the soft-correspondence computed by **NWDA** is used to initialize **FW-NC**, simply 2 iterations suffice to obtain a very high quality alignment, which is better than that obtained by executing 2 steps of **FW-NC** with the initialization switched to the solution of **FW-C**. To conclude, our experiments on these small instances (where all algorithms can be implemented with low memory and computational footprint) reveal that the L-relaxation outperforms the prevailing convex relaxation **(R1)** both in terms of computing hard alignments and serving as a better initialization for **FW-NC**. In fact, our results indicate that employing the L-relaxation for hard matching can even outperform a dedicated algorithm for hard matching.

We now report the results for hard matching on the remaining datasets in table IV. Due to the larger scale of the problem, we omit the **FW-C** algorithm due to the high per-iteration complexity of solving a linear assignment problem at each step, whereas for the L-relaxation, we apply the stochastic variant of NWDA, which we dub as **S-NWDA**. As

TABLE IV: Average edge-correctness (with standard deviation) for larger datasets

Graph	S-NWDA	S-NWDA + FW-C	SPECTRAL
POLBLOGS	0.18 ± 0.002	$\mathbf{0.84 \pm 0.005}$	0.29 ± 0.002
CHAMELEON	0.20 ± 0.003	$\mathbf{0.84 \pm 0.005}$	0.50 ± 0.01
FLIGHTS	0.12 ± 0.002	$\mathbf{0.70 \pm 0.002}$	0.56 ± 0.001
JAPANESEBOOK	0.16 ± 0.006	$\mathbf{0.75 \pm 0.002}$	0.33 ± 0.01

a consequence of the stochastic nature of **S-NWDA**, we do not obtain a high accuracy solution for the L-relaxation, and this is evident in terms of the quality of the obtained hard alignment. However, it can still serve as a high quality initialization for a single iteration of **FW-NC**, which results in a very high quality solution, vastly outperforming the spectral alignment method.

VIII. CONCLUSION

In this paper, we considered an alternative take on soft graph matching as a means of revealing the classes of topologically invariant vertices. Adopting this viewpoint, we proposed a new convex relaxation for graph matching, which we established can be viewed as minimizing the closest convex extension of the combinatorial graph matching objective (in a certain sense) over the set of doubly-stochastic matrices. Additionally, we proposed an efficient first-order algorithm that can adapt to the non-Euclidean geometry of the problem, and described its stochastic extension for scaling up to larger problem instances. Our experiments on real-world data revealed the very favorable performance of our algorithm.

REFERENCES

[1] F. Emmert-Streib, M. Dehmer, and Y. Shi, "Fifty years of graph matching, network alignment and network comparison", *Inform. Scien.*, vol. 346, pp. 180–197, June 2016.

[2] J. Yan, X. C. Yin, W. Lin, C. Deng, H. Zha, and X. Yang, "A short survey of recent advances in graph matching", in *Proc. of ACM Int. Conf. on Mult. Retr.*, pp. 167-174, June 2016, New York City, NY.

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

[4] 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.

[5] A. Konar, and N. D. Sidiropoulos, "Iterative graph alignment via supermodular approximation", *Proc. of IEEE Int. Conf. Data Mining*, pp. 1162–1167, Nov. 2019, Beijing, China.

[6] Y. Nesterov, "Primal-dual subgradient methods for convex problems", *Mathem. Programm.*, vol. 120, no. 1, pp. 221-259, Aug. 2009.

[7] 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.

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

[9] Y. Aflalo, A. Bronstein, and R. Kimmel, "On convex relaxation of graph isomorphism", in *Proc. of Nat. Academ. of Sciences*, vol. 112, no. 10, pp. 2942–2947, Mar. 2015.

[10] 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.

[11] V. Lyzinski, D. Fishkind, M. Fiori, J. Vogelstein, C. Priebe, and G. Sapiro, "Graph matching: Relax at your own risk", *IEEE Trans. Pattern Analys. Mach. Intell.*, vol. 38, no. 1, pp. 60—73, Jan. 2016.

