Determinant Regularization for Gradient-Efficient Graph Matching

Tianshu Yu[†], Junchi Yan[‡], Baoxin Li[†]
[†]Arizona State University, [‡]Shanghai Jiao Tong University
{tianshuy,baoxin.li}@asu.edu yanjunchi@sjtu.edu.cn

Abstract

Graph matching refers to finding vertex correspondence for a pair of graphs, which plays a fundamental role in many vision and learning related tasks. Directly applying gradient-based continuous optimization on graph matching can be attractive for its simplicity but calls for effective ways of converting the continuous solution to the discrete one under the matching constraint. In this paper, we show a novel regularization technique with the tool of determinant analysis on the matching matrix which is relaxed into continuous domain with gradient based optimization. Meanwhile we present a theoretical study on the property of our relaxation technique. Our paper strikes an attempt to understand the geometric properties of different regularization techniques and the gradient behavior during the optimization. We show that the proposed regularization is more gradient-efficient than traditional ones during early update stages. The analysis will also bring about insights for other problems under bijection constraints. The algorithm procedure is simple and empirical results on public benchmark show its effectiveness on both synthetic and real-world data.

1. Introduction

Compared with vector-like data, graph is a more versatile representation for many real-world problems whereby the relational structures are involved. One fundamental task in processing graph data is graph matching (GM), which involves establishing vertex correspondences among a pair or multiple graphs. Unlike point alignment that considers only unary (vertex) feature similarity, in general, graph matching establishes the vertex correspondence by considering both unary and higher-order namely hyperedge (typically second-order i.e. edge) information. Due to the introduction of higher-order information, graph matching becomes generally NP-hard. This is in contrast to point matching (linear assignment) which can be solved to the global optimal using the Hungarian method [29] in polynomial time regarding with the number of vertex.

For its robustness against local noise and deformation, graph matching has been widely used in the analysis of images [12], graphics [19], and genome [46], etc. Related approximate techniques have also been developed for cross social network matching [10]. In all these domains, graph matching serves as a building block for downstream applications such as image recognition, graphics clustering, whereby the similarity based on aligned structure is utilized.

Over the decades, extensive works have been developed on graph matching. Traditionally, most methods are focused on pairwise graph matching, i.e., each time only two graphs are involved for matching. Due to its NP-hardness, most methods seek approximation techniques to pursuit the trade-off between accuracy and efficiency. A popular conversion treatment from continuous to discrete solution is applying greedy algorithm or Hungarian method as projection. However such a conversion is likely to bring about arbitrary accuracy degeneration to the final solution.

Recently, a few regularizers have been developed and become an important way to graduated discretization along the solution path. Examples include entropy [35], factorization based convex-concave relaxation [50], and ℓ_2 norm [18]). However, there still lacks a clear investigation on the gradient behavior during the optimization: how does the regularized gradient impact the solution path and what can we do to improve?

This work is focused on investigating and improving the effectiveness of gradient provided by regularization, by providing more reliable gradient direction along the continuous solution path (being a continuation method). We also develop a new regularization technique combined with the simple gradient based continuous optimization. It explores the determinant of the matching matrix which is relaxed in the continuous domain, and achieves superior performance among several gradient-based solvers, while existing regularized methods [16, 35, 18] often perform less competitive and also are algorithmically more complex. The main contributions of the paper are:

i) To enable gradient-efficient continuation optimization for graph matching, we propose a novel graduated discretization technique. Specifically a determinant regulariza-

tion technique is devised on the matching solution. We analytically show the geometric property of our method compared to existing regularizers;

ii) We develop two types of sub-gradient updating rules to address the issue of irregular solution matrix, which have been proved effective and moderately efficient;

iii) Our approach shows promising experimental results on public benchmarks. Thanks to the clear theoretical foundation, the algorithm procedure is in general simple and insensitive to hyperparameters.

Notations are used as follows. Bold lower case $\mathbf{x}$ and upper case $\mathbf{X}$ refer to vector and matrix, respectively. While $\det(\cdot)$ and $\mathrm{sign}(\cdot)$ calculate matrix determinant and sign of a real value, respectively, and $\mathrm{diag}(\cdot)$ maps a vector to a diagonal matrix, and vice versa. $\mathbf{1}$ is a vector with all 1.

2. Related Works

We discuss two fundamental aspects for graph matching.

2.1. Objective Modeling

Graph matching aims to establish the node correspondence to maximize the affinity, which is in general defined as the summation of vertex-to-vertex and edge-to-edge affinity values. While some prior works relax graph matching with linear programming [37, 48, 34], we mainly focus on quadratic relaxation in this paper. In the literature, there are mainly two settings considered. One explores the affinity information confined within the second-order ones, which leads to the classic QAP formulation that has been widely used in graph matching methods [22, 23, 8, 24]:

$$J(\mathbf{x}) = \mathbf{x}^\top \mathbf{K}\mathbf{x} \tag{1}$$
$$\mathbf{X}\mathbf{1}_{n_2} = \mathbf{1}_{n_1}, \mathbf{X}^\top \mathbf{1}_{n_1} \leq \mathbf{1}_{n_2}, \mathbf{X} \in \{0,1\}^{n_1 \times n_2}$$

where $\mathbf{x} = \mathrm{vec}(\mathbf{X})$ is the column-wise vectorized version of matrix $\mathbf{X}$. The constraints assume each vertex in one graph shall find their one-to-one node correspondence in the other.

There is also an emerging line of works [47, 5, 11, 42, 30] resorting to higher-order affinity by tensor representation, whereby three or more nodes are grouped as a tuple and the tensor element stores the similarity between each pair of tuples. A widely used tensor based objective is [21]:

$$J(\mathbf{x}) = \mathbf{H} \otimes_1 \mathbf{x} \cdots \otimes_p \mathbf{x} \tag{2}$$

where $\mathbf{H}$ is the affinity tensor and p is the affinity order. Methods in this setting are also called hypergraph matching methods, which can increase the matching accuracy at the cost of notable complexity increase in both time and space.

Differing from using predefined affinity parameters, recent studies also aim to learn the parameters in either unsupervised or supervised way [24, 4, 25]. For instance, the approach proposed in [6] learns the graph structure model for a specific category that is assumed to contain similar graph samples for matching. Recently deep neural networks [45, 38] are adopted for learning graph matching.

