Neural Graph Matching Network: Learning Lawler's Quadratic Assignment Problem with Extension to Hypergraph and Multiple-graph Matching

Runzhong Wang, Junchi Yan, *Member, IEEE,* and Xiaokang Yang, *Fellow, IEEE*

Abstract—Graph matching involves combinatorial optimization based on edge-to-edge affinity matrix, which can be generally formulated as Lawlers Quadratic Assignment Problem (QAP). This paper presents a QAP network directly learning with the affinity matrix (equivalently the association graph) whereby the matching problem is translated into a vertex classification task. The association graph is learned by an embedding network for vertex classification, followed by Sinkhorn normalization and a cross-entropy loss for end-to-end learning. We further improve the embedding model on association graph by introducing Sinkhorn based matching-aware constraint, as well as dummy nodes to deal with unequal sizes of graphs. To our best knowledge, this is the first network to directly learn with the general Lawlers QAP. In contrast, recent deep matching methods focus on the learning of node and edge features in two graphs respectively. We also show how to extend our network to hypergraph matching, and matching of multiple graphs. Experimental results on both synthetic graphs and real-world images show its effectiveness. For pure QAP tasks on synthetic data and QAPLIB benchmark, our method can perform competitively and even surpass state-of-the-art graph matching and QAP solvers with notable less time cost. Source code will be made public at https://github.com/Thinklab-SJTU/.

Index Terms—Graph Matching, Deep Learning, Quadratic Assignment Problem, Combinatorial Optimization, Graph Neural Networks.

1 Introduction and Preliminaries

Graph matching (GM) has been a fundamental problem which is NP-complete in general [1]. It has various applicability and connection with vision and learning, which involves establishing node correspondences between two graphs based on the node-to-node and edge-to-edge affinity [2], [3]. This differs from the point-based techniques e.g. RANSAC [4] and iterative closest point (ICP) [5] without considering edge information.

We start with two-graph matching, which can be written as quadratic assignment programming (QAP) [6], where $\mathbf{X} \in \mathbb{R}^{n_1 \times n_2}$ is a (partial) permutation matrix encoding node-to-node correspondence (with constraints in second line of Eq. (1)), and $\mathrm{vec}(\mathbf{X})$ is its column-vectorized version:

$$\begin{aligned} & J(\mathbf{X}) = \mathrm{vec}(\mathbf{X})^\top \mathbf{K} \mathrm{vec}(\mathbf{X}) \\ s.t. \quad & \mathbf{X} \in \{0,1\}^{n_1 \times n_2}, \mathbf{X}\mathbf{1}_{n_2} = \mathbf{1}_{n_1}, \mathbf{X}^\top \mathbf{1}_{n_1} \leq \mathbf{1}_{n_2} \end{aligned} \tag{1}$$

Here $\mathbf{1}_n$ means column vector of length n whose elements all equal to 1, and $\mathbf{K} \in \mathbb{R}^{n_1 n_2 \times n_1 n_2}$ is the so-called affinity matrix [7]. Its diagonal and off-diagonal elements store the node-to-node and edge-to-edge affinities. For graph matching, the objective $J(\mathbf{X})$ is maximized, assuming perfect matching corresponds to the highest affinity score. One popular embodiment of $\mathbf{K}$ is fixed Gaussian kernel with Euclid distance over pairs of edge features [2], [8]:

$$K_{ia,jb} = \exp\left(\frac{||\mathbf{f}_{ij} - \mathbf{f}_{ab}||^2}{\sigma^2}\right) \tag{2}$$

where $\mathbf{f}_{ij}$ is the feature vector of the edge E_{ij} in graph 1, $\mathbf{f}_{ab}$ from edge E_{ab} in graph 2. $K_{ia,jb}$ is indexed by $(an_1 + i, bn_1 + j)$ under its matrix form.

TABLE 1
Summary of existing learning-free and learning-based methods on two popular formulations of QAP. Note that Lawler's QAP (Eq. 1) is most general which incorporates the Koopmans-Beckmann's QAP (Eq. 3).

	Koopmans-Beckmann's QAP	Lawler's QAP
learning-free	[9], [10], [11], [12]	[2], [7], [13], [14], [15]
learning-based	[16], [17]	ours

Note Eq. (1) in literature is called Lawler's QAP [18], which can incorporate other special forms. For instance, the popular Koopmans-Beckmann's QAP [19] is written by:

$$J(\mathbf{X}) = \mathrm{tr}(\mathbf{X}^\top \mathbf{F}_1 \mathbf{X} \mathbf{F}_2) + \mathrm{tr}(\mathbf{K}_p^\top \mathbf{X}) \tag{3}$$

where $\mathbf{F}_1 \in \mathbb{R}^{n_1 \times n_1}$, $\mathbf{F}_2 \in \mathbb{R}^{n_2 \times n_2}$ are weighted adjacency matrices. $\mathbf{K}_p \in \mathbb{R}^{n_1 \times n_2}$ is node-to-node affinity matrix. Its connection to Lawler's QAP becomes clear by letting $\mathbf{K} = \mathbf{F}_2 \otimes_{\mathcal{K}} \mathbf{F}_1$ ($\otimes_{\mathcal{K}}$ means Kronecker product).

As shown in Tab. 1, traditional learning-free solvers have been extensively studied for both QAP formulations in Eq. (1, 3), and there exist some recent advances in learning Koopmans-Beckmann's QAP [16], [17]. In this paper, we propose the first learning-based algorithm tackling the most

• *R. Wang, J. Yan are with Department of Computer Science and Engineering, and MoE Key Lab of Artificial Intelligence, AI Institute, Shanghai Jiao Tong University, Shanghai, 200240, P.R. China.*
X. Yang is with MoE Key Lab of Artificial Intelligence, AI Institute, Shanghai Jiao Tong University, Shanghai, 200240, P.R. China.
E-mail: {runzhong.wang, yanjunchi, xkyang}@sjtu.edu.cn
Junchi Yan is the corresponding author.

Manuscript, under review.

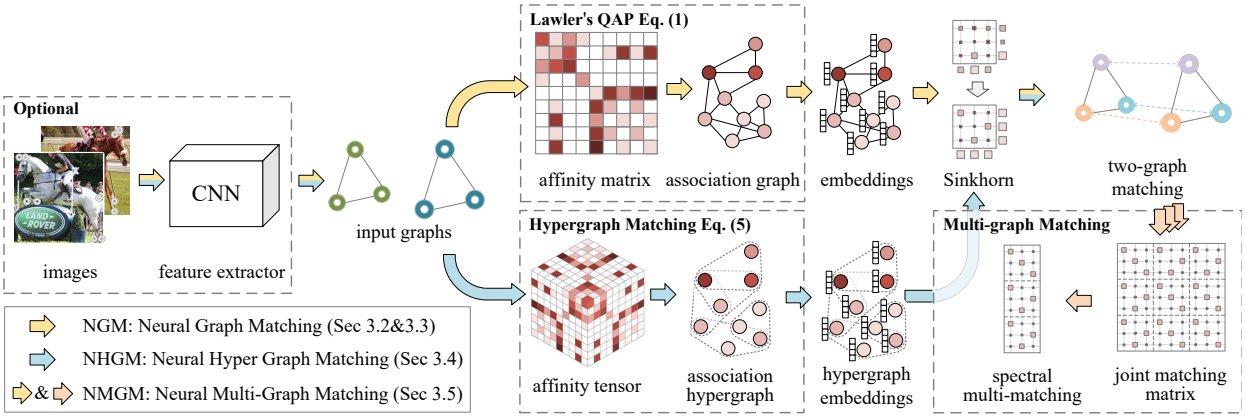


Fig. 1. Overview of the proposed neural graph matching pipeline. The method directly handles Lawler's QAP based on the embedding on the association graph. We further extend it to hypergraph matching by replacing the association graph with an association hypergraph (see Sec. 3.4), as well as to multiple graph matching by differentiable spectral multi-matching (see Sec. 3.5).

general QAP form – Lawler's QAP, and show its generalization to higher-order and multi-graph scenarios.