[12] J. Bento, and S. Ioannidis, "A family of tractable graph distances", in *Proc. of SIAM Int. Conf. Data Mining*, pp. 333–341, May 2018, San Diego, CA.

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

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

[15] 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.

[16] J. Edmonds, "Submodular functions, matroids and certain polyhedra", in *Combinatorial structures and their applications*, (G. Goos, J. Hartmanis, and J. van Leeuwen, eds.), vol. 11, Springer, 1970.

[17] S. Bubeck, "Convex optimization: Algorithms and complexity," *Found. Trends Mach. Learn.*, vol. 8, no. 3-4, pp. 231—357, 2015.

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

[19] M. Cuturi, "Sinkhorn distances: Lightspeed computation of optimal transport", *Proc. Neural. Inf. Process. Syst.*, pp. 2292–2300, Lake Tahoe, CA, Dec. 2013.

[20] J. Altschuler, J. Weed, and P. Rigollet, "Near-linear time approximation algorithms for optimal transport via Sinkhorn iteration", *Proc. Neural. Inf. Process. Syst.*, pp. 1964–1974, Long Beach, CA, Dec. 2017.

[21] L. Xiao, "Dual averaging methods for regularized stochastic learning and online optimization", *J. of Mach. Learn. Res.*. vol. 11, no. 88, pp. 2543-2596, Oct. 2010.

[22] M. Frank, and P. Wolfe, "An algorithm for quadratic programming", *Naval Res. Logist. Quart.*, vol. 3, no. 1-2, pp. 95–110, Mar. 1956.

[23] M. Jaggi, "Revisiting Frank-Wolfe: Projection-free sparse convex optimization", in *Proc. ICML*, pp. 427–435, Atlanta, GA, USA, June 2013.

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

[25] 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.

[26] J. Leskovec, and A. Krevl, SNAP Datasets: Stanford Large Network Dataset Collection, 2015. Available at http://snap.stanford.edu/data

[27] J. Kunegis, "Konect: The Koblenz network collection," *Proc. of WWW*, pp. 1343–1350, May 2013, Rio de Janeiro, Brazil.

[28] 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.

APPENDIX A
SINKHORN MATRIX BALANCING

Note that the projection problem (43) is strongly convex on $\mathbb{B}^n$, and hence possesses a unique solution $\mathbf{S}^*$. The KKT conditions of (43) provides the following characterization of the optimal solution $\mathbf{S}^* = \text{diag}(\mathbf{u})\mathbf{R}^k\text{diag}(\mathbf{v})$, where $\mathbf{R}^k \in \mathbb{R}_+^{n\times n}$ is the matricized version of the vector $\mathbf{r}^k$, and $\mathbf{u}, \mathbf{v} \in \mathbb{R}_+^n$ are non-negative scaling vectors to be determined. Since $\mathbf{S}^*$ must satisfy the constraints of (43), we obtain the equations

$$\text{diag}(\mathbf{u})\mathbf{R}^k\text{diag}(\mathbf{v})\mathbf{1} = \mathbf{1}, \text{diag}(\mathbf{v})(\mathbf{R}^k)^T\text{diag}(\mathbf{u})\mathbf{1} = \mathbf{1}. \quad (49)$$

This motivates using the following alternating optimization updates for $\mathbf{u}$ and $\mathbf{v}$,

$$\mathbf{u}^{t+1} = \mathbf{1}/\mathbf{R}^k\mathbf{v}^t, \quad \mathbf{v}^{t+1} = \mathbf{1}/(\mathbf{R}^k)^T\mathbf{u}^{t+1}, \forall\, t \in \mathbb{N} \quad (50)$$

where the "/" operator denotes element-wise division. The algorithm was originally proposed in [18] and only requires matrix-vector multiplications at each iteration. It has been shown [20] that $O(m \log m)$ alternating iterations suffice to converge to a near-optimal solution, and it has been empirically observed that the convergence rate is often much faster [19].