2.2. Optimization Methods

Many works resort to continuous relaxation to circumvent the more challenging discrete optimization problem. The classic work [22, 9] devises a spectral relaxations on the matching matrix, whereby the matching constraint is loosened by $\|\mathbf{x}\|_2 = 1$ for efficient computing. Some other works [16, 22, 23] resort to doubly-stochastic relaxation on the matching matrix, and the solution domain is relaxed to:

$$\mathbf{X}\mathbf{1}_{n_2} = \mathbf{1}_{n_1}, \mathbf{X}^\top \mathbf{1}_{n_1} \leq \mathbf{1}_{n_2}, \mathbf{X} \in [0,1]^{n_1 \times n_2}$$

There are efforts adopting semidefinite-programming (SDP) [36, 31], since SDP can achieve remarkably tight relaxations. The relaxation model can be written as [31]:

$$\min_{\mathbf{Y}} \mathrm{Tr}(\mathbf{Q}\mathbf{Y}) \quad \text{s.t.} \quad \mathbf{Y} \succeq \mathbf{0}, \quad \mathrm{Tr}(\mathbf{K}_i \mathbf{Y}) = c_i$$

where $\mathbf{Y} = \mathbf{x}\mathbf{x}^\top$ and constraints are defined by a number of $\mathbf{K}_i$ and c_i. The problem can be solved using standard SDP solver while it suffers from the scalability issue. DS* algorithm [2] seeks to solve this issue with a lifting-free paradigm, which depends on a series of linearization.

In practice, the continuous methods have a drawback due to the post binarization step, which can cause uncontrollable degeneration to the final solution quality. Therefore researchers have also tried to directly solve the problem in the discrete assignment space. The Integer Projected Fixed Point (IPFP) method [23] was developed, aiming to find an optimal solution along a (quasi) discrete course. Sampling methods [20, 33] directly generate discrete hypothesis via Monte Carlo Sampling, and the recent method [1] devises a graph matching tailored Tabu search strategy.

From the optimization perspective, there also exist a large number of path-following based algorithms for graph matching [46, 26, 49, 40, 39, 18], including the continuation method used in the classic work [16].

3. GM via Determinant Regularization

3.1. Theorem and Derived Objectives

Graph matching can be relaxed as following[1]:

$$\max_{\mathbf{x}} \mathbf{x}^\top \mathbf{K}\mathbf{x}, \quad \text{s.t. } \mathbf{H}\mathbf{x} = \mathbf{1}, \quad \mathbf{x} \in [0,1]^{n^2} \tag{3}$$

where $\mathbf{K} \in \mathbb{R}^{n \times n}$ is the so-called affinity matrix which is assumed pre-given in this paper, and $\mathbf{H}$ is a selection matrix encoding the one-to-one constraint for node correspondence. Here n is the number of nodes in each graph. Now we first present the following proposition.

[1]We assume $n_1 = n_2$. It can be fulfilled by introducing dummy nodes in one graph with less nodes which is common practice in literature e.g. [41, 49] being dated back to [16] and widely used in linear programming.

Proposition 1. *For any doubly stochastic matrix* $\mathbf{X}$*, we have* $|\det(\mathbf{X})| \leq 1$*. Equality holds iff* $\mathbf{X}$ *is a permutation.*

Proof. According to Hadamard's inequality [15], we have $|\det(\mathbf{X})| \leq \prod_{j=1}^{n} \|\mathbf{X}_j\|$, where $\mathbf{X}_j$ refers to the jth column of $\mathbf{X}$. As $\mathbf{X}$ is doubly stochastic, we have $\mathbf{X}_{ij} \in [0, 1]$ and $\sum_i \mathbf{X}_{ij} = 1$. Thus $\|\mathbf{X}_j\| \leq 1$ and equality holds iff there is only a 1 element in $\mathbf{X}_j$ and all the resting elements are 0s. Therefore, for any column j that's not in 0-1 mode, we must have $\|\mathbf{X}_j\| < 1$. This observation claims that, if $\mathbf{X}$ is not a permutation (in this case non-0/non-1 element exists), $|\det(\mathbf{X})| \leq \prod_{j=1}^{n} \|\mathbf{X}_j\| < 1$.

On the other hand, for any permutation $\mathbf{X}$, we must have $|\det(\mathbf{X})| = 1$ (As the columns of a doubly stochastic matrix are orthogonal). This completes the proof. □

This proposition gives an upper bound for the absolute value of the determinant. Together with the trivial lower bound we have $0 \leq |\det(\mathbf{X})| \leq 1$. The equality of the 0 side holds when the columns/rows of $\mathbf{X}$ are linearly dependent. Based on the discussion above, we construct the following objective involving determinant regularization:

$$\max_{\mathbf{x}} \mathbf{x}^\top \mathbf{K}\mathbf{x} + \lambda |\det(\mathbf{X})|, \quad \text{s.t. } \mathbf{H}\mathbf{x} = \mathbf{1} \tag{4}$$

We refer to objective (4) as $\mathbb{P}_\lambda$. Given objective (4) with a regularization term, two types of optimization strategies, namely multiplication and gradient, are widely applied. The multiplicative strategy is established on certain convergence guarantee where in the current iteration the updated affinity value is greater than the previous one under some mild conditions [17, 16]. In our case, however, it is difficult to devise a multiplication-based updating paradigm since the convergence condition involving matrix inversion cannot be easily found. Instead, we develop a gradient-based optimization algorithm under the framework of path-following [49]. Though we still face the difficulty of calculating matrix inversion in this setting, we provide an effective and efficient way to approximate the gradient of low-rank solution. These will be discussed in Section 4. Although it was reported that the multiplicative updating rule is more computationally efficient, we have found that our gradient-based algorithm can achieve remarkable improvement compared to multiplication-based ones in a reasonable time cost.

3.2. Geometric Property of Gradients

We show some geometric properties of the proposed models in this section and analyze the gradient behavior compared to two regularization counterparts. To this end, we unfold our analysis in Euclidean space, which is more intuitive. We also choose regularization model with ℓ_2 [18] ($\sum_i \mathbf{x}_i^2$) and entropy [16] ($-\sum_i \mathbf{x}_i \log(\mathbf{x}_i)$) for comparison on the polytope. As the underlying contours are visualized, we demonstrate the property of determinant by showing its bound vs ℓ_2 norm. Let us consider the following two sets:

$$\mathcal{S}_1 = \{\mathbf{X} | \|\mathbf{X}\|_F^2 = n\} \quad \mathcal{S}_2 = \{\mathbf{X} | |\det(\mathbf{X})| = 1\} \tag{5}$$

where $\mathcal{S}_1$ is the regularization introduced in [18]. For any $\mathbf{X} \in \mathcal{S}_1$, we have the following formula according to the geometric inequality and Hadamard's inequality:

$$\|\mathbf{X}\|_F^2/n = \sum_{j=1}^{n} \|\mathbf{X}_j\|_2^2/n \geq \sqrt[n]{\prod_{j=1}^{n} \|\mathbf{X}_j\|_2^2} \geq \sqrt[n]{|\det(\mathbf{X})|^2} \tag{6}$$