The above QAP models involve the second-order affinity, and can also be generalized to the higher-order case. A line of works [20], [21], [22], [23] adopt tensor marginalization based model for m-order ($m \geq 3$) hypergraph matching, resulting in a higher-order assignment problem:

$$\begin{aligned} J(\mathbf{x}) &= \mathbf{H} \otimes_1 \mathbf{x} \otimes_2 \mathbf{x} \ldots \otimes_m \mathbf{x} \\ s.t. \quad & \mathbf{X}\mathbf{1}_{n_2} = \mathbf{1}_{n_1}, \mathbf{X}^\top \mathbf{1}_{n_1} \leq \mathbf{1}_{n_2} \end{aligned} \tag{4}$$

where $\mathbf{x} = \mathrm{vec}(\mathbf{X}) \in \{0,1\}^{n_1 n_2 \times 1}$ is the column-vectorized form, and $\mathbf{H}$ is the m-order affinity tensor whose $(n_1 n_2)^m$ elements record the affinity between two hyperedges, operated by tensor product $\otimes_k$ [24]:

$$(\mathbf{H} \otimes_k \mathbf{x})_{\ldots,i_{k-1},i_{k+1},\ldots} = \sum_{i_k=1}^{n_1 n_2} \mathbf{H}_{\ldots,i_{k-1},i_k,i_{k+1},\ldots} \cdot \mathbf{x}_{i_k} \tag{5}$$

where $\otimes_k$ can be regarded as tensor marginalization at dimension k. Details of tensor multiplication can be referred to Sec. 3.1 in [21]. Most existing hypergraph matching works assume the affinity tensor is invariant w.r.t. the index of the hyperedge pairs for computational tractability.

As discussed above, either graph matching or hypergraph matching problem involves solving a combinatorial optimization problem. However, the objective functions may be biased and even the mathematically optimal solution can depart from the perfect matching in reality, either due to the noise observation or limited modeling capacity, or both. In fact, traditional methods are mostly based on predefined shallow affinity model with limited capacity e.g. Gaussian kernel with Euclid distance (see Eq. (2)), which has difficulty in providing enough flexibility for real-world data. This issue is partially addressed by affinity-learning based graph matching algorithms [17], [25], [26]. Along this promising direction, in this paper, a novel network based solver is proposed to directly learn Lawler's QAP whereby the affinity learning is also incorporated, as shown in Fig. 1. This approach is further extended to the case of joint matching of multiple graphs, which has been an important scenario in practice and has received wide attention in literature [8], [27], [28], [29], [30] however learning has not been considered. Also hypergraph matching is enabled in our framework.

Specifically, the proposed matching nets consist of several learnable layers as detailed in Fig. 4: 1) CNN layers taking raw images for node (and edge) feature extraction; 2) affinity metric learning layer for generating the affinity matrix i.e. the association graph; 3) vertex embedding layers using the association graph as input for vertex classification; 4) Sinkhorn net to convert the vertex score matrix into doubly-stochastic matrix. Sinkhorn technique is also adopted in the embedding module to introduce matching constraints; 5) cross-entropy loss layer whose input is the output of the Sinkhorn layer.

Note that the first two components are optional and can be treated as a plugin in the pipeline, and have nothing to do with Lawler's QAP. In contrast, the peer network [25] only allows for learning of node CNN features on images and their similarity metric i.e. the component 1) and 2). In fact, [25] is inapplicable to learning the QAP model. Our embedding also differs from [17], [31] as raw individual graphs must be required for embedding in these works. In fact, [16] shows that embedding on individual graphs can deal with some special cases of Koopmans-Beckmann's QAP, which is also a special case of Lawler's QAP as discussed above. Direct learning on Lawler's QAP enables a learning-based solver for real-world combinatorial problems beyond vision, e.g. QAPLIB instances [32], which can not be handled by previous graph matching learning algorithms.

Furthermore, we devise two generalizations to the above matching network. 1) hypergraph matching by embedding a higher-order affinity tensor; 2) multiple graph matching by devising an end-to-end compatible matching synchronization module by using the popular spectral fusion technique [29], [33]. The performance can also be boosted by adopting edge-embedding layers. The source code will be made publicly available.

The highlights of this paper are summarized as follows:

i) We show how to develop a deep network to directly tackle the (most) general graph matching formulation i.e. Lawlers Quadratic Assignment Problem beyond vision problems, in the sense of allowing the affinity matrix as the raw input. This is fulfilled by regarding the affinity matrix

TABLE 2
Summary of existing literature in learning graph matching based on different types of assignment problems solved by learning, learning modules including CNN, GNN and affinity metric, where GNN embedding is performed and loss functions. KB-QAP abbreviates Koopmans-Beckmann's QAP in Eq. (3). Lawler's QAP in Eq. (1) is the most general QAP form and higher-order assignment in Eq. (4) is its higher-order extension.

method	learned solver	multiple-graph	CNN	GNN module	embedded graph	affinity metric	loss function
Nowak *et al.* [16]	special case of KB-QAP	none	none	GCN	individual graphs	inner-product	multi-class cross-entropy
GMN [25]	none	none	VGG16	none	none	weighted exponential	pixel offset regression
Zhang *et al.* [31]	none	none	none	message-passing based CMPNN	individual graphs	inner-product	multi-class cross-entropy
PCA-GM [17]	special case of KB-QAP	none	VGG16	GCN + cross-graph conv.	individual graphs	weighted exponential	binary cross-entropy (BCE)
NGM (ours)	**Lawler's QAP (most general)**	none	VGG16	matching-aware GCN	association graph	weighted exponential	binary cross-entropy (BCE)
NGM+ (ours)	**Lawler's QAP (most general)**	none	VGG16	matching-aware edge conv.	association graph	weighted exponential	binary cross-entropy (BCE)
NHGM (ours)	**higher-order assignment**	none	VGG16	matching-aware hyper-GCN	association graph	weighted exponential	binary cross-entropy (BCE)
NMGM (ours)	**Lawler's QAP (most general)**	**end-to-end spectral method**	VGG16	matching-aware GCN	association graph	weighted exponential	binary cross-entropy (BCE)

as an association graph, whose vertices can be embedded by a deep graph neural network (GNN) [34] for classification, with a novel matching-aware graph convolution scheme. In contrast, existing works [17], [25], [26], [31] start with individual graphs' node and edge features for affinity learning instead of pairwise affinity encoded in affinity matrix.