In general, we have $|\det(\mathbf{X})| \leq 1$ if $\mathbf{X} \in \mathcal{S}_1$. The above analysis implies that, the set of $\{\mathbf{X} | \|\mathbf{X}\|_F^2 \leq n\}$ is a proper subset of $\{\mathbf{X} | |\det(\mathbf{X})| \leq 1\}$. To show the geometric relation among these regularizers in 2D space, we present an example by projecting the 2-permutation polytope onto the 2-dimensional Euclidean space, with two permutation:

$$\mathbf{A}_1 = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}, \qquad \mathbf{A}_2 = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \tag{7}$$

Any point on this polytope can be expressed as:

$$\alpha \mathbf{A}_1 + (1 - \alpha)\mathbf{A}_2 \tag{8}$$

We first give a 2-dimensional expression of determinant regularizer. Given Eq. (8), the corresponding absolute determinant value given α is:

$$\left| \det \left\{ \begin{pmatrix} \alpha & 1-\alpha \\ 1-\alpha & \alpha \end{pmatrix} \right\} \right| = |2\alpha - 1| \tag{9}$$

which is merely dependent on variable α. Upon this juncture, we linearly map permutation vertices $\mathbf{A}_1$, $\mathbf{A}_2$ and original point $(0, 0; 0, 0)$ onto $(0, 1)$, $(1, 0)$ and $(0, 0)$ in regular 2D space, respectively. This can be done with a naive projection matrix from vectorized 4D to 2D space:

$$\mathbf{P} = \begin{pmatrix} 0.5 & 0 & 0 & 0.5 \\ 0 & 0.5 & 0.5 & 0 \end{pmatrix} \tag{10}$$

In this case the 2-permutation polytope in a regular 2D Euclidean space is the interval between points $(0, 1)$ and $(1, 0)$. If we rescale any point on the polytope by $b > 0$, then the absolute value of determinant is $b^2|2\alpha - 1|$. This implies that the absolute value of determinant is the product of rescaled point $b|2\alpha - 1|$ and b. Since b corresponds to an axis orthogonal to the polytope, we conclude that the contour of absolute value of determinant is a hyperbola rotated by $45°$. The contours of ℓ_2 and entropy can be analogously obtained.

The contours of three types of regularizers can be found in Figure 1. Serving as another regulizer besides ℓ_2 and

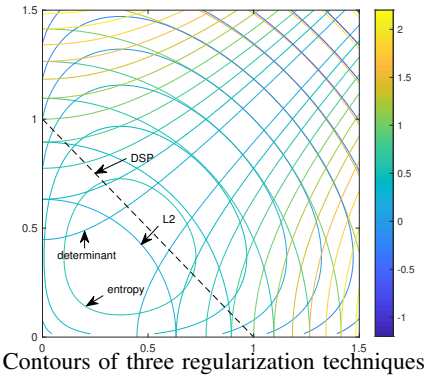


Figure 1. Contours of three regularization techniques on 2D Euclidean space. Black arrows indicate the contours of the regularization terms: entropy, ℓ_2 norm (ℓ_2) and determinant (DET). The black dashed line refers to a 2-dimensional doubly stochastic polytope (DSP), where the discrete solutions lie on coordinates $(0, 1)$ or $(1, 0)$. We see along with emphasizing the regularization weights, either term will push the solution to the discrete ones. However, in the deep interior of the DSP, determinant has much different behavior against the other two.

entropy, we note that absolute determinant shows much different behavior than the other two. When the current solution is in the interior of the polytope and near to the point $\frac{1}{n}\mathbf{1}\mathbf{1}^\top$, the gradients of ℓ_2 and entropy are almost orthogonal to the polytope. In this case, the gradients of regularizers have *no contribution* to the update. However, absolute determinant has more effective gradient (almost in the polytope subspace) in the interior. This effectiveness also holds along with the solution path until polytope vertices, yielding contributing gradient in each iteration.

Furthermore, it is a fact that $|\det(\mathbf{X})| > 0$ when $\mathbf{X}$ is strictly diagonal dominant (i.e. there exist an i for each j, such that $|\mathbf{X}_{ij}| \geq \sum_{k\neq i} \mathbf{X}_{kj}$). Thus for any (not necessarily doubly) stochastic matrix $\mathbf{Y}$, strictly diagonal dominance implies there exist an i such that $\mathbf{Y}_{ij} > 0.5$ for any j. As for doubly stochastic matrix, there will be only one element larger than 0.5 in each column/row, which means there's no ambiguity. In this case, there exists a permutation $\mathbf{P}$ such that $\mathbf{P}\mathbf{X}\mathbf{P}^\top$ is diagonally dominant. We call such $\mathbf{X}$ **doubly diagonally dominant**. Conversely, if for any continuous solution $\det(\mathbf{X}) = 0$, $\mathbf{X}$ cannot be doubly diagonally dominance, which implies ambiguity and is not of our interest. In general, larger determinant value of doubly stochastic solution implies less ambiguity, and a absolute determinant larger than 0 implies strong discriminativity to some extent ($\mathbf{X}_{ij} > 0.5$ exists).

4. Optimization Methods

4.1. Path-following for solving Objective (4)

Given objective (4), we devise a path-following-based method. Path-following, sometimes also called continuation method, is widely adopted in previous works on graph matching e.g. [49]. While the details differ, the common advantage of such strategy includes a guarantee on increasing objective score, as well as the optimum being a permutation, since a discrete solution is what we seek for the original problem. Another reason that we employ path-following rather than multiplication is that it is extremely difficult to find a theoretical ascending guarantee for the determinant-regularized model under multiplicative strategy.

Such an algorithm requires the calculation of gradient of $\det(\mathbf{X})$ w.r.t. $\mathbf{X}$ involve a matrix inversion. However in practice, $\mathbf{X}$ is not necessarily invertible, especially in the early stage of optimization when $\mathbf{X}$ is deep in the interior of the permutation polytope. For the time being, we assume $\mathbf{X}$ is invertible, hence $\mathbf{X}^{-1}$ exists. We will discuss how to handle the case of low-rank $\mathbf{X}$ in the next section.

Our path-following starts from an initial $\lambda_0 > 0$. After a gradient-based algorithm finds the local optimum $\mathbf{X}_0^{\mathcal{O}}$ corresponding to λ_0, it proceeds by amplifying the value with $\lambda_{t+1} > \lambda_t$ and solves problem $\mathbb{P}_{\lambda_{t+1}}$ for the next stage. In all the experiments we employ the amplifying procedure $\lambda_{t+1} = 2\lambda_t$. The gradient of (4) w.r.t. $\mathbf{X}$ is:

$$\hat{\mathbf{X}} = 2\mathbf{K}\mathbf{x} + \lambda \text{sign}(\det(\mathbf{X}))\mathbf{X}^{-\top} \tag{11}$$

Then the algorithm iterates by $\mathbf{X} = \mathbf{X} + \Delta\hat{\mathbf{X}}$, where $\Delta > 0$ is a small increment. Likewise we need to project $\mathbf{X}$ back to the permutation polytope. Rather than Sinkhorn's algorithm [32], we employ a slightly revised projection method from [44] which manipulates columns and rows simultaneously. To this end, we first let $\mathbf{X}_{ij} = 0$ if $\mathbf{X}_{ij} < 0$, then repeat the following projection:

$$\mathbf{X} = \mathbf{X} + \frac{1}{n}\mathbf{1}\mathbf{1}^\top - \frac{1}{n}\mathbf{X}\mathbf{1}\mathbf{1}^\top - \frac{1}{n}\mathbf{1}\mathbf{1}^\top\mathbf{X} + \frac{1}{n^2}\mathbf{1}\mathbf{1}^\top\mathbf{X}\mathbf{1}\mathbf{1}^\top \tag{12}$$

which theoretically ensures to converge to a point in the permutation polytope. This algorithm has fast convergence in practice (less than 10 iterations to an acceptable precision).

4.2. Spectral Sub-gradient

As discussed earlier, $\mathbf{X}$ is not necessarily invertible. In this case, the assumption of updating rule (11) no longer holds. This issue possibly happens when the current solution is close to the center of the polytope interior. One may imagine a naive replacement of matrix inversion with the pseudo-inversion, which can be applied on singular matrices. However, pseudo-inversion will keep the zero eigenvalues intact, and will not fulfill the purpose of maximizing the absolute determinant.

To address this issue, we first diverge to look into the partial objective $\max |\det(\mathbf{X})|$. Since this partial term seeks to maximize the absolute determinant, any small value above 0 will be a proper increment on $|\det(\mathbf{X})|$ if the current determinant is 0. Thus any incremental direction in terms of

Input: $\mathbf{X}$; tolerance ϵ
Output: sub-gradient $\bar{\mathbf{X}}$
$\mathbf{U}\mathbf{\Lambda}\mathbf{U}^{-1} \leftarrow \mathbf{X}$ (eigen-decomposition)
$\hat{\mathbf{\Lambda}} \leftarrow \mathbf{\Lambda}$
for $\forall i, |\mathbf{\Lambda}_{ii}| < \epsilon$ **do**
  if $\mathbf{\Lambda}_{ii} = 0$ **then**
    $\hat{\mathbf{\Lambda}}_{ii} = \epsilon$
  end
  $\hat{\mathbf{\Lambda}}_{ii} = \mathrm{sign}(\mathbf{\Lambda}_{ii})\epsilon$
end
$\mathbf{X}_u = \mathbf{U}\hat{\mathbf{\Lambda}}\mathbf{U}^{-1}$
return $\bar{\mathbf{X}} = \mathbf{X}_u - \mathbf{X}$

Algorithm 1: Spectral sub-gradient for DetGM$_1$.

absolute determinant implies a proper "sub-gradient", and we can adopt such sub-gradient for update. To this end, we first perform eigen-decomposition on $\mathbf{X}$:

$$\mathbf{X} = \mathbf{U}\mathbf{\Lambda}\mathbf{U}^{-1} \tag{13}$$

where $\mathbf{U}$ is an orthogonal matrix containing both the basis of linear and null spaces and $\mathbf{\Lambda}$ is the diagonal matrix with eigen-values in the magnitude descending order $(\sigma_1, ..., \sigma_q, 0, ..., 0)$, $|\sigma_{t-1}| \geq |\sigma_t|$ for any t. Now we present two types of proper sub-gradients.

4.2.1 Method Design by Increment Update: DetGM$_1$

We employ a simple yet effective amplification procedure by letting any eigenvalue with magnitude smaller than a tolerance $\epsilon > 0$ to ϵ. The amplified eigenvalues become $(\sigma_1, ..., \sigma_s, \mathrm{sign}(\sigma_{s+1})\sigma_{s+1}, ..., \mathrm{sign}(\sigma_r)\sigma_r, \epsilon, ..., \epsilon)$, where $s \leq q$, $0 < |\sigma_{s+1}| < \epsilon$, $|\sigma_s| \geq \epsilon$, $\sigma_r \neq 0$ and $\sigma_{r+1} = 0$ in the original eigen-values and q is the index of the first element which is negative value but with magnitude larger than ϵ. We add a tolerance ϵ to accommodate the calculating precision of float numbers. As such, each eigen-value is above 0, thus the determinant will not vanish. Let the diagonalized amplified eigenvalue matrix be $\hat{\mathbf{\Lambda}}$, then the modified matrix with small non-zero determinant can be written as:

$$\mathbf{X}_u = \mathbf{U}\hat{\mathbf{\Lambda}}\mathbf{U}^{-1} \tag{14}$$

Then the difference $\bar{\mathbf{X}} = \mathbf{X}_u - \mathbf{X} \approx \frac{\partial |\det(\mathbf{X})|}{\mathbf{X}}$ can be viewed as a proper ascending direction w.r.t. $\mathbf{X}$, as by adding $\bar{\mathbf{X}}$, $|\det(\mathbf{X})|$ becomes above 0. This procedure is summarized in Algorithm 1.

4.2.2 Method Design by Geometric Update: DetGM$_2$

This line of sub-gradient is motivated by the geometric inequality $\frac{1}{n}\sum_i a_i \geq \sqrt[n]{\prod_i a_i}$ for $a_i \geq 0$. It is easy to show that $\prod_i a_i$ is concave in the affine subset $\sum_i a_i = d$, where $a_i \geq 0$ and $d > 0$ is a constant. The maximal value is reached if and only if $a_i = d/n$ for each i. According to the concavity and the reachable maximum, it is easy to conclude that for any $0 < \theta < 1$ and the reweighted point $\mathbf{b} = (1-\theta)\mathbf{a} + \theta\mathbf{m}$, it have to be $\prod_i b_i \geq \prod_i a_i$. Here $\mathbf{a} = (a_1, ..., a_n)$ and $\mathbf{m} = (n/d, ..., n/d)$. Motivated by this observation, we devise another type of valid sub-gradient for absolute determinant in graph matching.

Input: $\mathbf{X}$; tolerance ϵ; step θ
Output: sub-gradient $\bar{\mathbf{X}}$
$\mathbf{U}\mathbf{\Lambda}\mathbf{U}^{-1} \leftarrow \mathbf{X}$ (eigen-decomposition)
$g \leftarrow \sum_i |\mathbf{\Lambda}_{ii}|$
for i **do**
  if $\mathbf{\Lambda}_{ii} \neq 0$ **then**
    $\mathbf{\Gamma}_i \leftarrow \frac{g}{n}\mathrm{sign}(\mathbf{\Lambda}_{ii}) - \mathbf{\Lambda}_{ii}$
  else
    $\mathbf{\Gamma}_i \leftarrow \frac{g}{n}$
  end
end
return $\bar{\mathbf{X}} \leftarrow \theta\mathbf{U}\mathrm{diag}(\mathbf{\Gamma})\mathbf{U}^{-1}$

Algorithm 2: Spectral sub-gradient for DetGM$_2$.