ii) Our network solver for Lawlers QAP can be trained either in a supervised setting given ground truth node correspondence from labeled training set (e.g. for image matching), or by the final matching score without supervision (e.g. for QAPLIB problems).

iii) We extend our second-order graph matching networks to the hypergraph (third-order) matching case. This is fulfilled by building the hyperedge based association hypergraph to replace the second-order one. To our best knowledge, this is the first work for deep learning of hypergraph matching (with explicit treatment on the hyperedges).

iv) We also extend our matching network to the multiple-graph matching case by end-to-end spectral multi-graph matching, with explicit treatment for stabilized learning. To our best knowledge, there is no multiple-graph matching neural network in the existing literature.

v) Experimental results on synthetic and real-world data show the effectiveness of our devised components. The extended versions for hypergraph matching and multiple-graph matching also show competitive performance. Our model can learn with the Lawler's QAP as input while state-of-the-art graph matching networks [17], [25], [31] cannot. This allows for the evaluation of our network on the QAPLIB benchmark directly, which to our best knowledge, is the first test for network-based methods for QAPLIB.

The paper is organized as follows. Section 2 discusses the related work to graph matching and its recent learning based methods. The main approach is described in Section 3 and the experiments are introduced in Section 4. Section 5 concludes this paper.

2 Related Work

2.1 Learning-free Graph Matching Methods

Two-graph matching and QAP. Lawler's Quadratic Assignment Problem [18] is known for its application of matching two graphs by maximizing a quadratic objective function. Traveling salesman problem (TSP) and Koopmans-Beckmann's QAP are two popular variants from Lawler's QAP, with their wide range of application beyond vision, e.g. economic activities modeled by Koopmans-Beckmann's QAP [19], [32]. The most general Lawler's QAP also refers to two-graph matching in pattern recognition, and is traditionally addressed in a learning-free setting. Classically, small or medium sized problems are tractable by branch-and-bound with dual bound solved approximately [9], [13]. Modern approximate solvers [2], [7], [14], [15], [35] achieve better accuracy-speed trade-off and thus are more applicable to larger-sized problems. In two-graph matching, most methods focus on seeking approximate solution given fixed affinity model which is often set in simple parametric forms. Euclid distance in node/edge feature space together with a Gaussian kernel to derive a non-negative similarity, is widely used in the above works.

Hypergraph matching methods. Going beyond the traditional second-order graph matching, hypergraphs have been built for matching [23] and their affinity is usually represented by a tensor to encode the third-order [22], [24], [36] or even higher-order information [37]. The advantage is that the model can be more robust against noise at the cost of exponentially increased complexity for both time and space.

Multiple-graph matching methods. It has been recently actively studied for its practical utility against local noise and ambiguity. The hope is that the joint matching of multiple graphs can provide a better venue to fuse the information across graphs, leading to better robustness against local noise and ambiguity. Among the literature, a thread of works [28], [29] first generate the pairwise matching between two graphs via certain two-graph matching solvers, and then impose cycle-consistency on the pairwise matchings to improve the matching accuracy. The other line of methods impose cycle-consistency during the iterative finding of pairwise matchings and usually can achieve better results [8], [27], [38], [39]. The online setting for solving multiple graph matching is studied in [40].

Note both hypergraph or multiple graph matching paradigms try to improve the affinity model either by lifting the affinity order or imposing additional consistency regularization. As shown in the following, another possibly more effective and efficient way is adopting learning to find more adaptive affinity model parameters, or further improving the solver with deep neural networks.

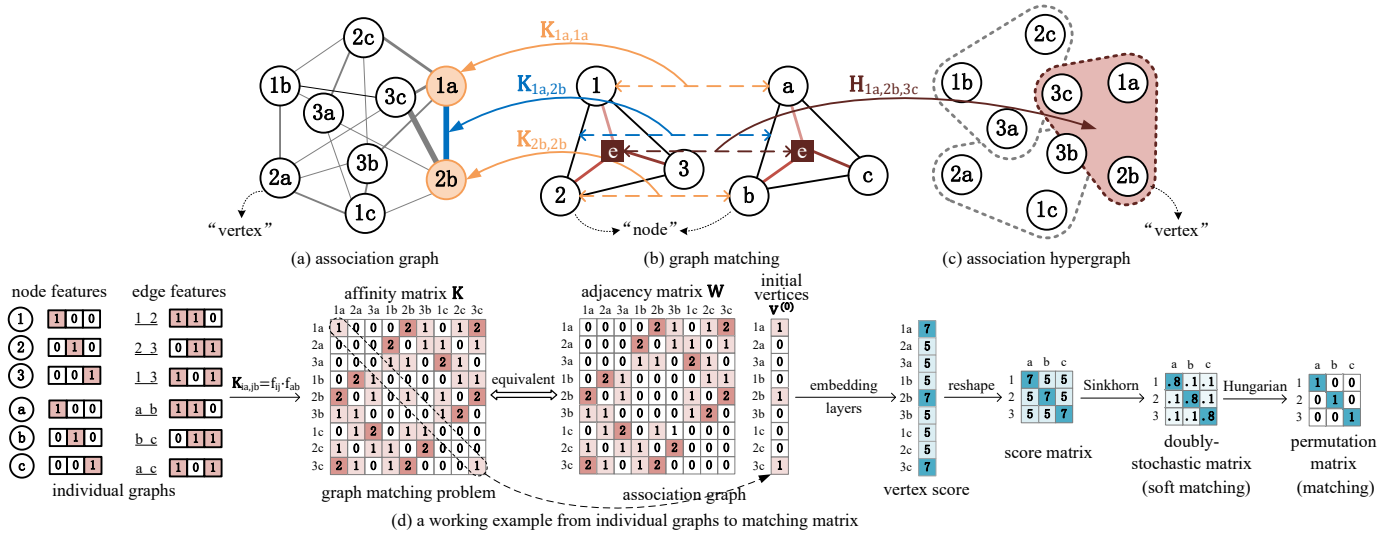


Fig. 2. Graphs with affinity matrix $\mathbf{K}$ and affinity tensor $\mathbf{H}$ in (b), w.r.t (a) association graph and (c) association hypergraph. We distinguish the nodes on association (hyper)graph and individual graphs for matching by terms *vertex* and *node* through this paper. The node-to-node matching problem in (b) can therefore be formulated as the vertex classification task on the association graph whose edge weights can be induced by the affinity matrix. Such a perspective is also widely taken in literature for graph matching e.g. [2] and hypergraph matching [24]. (d) shows a toy working example from individual graphs to final matching result: affinity matrix $\mathbf{K}$ is built from individual graphs, and the matching problem is equivalent to vertex classification on the association graph. Matching-aware embedding and vertex classification are applied on association graph to generate vertex scores, followed with reshaping and Sinkhorn normalization to obtain a double-stochastic matrix i.e. a convex hull of permutation matrix.

2.2 Learning-based Graph Matching Methods

Shallow-learning methods. The structural SVM based supervised learning method [26] incorporates earlier graph matching learning methods [41], [41], [42], [43]. Learning can also be fulfilled by unsupervised [43] and semi-supervised [44]. In these earlier works, no network is adopted until the recent seminal work [25].