To this end, we first calculate the summation of all the absolute eigen-values in $\mathbf{\Lambda}$ as $g = \sum_i |\mathbf{\Lambda}_{ii}|$. g is regarded as a constant. Then under fixed summation g, the maximal determinant value can be achieved iff each eigen-value is equal to g/n. Then we can calculate the gap $\mathbf{\Gamma} = \mathbf{m} - \mathbf{a}$ between $\mathbf{m} = (g/n, ..., g/n)$ and $\mathbf{a} = (\mathbf{\Lambda}_{11}, ..., \mathbf{\Lambda}_{nn})$. To avoid the gradient to be too aggressive, we assign a small step θ to $\mathbf{\Gamma}$. Thus the sub-gradient is written as:

$$\bar{\mathbf{X}} = \theta\mathbf{U}\mathrm{diag}(\mathbf{\Gamma})\mathbf{U}^{-1} \tag{15}$$

This procedure is given in Algorithm 2. The main overhead for both algorithms lies in the eigen-decomposition, resulting in similar computational efficiency.

5. Experiments

All the experiments are conducted on a laptop with 3.0GHz CPU and 16GB memory.

Datasets and metrics. Experiments involve both synthetic data and real-world images. The synthetic dataset is from the popular Random Graph Matching [8]. The real images include the CMU house sequence [3], Caltech-101/MSRC object matching [8] and Pascal dataset [24]. For evaluation, *accuracy*, *score* and *ratio* are evaluated, where *accuracy* measures the portion of correctly matched nodes with respect to all nodes, *score* represents the value of the objective function and *ratio* emphasizes the ratio between current objective value and the maximal one.

Method	GAGM	BGM	RRWM	BPF	GGM_{lp}	DetGM
Acc.	73.66	76.56	72.95	75.14	76.69	**77.44**
Ratio	0.933	0.970	0.946	1	0.972	0.985

Table 1. Accuracy (%) and ratio on Caltech-101 natural images.

Compared Methods. Compared methods include Integer Projected Fixed Point (**IPFP**) [23], Graduated Assignment (**GAGM**) [16], Reweighted Random Walk (**RRWM**) [8], Binary-preserving Graph Matching (**BGM**) [18], and two very recent state-of-the-art solvers: Branching Path Following Matching (**BPF**) [40, 39] and Generalized Graph Matching (**GGM**) [43]. For **GGM**, we select the setting of $\mathbf{GGM_{lp}}$ (refer [43] for more details). We term our method $\mathbf{DetGM}_1$ and $\mathbf{DetGM}_2$ for Sec. 4.2.1 and Sec. 4.2.2, respectively. In all experiments, all algorithms are initialized with a uniform matching.

5.1. Results on Synthetic Data

For each trial of the matching, a pair of graphs G^S and G^D is generated with n_{in} inlier nodes on both, where we let $n_{in} = 40$ in all tests ($n_{in} = 40$ is more challenging than a usual setting $n_{in} = 20$ such as in [8]). The attribute $\mathbf{a}_{ij}^k$ with $k \in \{S, D\}$ on edge is randomly generated from a multivariate uniform distribution ($i = j$ and $i \neq j$ correspond to node and edge attributes, respectively). Attributes on graph G^D are the copies of those on graph G^S with Gaussian noise $\epsilon \sim \mathcal{N}(0, \delta)$, namely $\mathbf{a}_{ij}^D = \mathbf{a}_{ij}^D + \epsilon$. For a pair of edges (i, j) in G^S and (a, b) in G^D, the corresponding affinity is calculated as $\mathbf{A}_{ij:ab} = \exp(-|\mathbf{a}_{ij}^S - \mathbf{a}_{ab}^D|^2/\sigma_s^2)$, where σ_s is a parameter and set to be 0.15 during all the synthetic tests. We conduct three types of empirical experiments with varying deformation noise ϵ, outlier count n_{out} and edge density ρ [8]. In each experiment, we independently conduct 200 times of the single trial and report the average *accuracy* and *score* (objective value).

Figure 2 summarizes the synthetic experimental results. We see that DetGM_1 and DetGM_2 outperform all the selected algorithms in all settings. Particularly in the outlier test, the algorithms of DetGM significantly surpass both BPF and GGM_{lp} when there are massive outliers. We also observe that, though the objective *scores* of $\text{DetGM}_{1,2}$, BFP and GGM_{lp} are similar, there is a significant gap w.r.t. *accuracy*. This fact indicates again that the score may not necessarily reflect the true matching, which has also been pointed out in [41, 43]. $\text{DetGM}_{1,2}$, BPF and GGM_{lp} show similar performance in edge density test.

Remarks IPFP, GAGM and RRWM are in the line of *multiplication-based* updating strategy, where in iteration the product between the affinity and the previous solution contributes most to the current solution. Instead, $\text{DetGM}_{1,2}$, BGM, BPF and GGM_{lp} are *gradient-based*, which implies these algorithms employ a much more cautious update in each iteration other than multiplication-based ones. While the performance of the two categories of updating strategies hardly differentiates when the number of nodes is small (up to 20 for inliers and 10 for outliers), which has been verified by multiple previous works [8, 18, 43], it can be concluded from our experiments that gradient-based methods have more stable performance compared with multiplication-based ones *if number of nodes is large*. We infer the reason of such phenomenon is as follows. With increasing problem dimension, the affinity matrix $\mathbf{A}$ tends to deliver more complex (2nd-order) local behavior, thus an aggressive update strategy such as multiplication will likely jump over previous local region and diverge to an inconsistent solution path. Besides, the convergence criterion of multiplicative strategy is typically due to the gap of two consecutive objective scores. When $\mathbf{A}$ is complex, with a higher probability an ascending gradient may still exist even if the gap is small. The aforementioned factors will result in weak solution.

5.2. Results on Real-world Data

CMU House and Hotel Sequence. CMU house sequence has 110 images in total. We follow the widely adopted protocol in [8] to conduct the test under varying sequence gap $(10, 20, ..., 100)$, resulting in 560 image pairs. For each image, 30 landmark points are manually labelled. And for each pair of images, 10 landmarks are randomly removed from the first image. The graph is established with Delaunay triangulation. The affinity is obtained as $\mathbf{K}_{ij:ab} = \exp(-|\mathbf{a}_{ij}^S - \mathbf{a}_{ab}^D|^2/\sigma_s^2)$, where $\mathbf{a}_{ij}^S$ measures the Euclidean distance between point i and j, and $\sigma_s^2 = 2500$ empirically.

Typical matching example with BPF, DetGM_1 and DetGM_2 are shown in Figure 5, and we report *accuracy* in Figure 4. It can be observed that in both settings of the proposed algorithm can reach out competitive performance against state-of-the-art algorithms BPF and GGM_{lp}, and significantly outperforms the resting algorithms in *accuracy*. The two settings DetGM_1 and DetGM_2 show very similar performance in CMU house and synthetic tests, indicating both of the updating rules are valid and the corresponding solution paths do not diverge much. For the rest if the experiments, we only report the behavior of DetGM_1 due to their high similarity while abbreviating it as **DetGM**.

Caltech-101. It [8] consists of 30 pairs of images from Caltech-101 [14] and MSRC[2]. The features points are generated by MSER detector [28] and each point is assigned its SIFT feature [27]. The candidate matchings are selected by comparing the feature distance with a threshold 0.6, which

[2] http://research.microsoft.com/vision/cambridge/recognition/

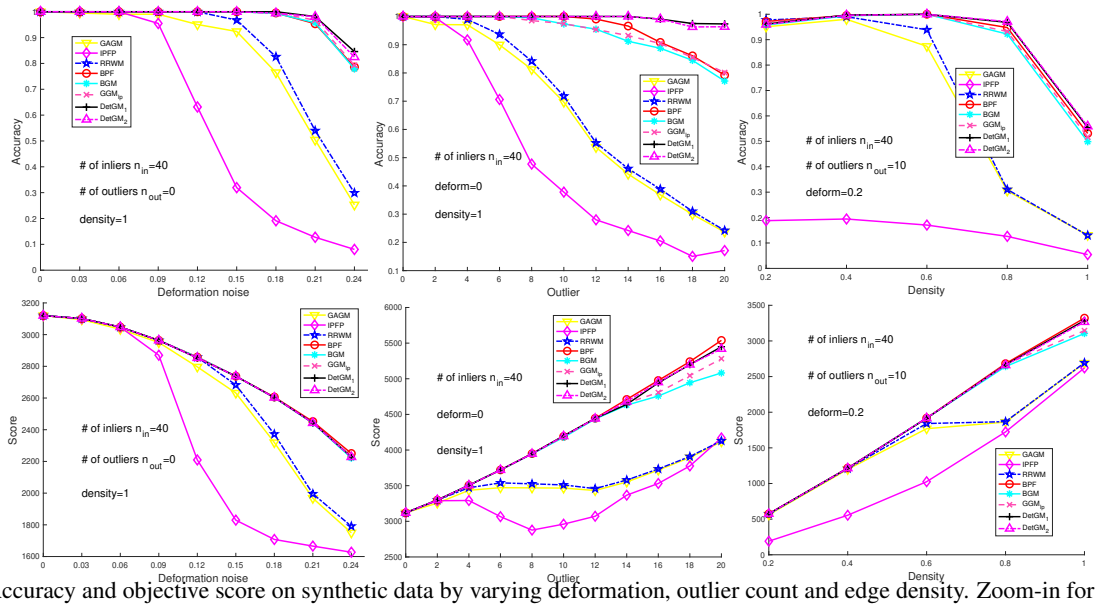


Figure 2. Accuracy and objective score on synthetic data by varying deformation, outlier count and edge density. Zoom-in for better view.

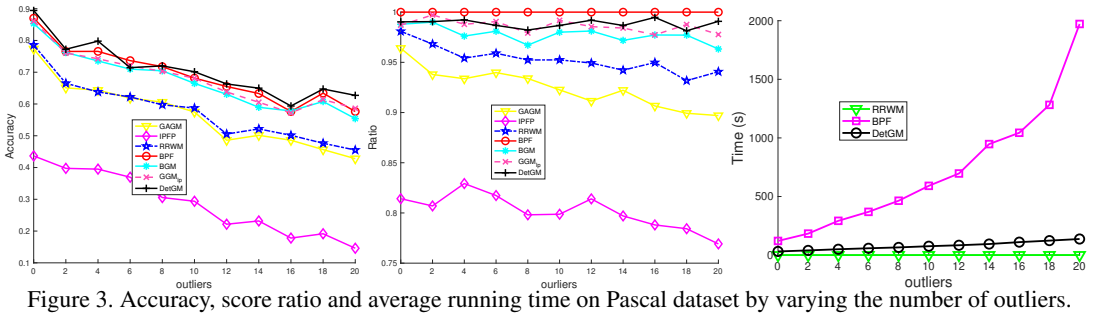


Figure 3. Accuracy, score ratio and average running time on Pascal dataset by varying the number of outliers.

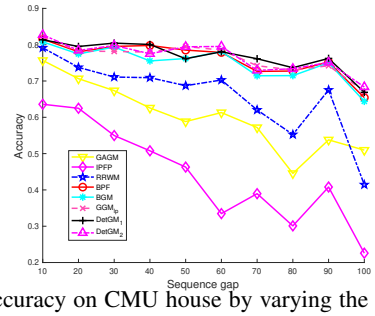


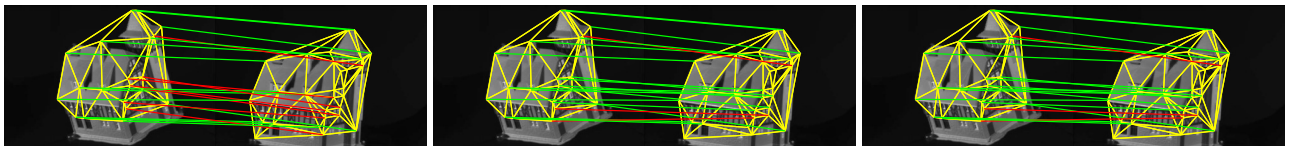
Figure 4. Accuracy on CMU house by varying the sequence gap, for two graphs with 20 and 30 nodes respectively.

allows multiple correspondences for each feature. The dissimilarity between a candidate pair (i, j) and (a, b) is obtained with $\mathbf{K}_{ij:ab} = \max(50 - d_{ij:ab}, 0)$ where $d_{ij:ab}$ refers to the mutual projection error [7].

Results are reported in Table 1 and matching examples are shown in Figure 6. One can see that our method outperforms the peers including most up-to-date BPF and GGM_{lp}.