Deep-learning methods. A pioneer work [16] considers the alignment of graphs by embedding on individual graphs, which can be regarded a special case of Koopmans-Beckmann's QAP. Deep learning is recently applied for graph matching on images [25], whereby convolutional neural network (CNN) is used to extract node features from images followed with spectral matching and CNN is learned using a regression-like node correspondence supervision. This work is improved by introducing GNN to encode structural [17] or geometric [31] information, with a combinatorial loss based on cross-entropy loss, and Sinkhorn network [45] as adopted in [17].

As shown in the extensive summary of network-based graph matching algorithms in Tab. 2, one shortcoming of existing graph matching networks is that they cannot directly deal with the most general Lawler's QAP form which limits their applicability to tasks when no individual graph information is available (see QAPLIB – http://anjos.mgi.polymtl.ca/qaplib/). In contrast, our approach can directly work with the affinity matrix, and we further extend to dealing with affinity tensor for hypergraph matching, as well as the setting under multiple graphs.

3 Proposed Approaches

In Sec. 3.1, we show the connection between graph matching and association graph, on which our methods are based. Then our Neural Graph Matching (NGM) network is presented in Sec. 3.2, which can solve Lawler's QAP for two-graph matching directly. In Sec. 3.3 the enhanced model NGM+ is devised by introducing edge embeddings. Also, we show the extension to hypergraph matching, i.e. Neural Hyper-Graph Matching (NHGM) in Sec. 3.4, and to multiple graph matching i.e. Neural Multi-Graph Matching (NMGM) in Sec. 3.5. All these three settings to our knowledge have not been addressed by neural network solvers before.

3.1 Preliminaries

Our models aim to match weighted graph $G^1 = (V^1, E^1)$ and $G^2 = (V^2, E^2)$ (in capital letters), where the superscript means the index of graphs and the subscript represents the index of nodes. Without loss of generality, $|V^1| = n_1 = n_{in}$ are all inlier nodes, and $|V^2| = n_2 = n_{in} + n_{out}$ contains both inliers and optional outliers. E^1, E^2 are attributed edge sets with second-order features in graphs and $|E^1| = n_{e1}, |E^2| = n_{e2}$. Lawler's QAP as given in Eq. (1) is relaxed via popular doubly-stochastic relaxation:

$$J(\mathbf{S}) = \mathrm{vec}(\mathbf{S})^\top \mathbf{K} \mathrm{vec}(\mathbf{S}), \tag{6}$$
$$\mathbf{S} \in [0,1]^{n_1 \times n_2}, \quad \mathbf{S}\mathbf{1}_{n_2} = \mathbf{1}_{n_1}, \quad \mathbf{S}^\top \mathbf{1}_{n_1} \leq \mathbf{1}_{n_2}$$

where $\mathbf{S}$ is a (partial) doubly-stochastic matrix, where all its rows sum to 1 and all its column sums are $\leq$ 1. For affinity matrix $\mathbf{K} \in \mathbb{R}^{n_1 n_2 \times n_1 n_2}$, diagonal elements $\mathbf{K}_{ia,ia} = \mathbf{s}_v(V_i^1, V_a^2)$ are first order (node) similarities and off-diagonal elements $\mathbf{K}_{ia,jb} = \mathbf{s}_e(E_{ij}^1, E_{ab}^2)$ are second order (edge) similarities, where $\mathbf{s}_v, \mathbf{s}_e$ are similarity measurements for nodes and edges, respectively.

As shown in Fig. 2, graph matching can be viewed in a perspective based on the definition of the so-called association graph $\mathcal{G}^A = (\mathcal{V}^A, \mathcal{E}^A)$ [2], [7] (in handwritten letters with superscript A). To avoid ambiguity between graphs and association graphs, we name the entities in graphs (V) as *nodes* and the entities in association graph ($\mathcal{V}^A$) as *vertices*.

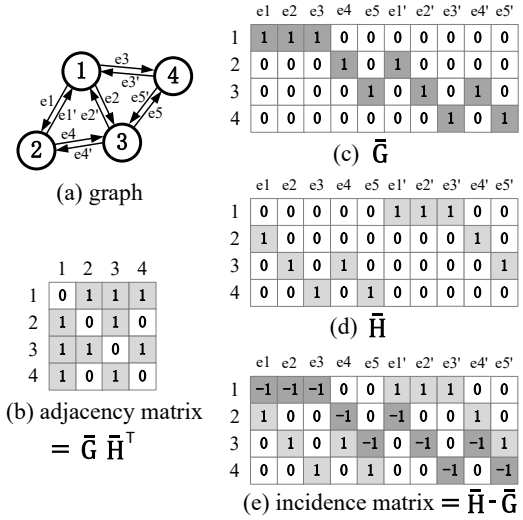


Fig. 3. A toy example of connectivity matrices (c) $\bar{\mathbf{G}}$, (d) $\bar{\mathbf{H}}$, and their connection to (a) the original graph, (b) adjacency matrix and (e) incidence matrix. All edges are directed, and $\bar{\mathbf{G}}_{i,k} = \bar{\mathbf{H}}_{j,k} = 1$ means edge k starts from node i and ends at node j. In incident matrix, -1 denotes the starting node and 1 denotes the ending node of all edges.

Readers should distinguish these two concepts as they will be repeatedly encountered through this paper.

The vertices of association graph $\mathcal{V}^A = V^1 \times V^2$ encode candidate node-to-node correspondence $\mathcal{V}^A_{ia} = (V^1_i, V^2_a)$ corresponding to the matching matrix $\mathbf{X}_{i,a}$, therefore the vectorized assignment matrix $\mathrm{vec}(\mathbf{X})$ is equivalent to the vertex set of association graph. The edges $\mathcal{E}^A$ represent the agreement between two pairs of correspondence $\mathcal{E}^A_{ia,jb} = \{(V^1_i, V^2_a), (V^1_j, V^2_b)\}$ modeled by $\mathbf{K}_{ia,jb}$, so that the off-diagonal part of affinity matrix $\mathbf{K}$ is equivalent to the adjacency matrix of association graph. The matching between two graphs can therefore be transformed into vertex classification on the association graph, following [2], [7]. In this paper, diagonal elements $\mathbf{K}_{ia,ia}$ are further assigned as vertex attributes $\mathcal{V}^A$, to better exploit the first-order similarities. Such formulation can also be generalized to hypergraph matching problems, with edges replaced by hyperedges, as shown in Fig. 2(c). The association graph prohibits links that violate the one-to-one matching constraint (e.g. there is no link between vertex '1a' and '1c' in Fig. 2(a)).

3.2 NGM: Neural Graph Matching for QAP

Neural Graph Matching (NGM) solves relaxed Lawler's QAP in Eq. (6), by vertex classification via Graph Convolutional Networks (GCN) [46] with novel matching-aware embedding modules. The vertex classification is performed on the association graph induced by the affinity matrix, followed by a Sinkhorn operator. As shown by existing learning-free graph matching solvers [2], [7], graph matching problem is equivalent to vertex classification on the association graph. NGM accepts either raw image (with jointly learned CNN and affinity metric), or affinity matrix (without CNN or affinity metric), and learns end-to-end from ground truth correspondence or by self-supervision for QAPLIB problems.