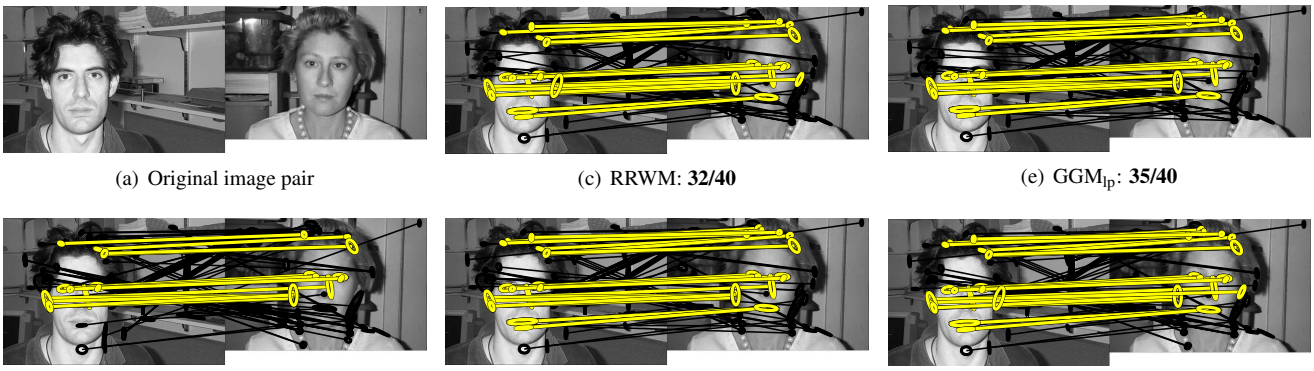
Pascal Dataset. Pascal dataset [24] consists of 20 pairs of motorbike images and 30 pairs of car images collected from Pascal07 dataset [13]. For each image pair, feature points and corresponding ground-truth correspondence are manually labelled. We follow the popular protocol [49] to randomly select 0 to 20 outliers from the background to evaluate the algorithms under degradation. We follow [49] to generate the graph and the corresponding affinity. For each node i, a feature $\mathbf{f}_i$ is assigned by taking its orientation of the normal vector at that node to the contour where the point was sampled. Then the node affinity between node i and j is computed as $\exp(-|\mathbf{f}_i - \mathbf{f}_j|)$. Delaunay triangulation is performed to obtain the graphs, and pairwise distance d_{ij} and absolute angle θ_{ij} are calculated on any valid edge (i, j). Thus the affinity between edge (i, j) in G^S and edge (a, b) in G^D is $\mathbf{K}_{ij:ab} = \exp(-(|d^S_{ij} - d^D_{ab}| + |\theta^S_{ij} - \theta^D_{ij}|))/2$.

Figure 7 presents an example of matching results on 6 counterparts and Figure 3 shows the quantitative results of the experiments. We also present time cost comparison against two typical solvers: RRWM and BPF. It can be seen that our algorithm achieves competitive performance



(a) BPF: **13/20** (b) DetGM$_1$: **17/20** (c) DetGM$_2$: **17/20**

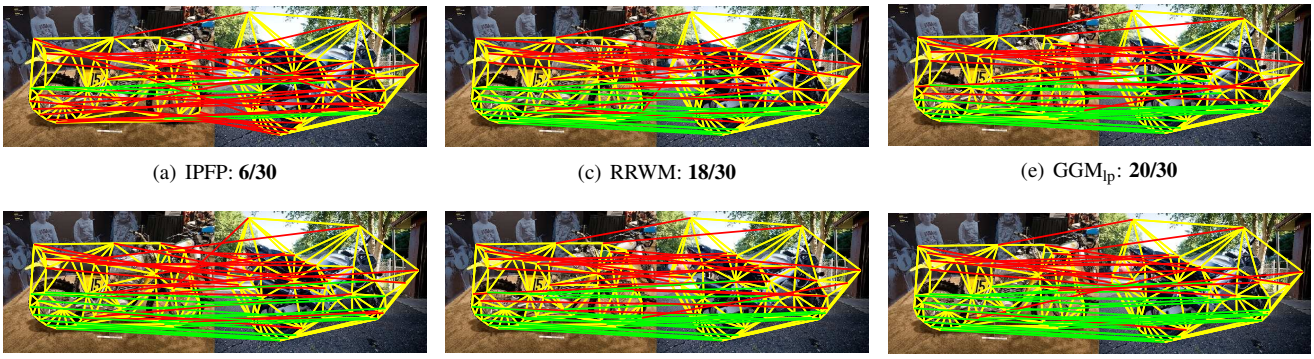
Figure 5. Matching examples on CMU house dataset with 20 and 30 nodes for each graph respectively. Correct matchings are in green



(a) Original image pair (c) RRWM: **32/40** (e) GGM$_{\text{lp}}$: **35/40**

(b) GAGM: **20/40** (d) BPF: **35/40** (f) DetGM: **36/40**

Figure 6. Examples on Caltech-101 dataset with 40 nodes on a pair of face images. Correct matchings are in yellow.



(a) IPFP: **6/30** (c) RRWM: **18/30** (e) GGM$_{\text{lp}}$: **20/30**

(b) GAGM: **18/30** (d) BPF: **20/30** (f) DetGM: **24/30**

Figure 7. Examples on Pascal dataset with **30** inliers and **12** outliers on a pair of motorbike images. Correct matchings are in green.

against BPF and GGM$_{\text{lp}}$ and outperforms them in some specific outlier settings. It should also be noted that, though BPF achieves high *accuracy*, the running speed of BPF is extremely slow. From the third subfigure of Figure 3, we can find that in a typical setting with 20 outliers, BPF costs over $2,000$ seconds to finish one trial in average. However, our algorithm only need around 60 seconds in average. Though RRWM has higher efficiency, our algorithm still shows moderate computational cost in practical use.

6. Conclusion

To enable gradient-efficient continuation optimization, the paper has presented a novel regularization technique for graph matching, as derived from the determinant analysis on the node matching matrix between two graphs. Theoretical property of our relaxation technique is studied and we also give some analysis on the geometric behavior compared to existing regularizers, which has rarely been considered. These findings are anticipated to bring about insights to other regularized objective with affine constraints. Extensive experiments are performed which show the state-of-the-art accuracy as well as efficiency of our method.

Acknowledgement

Tianshu Yu and Baoxin Li were supported in part by a grant from ONR. Any opinions expressed in this material are those of the authors and do not necessarily reflect the views of ONR. Junchi Yan was partially supported by China Major State Research Development Program 2018AAA0100704, NSFC (61972250, U19B2035), and the Open Project Program of the National Laboratory of Pattern Recognition (NLPR), China.

References

[1] K. Adamczewski, Y. Suh, and K. Lee. Discrete tabu search for graph matching. In *ICCV*, 2015.

[2] Florian Bernard, Christian Theobalt, and Michael Moeller. Ds*: Tighter lifting-free convex relaxations for quadratic matching problems. In *CVPR*, 2018.

[3] T. Caetano, T. Caelli, D. Schuurmans, and D. Barone. Graphical models and point pattern matching. *TPAMI*, 28(10):1646–1663, 2006.

[4] T. Caetano, J. McAuley, L. Cheng, Q. Le, and A. J. Smola. Learning graph matching. *TPAMI*, 31(6):1048–1058, 2009.

[5] M. Chertok and Y. Keller. Efficient high order matching. *TPAMI*, 2010.

[6] M. Cho, K. Alahari, and J. Ponce. Learning graphs to match. In *ICCV*, 2013.

[7] Minsu Cho, Jungmin Lee, and Kyoung Mu Lee. Feature correspondence and deformable object matching via agglomerative correspondence clustering. In *ICCV*, 2009.

[8] M. Cho, J. Lee, and K. M. Lee. Reweighted random walks for graph matching. In *ECCV*, 2010.

[9] T. Cour, P. Srinivasan, and J. Shi. Balanced graph matching. In *NIPS*, 2006.

[10] X. Du, J. Yan, and H. Zha. Joint link prediction and network alignment via cross-graph embedding. In *IJCAI*, 2019.

[11] O. Duchenne, F. Bach, I. Kweon, and J. Ponce. A tensor-based algor ithm for high-order graph matching. *TPAMI*, 2011.

[12] O. Duchenne, A. Joulin, and J. Ponce. A graph-matching kernel for object categorization. In *ICCV*, 2011.

[13] M. Everingham, L. Van Gool, C. K. Williams, J. Winn, and A. Zisserman. The pascal visual object classes challenge 2007 (voc2007) results. 2007.

[14] L. Fei-Fei, R. Fergus, and P. Perona. Learning generative visual models from few training examples: An incremental bayesian approach tested on 101 object categories. *CVIU*, 106(1):59–70, 2007.

[15] A. M. Fink. Hadamard's inequality for log-concave functions. *Mathematical and Computer Modelling*, 32(5–6):625–629, 2000.

[16] S. Gold and A. Rangarajan. A graduated assignment algorithm for graph matching. *TPAMI*, 1996.

[17] B. Jiang, J. Tang, C. Ding, Y. Gong, and B. Luo. Graph matching via multiplicative update algorithm. In *NIPS*, 2017.

[18] Bo Jiang, Jin Tang, Chris Ding, and Bin Luo. Binary constraint preserving graph matching. In *CVPR*, 2017.

[19] V. G. Kim, W. Li, N. J. Mitra, S. DiVerdi, and T. Funkhouser. Exploring collections of 3d models using fuzzy correspondences. In *SIGGRAPH*, 2012.

[20] J. Lee, M. Cho, and K.M. Lee. A graph matching algorithm using data-driven markov chain monte carlo sampling. In *ICPR*, 2010.

[21] J. Lee, M. Cho, and K. M. Lee. Hyper-graph matching via reweighted randomwalks. In *CVPR*, 2011.

[22] M. Leordeanu and M. Hebert. A spectral technique for correspondence problems using pairwise constraints. In *ICCV*, 2005.

[23] M. Leordeanu, M. Hebert, and R Sukthankar. An integer projected fixed point method for graph matching and map inference. In *NIPS*, 2009.

[24] M. Leordeanu, R. Sukthankar, and M. Hebert. Unsupervised learning for graph matching. *Int. J. Comput. Vis.*, pages 28–45, 2012.

[25] M. Leordeanu, A. Zanfir, and C. Sminchisescu. Semi-supervised learning and optimization for hypergraph matching. In *ICCV*, 2011.

[26] Z.Y. Liu, H. Qiao, and L. Xu. An extended path following algorithm for graph-matching problem. *TPAMI*, pages 1451–1456, 2012.

[27] D. G. Lowe. Object recognition from local scale-invariant features. In *ICCV*, page 1150, 1999.

[28] J. Matas, O. Chum, M. Urban, and T. Pajdla. Robust wide-baseline stereo from maximally stable extremal regions. *Image and vision computing*, 22(10):761–767, 2004.

[29] J. Munkres. Algorithms for the assignment and transportation problems. *Journal of the Society for Industrial & Applied Mathematics*, 5(1):32–38, 1957.

[30] Q. Ngoc, A. Gautier, and M. Hein. A flexible tensor block coo rd inate ascent scheme for hypergraph matching. In *CVPR*, 2015.

[31] C. Schellewald and C. Schnörr. Probabilistic subgraph matching based on convex relaxation. In *EMMCVPR*, 2005.

[32] R. Sinkhorn. A relationship between arbitrary positive matrices and doubly stochastic matrices. *The annals of mathematical statistics*, 35(2):876–879, 1964.

[33] Y. Suh, M. Cho, and K. M. Lee. Graph matching via sequential monte carlo. In *ECCV*, 2012.

[34] Paul Swoboda, Carsten Rother, Hassan Abu Alhaija, Dagmar Kainmuller, and Bogdan Savchynskyy. A study of lagrangean decompositions and dual ascent solvers for graph matching. In *CVPR*, 2017.

[35] Y. Tian, J. Yan, H. Zhang, Y. Zhang, X. Yang, and H. Zha. On the convergence of graph matching: Graduated assignment revisited. In *ECCV*, 2012.

[36] P. H. S. Torr. Solving markov random fields using semidefinite programmin. In *AISTATS*, 2003.

[37] L. Torresani, V. Kolmogorov, and C. Rother. Feature correspondence via graph matching: Models and global optimization. In *ECCV*, 2008.

[38] R. Wang, J. Yan, and X. Yang. Learning combinatorial embedding networks for deep graph matching. In *ICCV*, 2019.

[39] T. Wang, H. Ling, C. Lang, and S. Feng. Graph matching with adaptive and branching path following. *TPAMI*, 40(12):2853–2867, 2018.

[40] T. Wang, H. Ling, C. Lang, and J. Wu. Branching path following for graph matching. In *ECCV*, 2016.

[41] J. Yan, M. Cho, H. Zha, X. Yang, and S. Chu. Multi-graph matching via affinity optimization with graduated consistency regularization. *TPAMI*, 2016.

[42] J. Yan, C. Zhang, H. Zha, W. Liu, X. Yang, and S. Chu. Discrete hyper-graph matching. In *CVPR*, 2015.

[43] T. Yu, J. Yan, Y. Wang, W. Liu, and B. Li. Generalizing graph matching beyond quadratic assignment model. In *NIPS*, 2018.

[44] T. Yu, J. Yan, J. Zhao, and B. Li. Joint cuts and matching of partitions in one graph. In *CVPR*, 2018.

[45] A. Zanfir and C. Sminchisescu. Deep learning of graph matching. In *CVPR*, 2018.

[46] M. Zaslavskiy, F. R. Bach, and J.-P. Vert. A path following algorithm for the graph matching problem. *TPAMI*, 2009.

[47] R. Zass and A. Shashua. Probabilistic graph and hypergraph matching. In *CVPR*, 2008.

[48] Zhen Zhang, Qinfeng Shi, Julian McAuley, Wei Wei, Yanning Zhang, and Anton Van Den Hengel. Pairwise matching through max-weight bipartite belief propagation. In *CVPR*, 2016.

[49] F. Zhou and F. D. Torre. Factorized graph matching. In *CVPR*, 2012.

[50] F. Zhou and F. D. Torre. Deformable graph matching. In *CVPR*, 2013.