3.2.1 Affinity matrix building from natural images

Our graph matching net allows either affinity matrix or raw images as input. The image processing module is optional and treated as a plug-in for dealing with images, following the protocol in [25] whereby the affinity matrix is built from pre-given keypoints in images. As shown in the upper half of Fig. 4, image features are extracted by learnable CNN layers such as VGG16 [47]. Given two input images with labeled keypoints, we adopt CNN layers to extract per-node features $\bar{\mathbf{F}}^1, \bar{\mathbf{U}}^1 \in \mathbb{R}^{n_1 \times d}$ for G^1 and $\bar{\mathbf{F}}^2, \bar{\mathbf{U}}^2 \in \mathbb{R}^{n_2 \times d}$ for G^2, where d is feature dimension size and $\bar{\mathbf{F}}, \bar{\mathbf{U}}$ are extracted from different CNN layers (e.g. VGG16 `relu5_1` for $\bar{\mathbf{F}}$ and `relu4_2` for $\bar{\mathbf{U}}$) and utilized for edge representation and node representation, respectively. Features are obtained by bi-linear interpolation on the CNN feature map. As shown in Fig. 3, the connectivity of two graphs are represented by $\bar{\mathbf{G}}^1, \bar{\mathbf{H}}^1 \in \{0,1\}^{n_1 \times n_{e1}}$ and $\bar{\mathbf{G}}^2, \bar{\mathbf{H}}^2 \in \{0,1\}^{n_2 \times n_{e2}}$, where $\bar{\mathbf{A}}^1 = \bar{\mathbf{G}}^1 \bar{\mathbf{H}}^{1\top}, \bar{\mathbf{A}}^2 = \bar{\mathbf{G}}^2 \bar{\mathbf{H}}^{2\top}$ are the adjacency matrices of two graphs, and $\bar{\mathbf{G}}_{i,k} = \bar{\mathbf{H}}_{j,k} = 1$ means edge k links node i to node j. The edge representations are built by concatenating node features at both ends of the edge:

$$\bar{\mathbf{X}} = [\bar{\mathbf{G}}_1^\top \bar{\mathbf{F}}_1 \quad \bar{\mathbf{H}}_1^\top \bar{\mathbf{F}}_1], \ \bar{\mathbf{Y}} = [\bar{\mathbf{G}}_2^\top \bar{\mathbf{F}}_2 \quad \bar{\mathbf{H}}_2^\top \bar{\mathbf{F}}_2] \tag{7}$$

where $[\ \cdot\ \ \cdot\ ]$ means concatenating two matrices along columns. The node-to-node similarity matrix $\mathbf{K}_p \in \mathbb{R}^{n_1 \times n_2}$ and edge-to-edge similarity $\mathbf{K}_e \in \mathbb{R}^{n_{e1} \times n_{e2}}$ are built via

$$\mathbf{K}_e = \bar{\mathbf{X}} \mathbf{\Lambda} \bar{\mathbf{Y}}^\top, \mathbf{K}_p = \bar{\mathbf{U}}^1 \bar{\mathbf{U}}^{2\top} \tag{8}$$

where $\mathbf{\Lambda} \in \mathbb{R}^{2d \times 2d}$ is the learnable parameter for affinity metric. The QAP affinity matrix is built following the factorized formulation of $\mathbf{K}$ [48]:

$$\mathbf{K} = \mathrm{diag}(\mathrm{vec}(\mathbf{K}_p)) + (\bar{\mathbf{G}}_2 \otimes_{\mathcal{K}} \bar{\mathbf{G}}_1) \mathrm{diag}(\mathrm{vec}(\mathbf{K}_e)) (\bar{\mathbf{H}}_2 \otimes_{\mathcal{K}} \bar{\mathbf{H}}_1)^\top \tag{9}$$

where $\mathrm{diag}(\cdot)$ means building a diagonal matrix from input vector, and $\otimes_{\mathcal{K}}$ means Kronecker product. All the forementioned operations allow back propagation, and we adopt the efficient GPU implementation provided by [17].

3.2.2 Association graph construction

We derive the association graph that contains vertices and edges, from the affinity matrix $\mathbf{K}$. The weighted adjacency matrix of association graph $\mathbf{W}$ comes from the off-diagonal elements of $\mathbf{K}$. We denote $\mathbf{v}^{(k)} \in \mathbb{R}^{n_1 n_2 \times l_k}$ as l_k-dimensional vertex embeddings on layer k (starting with $k = 0$). Initial embeddings are scalar, i.e. $l_0 = 1$, taken from the diagonal of $\mathbf{K}$.

$$\mathbf{W}_{ia,jb} = \mathbf{K}_{ia,jb\ (ia \neq jb)}, \quad \mathbf{v}^{(0)}_{ia} = \mathbf{K}_{ia,ia} \tag{10}$$

$\mathbf{W}$ contains both connectivity and weight information in the association graph. In case when the first-order similarity $\mathbf{K}_{ia,ia}$ is absent, we can assign a constant (e.g. 1) for all $\mathbf{v}^{(0)}$.

3.2.3 Matching aware embedding of association graph

The matching problem can be transformed to finding the vertices in the association graph that encode the node-to-node correspondence between two input graphs, as illustrated in Fig. 2. Specifically for vertex classification on the association graph, we use GCN [46] for its effectiveness and simplicity. Firstly, based on (unweighted) adjacency matrix

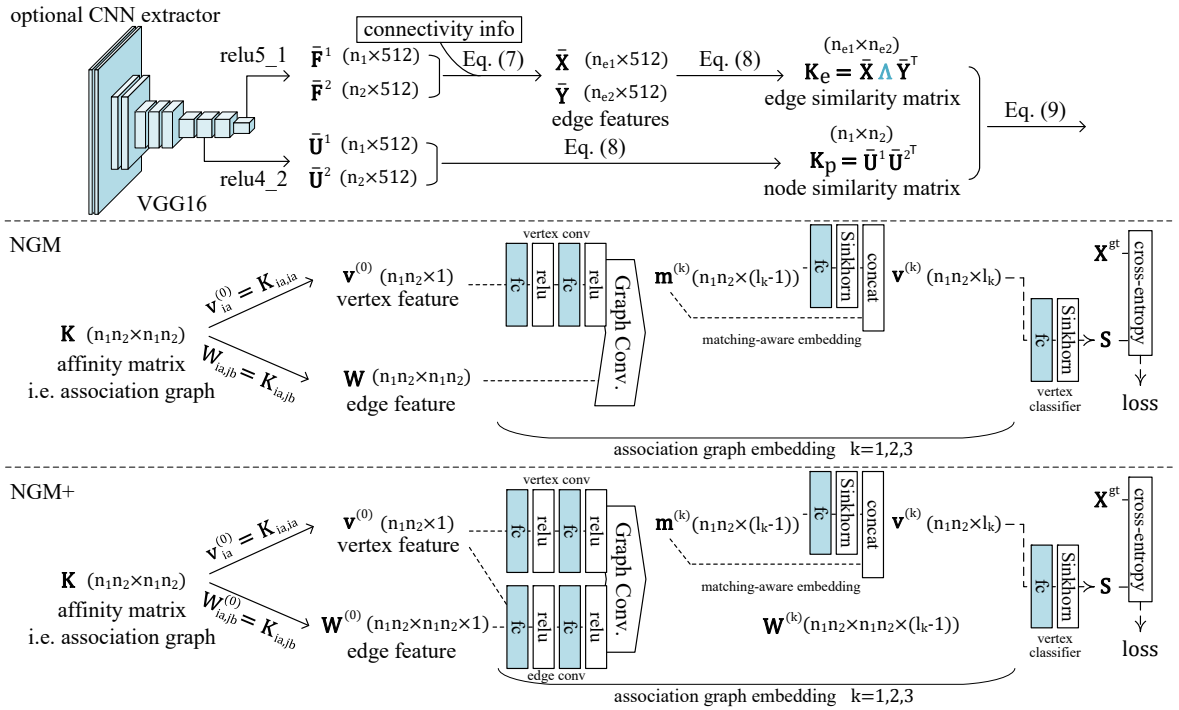


Fig. 4. The proposed NGM & NGM+ architecture for two-graph matching. The components with blue FC layers i.e. vertex convolution, edge convolution (in NGM+), matching-aware embedding module and vertex classifier are jointly learned with optional CNN and similarity metric $\mathbf{\Lambda}$.

of association graph $\mathbf{A} \in \{0,1\}^{n_1 n_2 \times n_1 n_2}$: $\mathbf{A}_{ia,jb} = 1$ if $\mathbf{K}_{ia,jb} > 0$ and otherwise 0. Since $\mathbf{A}$ is symmetric, we compute the degree matrix for normalization:

$$\mathbf{D} = \mathrm{diag}(\mathbf{A}\mathbf{1}_{n_1 n_2}) \tag{11}$$

where $\mathrm{diag}(\cdot)$ builds a diagonal matrix from input vector. The vertex aggregation step is according to:

$$\mathbf{m}^{(k)} = \mathbf{D}^{-1}\mathbf{W} f_m(\mathbf{v}^{(k-1)}) + f_v(\mathbf{v}^{k-1}),\ \mathbf{v}^{(k)} = \mathbf{m}^{(k)} \tag{12}$$

where the message passing function $f_m : \mathbb{R}^{l_{k-1}} \to \mathbb{R}^{l_k}$ and vertex's self update function $f_v : \mathbb{R}^{l_{k-1}} \to \mathbb{R}^{l_k}$ are both implemented by networks with two fully-connected layers and ReLU activation.

The above general vanilla vertex embedding procedure in Eq. (12) does not consider the one-to-one assignment constraint for matching. Here we develop a matching constraint aware embedding model: in each layer a soft permutation (i.e. doubly-stochastic matrix) is scored via classifier with Sinkhorn network $\mathrm{Classifier} : \mathbb{R}^{n_1 n_2 \times l_k} \to [0,1]^{n_1 \times n_2}$ (see discussions in Sec. 3.2.4) followed by vectorization operator $\mathrm{vec}(\cdot)$. The predicted soft permutation is concatenated to vertex embeddings whereby matching information is considered in embedding layers. With $f_m, f_v : \mathbb{R}^{l_{k-1}} \to \mathbb{R}^{(l_k - 1)}$, such a matching-aware embedding scheme is denoted as **Sinkhorn embedding** in the rest of the paper.

$$\mathbf{v}^{(k)} = [\mathbf{m}^{(k)} \quad \mathrm{vec}(\mathrm{Classifier}(\mathbf{m}^{(k)}))] \tag{13}$$

where $[\ \cdot\ \ \cdot\ ]$ means concatenation. We experiment both vanilla vertex embedding in Eq. (12) and matching-aware Sinkhorn embedding in Eq. (13), to validate the necessity of adding assignment constraint.

3.2.4 Vertex classification with Sinkhorn network

As graph matching is equivalent to vertex classification on association graph (see Fig. 2), a vertex classifier with Sinkhorn network is adopted to predict the matching result. Specifically we use a single layer fully-connected classifier denoted by $f_c : \mathbb{R}^{l_k} \to \mathbb{R}$, followed by exponential activation with regularization factor α:

$$\mathbf{s}_{ia}^{(k)} = \exp\left(\alpha f_c(\mathbf{v}_{ia}^{(k)})\right) \tag{14}$$

After reshaping classification scores into $\mathbb{R}^{n_1 \times n_2}$, one-to-one assignment constraint is enforced to $\mathbf{s}$ by Sinkhorn network [45], [49]. It takes a non-negative square matrix as input and outputs a doubly-stochastic matrix [45], [50]. As the scoring matrix can be non-square for different sizes of graphs, the input matrix $\mathbf{S} \in \mathbb{R}^{n_1 \times n_2}$ is padded into a square one (assume $n_1 \le n_2$) with small elements e.g. $\epsilon = 10^{-3}$. A doubly-stochastic matrix is obtained by repeatedly running:

$$\mathbf{S} = \mathbf{S} \oslash (\mathbf{1}_{n_2}\mathbf{1}_{n_2}^\top \cdot \mathbf{S}),\ \mathbf{S} = \mathbf{S} \oslash (\mathbf{S} \cdot \mathbf{1}_{n_2}\mathbf{1}_{n_2}^\top) \tag{15}$$

where $\oslash$ means element-wise division. By taking column-normalization and row-normalization in Eq. (15) alternatively, $\mathbf{S}$ converges to a doubly-stochastic matrix whose rows and columns all sum to 1. The dummy elements are discarded in the final output, whose column sum may be < 1 given umatched nodes from the bigger graph. Sinkhorn operator is fully differentiable and can be efficiently implemented by automatic differentiation techniques [51]. The proposed vertex classifier with Sinkhorn network is denoted as $\mathrm{Classifier} : \mathbb{R}^{n_1 n_2 \times l_k} \to [0,1]^{n_1 \times n_2}$, which is also mentioned in matching-aware Sinkhorn embedding in Eq. (13).

3.2.5 Loss for end-to-end training

Recall the obtained predicted matrix $\mathbf{S}$ from the above procedure is a doubly-stochastic matrix. Each element can

be regarded as a binary classification where each vertex should be classified to 1 (matched) or 0 (unmatched). Hence we adopt the binary cross-entropy as the final loss, given the ground truth node-to-node correspondence $\mathbf{X}^{gt}$:

$$\ell = -\sum_{i=1}^{n_1}\sum_{a=1}^{n_2} \mathbf{X}^{gt}_{i,a} \log \mathbf{S}_{i,a} + (1-\mathbf{X}^{gt}_{i,a}) \log(1-\mathbf{S}_{i,a}) \quad (16)$$

Our approach also allows self-supervised learning over optimization objectives for QAPLIB problems, which will be discussed in Sec. 4.2 in details.

All the components are differentiable, hence NGM, including optional CNN for keypoint feature extraction from images, is learned via backpropagation and gradient descent. We further enable NGM with edge embedding.

3.3 NGM+: Improved NGM with Edge Embedding

Edge embedding has been confirmed effective to enhance vertex embedding learning [52] and we improve the model capacity of NGM with additional edge embeddings, resulting in the enhanced method called NGM+.

3.3.1 Edge embedding update

We extend GCN [46] with (l_k-1)-dimensional edge embedding $\mathbf{W}^{(k)} \in \mathbb{R}^{n_1n_2\times n_1n_2\times(l_k-1)}$ on layer k, whose feature is updated from features of the same edge and its adjacent nodes. Initial edge embeddings are scalar, taken from off-diagonal elements of $\mathbf{K}$:

$$\mathbf{W}^{(0)}_{ia,jb} = \mathbf{K}_{ia,jb\ (ia\neq jb)} \quad (17)$$

Our method starts with edge feature updating from the edge and its adjacent nodes in the previous layer, then concatenated and passed by edge update function $f_e(\cdot)$:

$$\mathbf{W}^{(k)}_{ia,jb} = f_e([\mathbf{W}^{(k-1)}_{ia,jb} \quad \mathbf{v}^{(k-1)}_{jb}])_{\forall ia,jb\in\mathbf{K}_{ia,jb}>0} \quad (18)$$

$f_e : \mathbb{R}^{2l_{k-1}} \to \mathbb{R}^{(l_k-1)}$ is a neural network containing two fully-connected layers with ReLU activation. By enforcing $ia, jb \in \mathbf{K}_{ia,jb} > 0$, only the edges that originally with positive attributes in association graph are updated.

3.3.2 Channel-wise embedding of association graph

Similar to [52], along the third dimension, i.e. feature channels of $\mathbf{W}^{(k)}$, edge embeddings are regarded as channel-wise aggregation weights for vertex features. The feature aggregation step is performed as follows:

$$\begin{aligned}\mathbf{m}^{(k)} &= (\mathbf{D}^{-1})_3 \circ \mathbf{W}^{(k)} \circ f_m(\mathbf{v}^{(k-1)}) + f_v(\mathbf{v}^{(k-1)}) \\ \mathbf{v}^{(k)} &= \left[\mathbf{m}^{(k)} \quad \mathrm{vec}(\mathrm{Classifier}(\mathbf{m}^{(k)}))\right]\end{aligned} \quad (19)$$

where $(\mathbf{D}^{-1})_3$ means expanding the inversed degree matrix $\mathbf{D}^{-1}$ along third dimension and $\circ$ denotes matrix multiplication split over feature channels: $(\mathbf{A}\circ\mathbf{B})_{i,j,k} = \sum_p \mathbf{A}_{i,p,k}\mathbf{B}_{p,j,k}$. $f_m, f_v : \mathbb{R}^{l_{k-1}} \to \mathbb{R}^{(l_k-1)}$ are two fully-connected layers with ReLU activation. We also adopt Sinkhorn embedding in NGM+. The other compoments of NGM+ are consistent with NGM in Sec. 3.2.

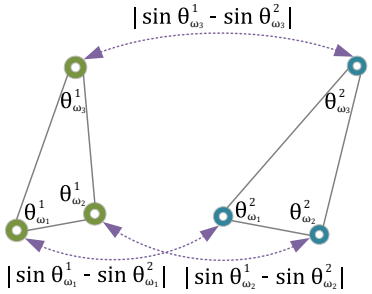


Fig. 5. Third-order affinity adopted in hypergraph matching. It in general follows the previous hypergraph matching works [22], [24].

3.4 NHGM: Neural Hypergraph Matching

For Neural Hyper-Graph Matching (NHGM), the higher-order structure is exploited for more robust correspondence prediction. NHGM owns a nearly identical pipeline compared to NGM, while a more general message-passing scheme is devised for feature aggregation in hypergraphs, as previously shown in [53], [54]. Due to the explosive computational cost ($O((n_1n_2)^t)$ with order t), here we limit hypergraph to third-order which is also in line with the majority of existing works [22], [24], while the scheme is generalizable to any order t.

3.4.1 Association hypergraph construction

The second-order affinity matrix $\mathbf{K}$ is generalized to affinity tensor $\mathbf{H}^{\langle t\rangle}$ of order t in hypergraph matching. In line with the hypergraph matching literature [21], [23], [24], the third-order affinity tensor is specified as:

$$\mathbf{H}^{\langle 3\rangle}_{\omega_1,\omega_2,\omega_3} = \exp\left(-\left(\sum_{q=1}^{3} |\sin\theta^1_{\omega_q} - \sin\theta^2_{\omega_q}|\right)/\sigma_3^2\right) \quad (20)$$

where $\theta^1_{\omega_q}, \theta^2_{\omega_q}$ denotes the angle in graph $\mathcal{G}_1$ and $\mathcal{G}_2$ of each correspondence ω_q, respectively. An illustration of third order affinity can be found in Fig. 5, where the similarity between triplets of nodes is compared. Third order affinity is usually defined on geometric consistency and it preserves both scaling and rotation invariance.

Extending from the second-order association graph, a hyper association graph is constructed from $\mathbf{H}$. The association hypergraph $\mathcal{H}^A = (\mathcal{V}^A, \mathcal{E}^A)$ takes node-to-node correspondence $\omega = (V^1_i, V^2_j)$ as vertices $\mathcal{V}^A$ (which is consistent with second-order association graph) and higher-order similarity among $\{(V^1_{\omega_1}, V^2_{\omega_1}), \cdots, (V^1_{\omega_t}, V^2_{\omega_t})\}$ as hyperedges $\mathcal{E}^a$, as shown by Fig. 2(c). Elements of $\mathbf{H}$ are adjacency weights for the association hypergraph accordingly. In NHGM, hyperedge feature aggregation is defined for hypergraph matching.

3.4.2 Matching aware association hypergraph embedding

As an extension of Eq. (12), vertex embeddings are updated from all vertices linked by hyperedges in association hypergraph. We compute normalized degree tensor of order t:

$$\mathbf{D}^{\langle t\rangle\ -1}_{\omega_1,\cdots,\omega_t} = \mathbf{A}^{\langle t\rangle}_{\omega_1,\cdots,\omega_t} / (\mathbf{A}^{\langle t\rangle} \otimes_2 \mathbf{1} \cdots \otimes_t \mathbf{1})_{\omega_1} \quad (21)$$

Then an aggregation scheme extended from Eq. (12) is taken:

$$\begin{aligned}\mathbf{p}^{(k)} &= f_m^{\langle t\rangle}(\mathbf{v}^{(k-1)}),\ \mathbf{H}^{\langle t\rangle\prime} = (\mathbf{D}^{\langle t\rangle\ -1} \odot \mathbf{H}^{\langle t\rangle})_{t+1} \\ \mathbf{m}^{(k)} &= \sum_t \lambda_t\ \mathbf{H}^{\langle t\rangle\prime} \otimes_t \mathbf{p}^{(k)} \cdots \otimes_2 \mathbf{p}^{(k)} + f_v \\ \mathbf{v}^{(k)} &= \left[\mathbf{m}^{(k)} \quad \text{vec}(\text{Classifier}(\mathbf{m}^{(k)}))\right]\end{aligned} \tag{22}$$

where f_v abbreviates $f_v(\mathbf{v}^{(k-1)})$, $\otimes_i$ denotes tensor product by dimension i (see Eq. (5)), $\odot$ denotes element-wise multiply, $(\cdot)_{t+1}$ means expanding along dimension $(t+1)$. $f_m^{\langle t\rangle} : \mathbb{R}^{l_{k-1}} \to \mathbb{R}^{(l_k-1)}$ is message passing function at order t. Different orders of features are fused by weighted summation with λ_t.

The other modules of NHGM, including classifier and cross-entropy loss, are identical to NGM. Therefore, NGM can be viewed as a special case of NHGM, where the order is restricted to 2. The sparsity of $\mathbf{H}$ is exploited for efficient implementation. Note that the edge embedding scheme introduced in Sec. 3.3 is also applicable to NHGM, but we stick to the simple version for cost-effectiveness.

3.5 NMGM: Neural Multi-graph Matching

We further explore learning multi-graph matching, where matching information is fused among multiple graphs by so-called cycle-consistency. Cycle-consistency denotes a condition where the matching result between any two graphs is consistent when passed through any other graphs, i.e. $\mathbf{X}_{ij} = \mathbf{X}_{ik}\mathbf{X}_{kj}$ for all i, j, k, which can be viewed as each graph matched to a n-sized reference $\mathbf{X}_i \in \{0,1\}^{n\times n}$. The pariwise matching between G_i, G_j can be represented with $\mathbf{X}_{ij} = \mathbf{X}_i\mathbf{X}_j^\top$. In this paper, we refer to the line of works involving post-synchronization given initial pairwise matchings [28], [29], [33], [55]. Spectral fusion is adopted in NMGM for its effectiveness and simplicity, and most importantly, its capability in end-to-end training. We assume all graphs are of equal size and consider bijection.

3.5.1 Building joint matching matrix

We first obtain initial two-graph matchings by NGM to build a symmetric joint matching matrix $\mathcal{S} \in \mathcal{R}^{nm\times nm}$. For each pair G_i and G_j with n nodes, $\mathbf{S}_{ij} \in [0,1]^{n\times n}$ is computed by NGM as the soft (i.e. doubly-stochastic) matching matrix. For m graphs, $\mathcal{S}$ can be built from all combinations of pairwise matchings $\mathbf{S}_{ij}$:

$$\mathcal{S} = \begin{pmatrix} \mathbf{S}_{00} & \cdots & \mathbf{S}_{0m} \\ \vdots & \ddots & \vdots \\ \mathbf{S}_{m0} & \cdots & \mathbf{S}_{mm} \end{pmatrix} \tag{23}$$

where $\mathcal{S}$ is of size $mn \times mn$. For the diagonal part of $\mathcal{S}$, $\mathbf{S}_{ii}$ are all identical matrices. Note $\mathbf{S}_{ij}$ are all square matrices. The objective of spectral fusion results in getting a cycle-consistent joint matching matrix $\hat{\mathcal{S}}$, whose innerproduct distance against $\mathcal{S}$ is minimized:

$$\min_{\hat{\mathcal{S}}} = \text{tr}(\hat{\mathcal{S}}^\top \mathcal{S}) \tag{24}$$

where it holds $\hat{\mathbf{S}}_{ij} = \hat{\mathbf{S}}_{ik}\hat{\mathbf{S}}_{kj}$ for all elements in $\hat{\mathcal{S}}$. Since cycle-consistency holds in $\hat{\mathcal{S}}$, it can be decomposed as matching to the reference:

$$\begin{aligned}\hat{\mathbf{U}}\hat{\mathbf{U}}^\top &= \hat{\mathcal{S}} \\ \text{where} \quad \hat{\mathbf{U}} &= \begin{pmatrix} \hat{\mathbf{S}}_0 \\ \vdots \\ \hat{\mathbf{S}}_m \end{pmatrix}\end{aligned} \tag{25}$$

under ideal condition where $\hat{\mathbf{S}}_i$ are all permutation matrices, each column of $\hat{\mathbf{U}}$ is linearly independent and $\hat{\mathbf{U}}/\sqrt{m}$ are n eigenvectors of $\hat{\mathcal{S}}$ with eigenvalue m. The permutation constraint in $\hat{\mathbf{S}}_i$ is relaxed for computational feasibility and Eq. (24) results in a generalized Rayleigh problem and solved via spectral fusion, as shown follows.

3.5.2 Differentiable spectral fusion of pairwise matchings

Multi-graph matching information can be fused by eigenvector decomposition (i.e. spectral method) on $\mathcal{S}$. Based on generalized Rayleigh problem, given n nodes in each graph, we extract the eigenvectors corresponding to the top-n eigenvalues of symmetric matrix $\mathcal{S}$:

$$\mathbf{U}\ \mathbf{\Sigma}\ \mathbf{U}^\top = \mathcal{S} \tag{26}$$

where diagonal matrix $\mathbf{\Sigma} \in \mathbb{R}^{n\times n}$ contains top-n eigenvalues and $\mathbf{U} \in \mathbb{R}^{mn\times n}$ are the n corresponding eigenvectors. It has been shown that the computation of eigenvalues and eigenvectors are differentiable [56] which makes them fixed components in our end-to-end learning network pipeline. The fusion of the input $\mathcal{S}$ can be written as follows, which can be seen as a smoothed version minimizing $\text{tr}(\hat{\mathcal{S}}^\top\mathcal{S})$:

$$\hat{\mathcal{S}} = m\ \mathbf{U}\ \mathbf{U}^\top \tag{27}$$

Te gradient of eigen decomposition in Eq. (26) is [56]:

$$\begin{aligned}\frac{\partial L}{\partial \mathcal{S}} &= \mathbf{U}\left(4m\left(\mathbf{Y}^\top \odot \left(\mathbf{U}^\top\left(\frac{\partial L}{\partial \hat{\mathcal{S}}}\right)_{sym}\mathbf{U}\right)_{sym}\right)\right)\mathbf{U}^\top \\ \text{where} \quad \mathbf{Y}_{ij} &= \begin{cases} 1/(\sigma_i - \sigma_j) & i \neq j \\ 0 & i = j \end{cases}\end{aligned} \tag{28}$$

where L denotes the loss, $\odot$ means element-wise multiplication, $\mathbf{A}_{sym} = (\mathbf{A} + \mathbf{A}^\top)/2$ and $\sigma_i = \mathbf{\Sigma}_{ii}$ is the i-th eigenvalue. According to this backward formulation, if there exist non-distinctive eigenvalues, i.e. $\sigma_i = \sigma_j$ for $i \neq j$, a numerical divided-by-zero error will be caused. This issue usually happens when cycle-consistency (i.e. $\mathbf{S}_{ij} = \mathbf{S}_{ik}\mathbf{S}_{kj}$) is already met in the input $\mathbf{S}$, under such circumstances the fused matching results are nearly identical to the original matchings. To avoid numerical issues, we assign $\hat{\mathbf{S}}_{ij} = \mathbf{S}_{ij}$ to bypass eigendecomposition if the minimum residual among top-n eigenvalues is smaller than tolerance δ, e.g. 10^{-4}. This strategy is found effective to stabilize learning.

Final matching results are obtained by differentiable Sinkhorn network:

$$\bar{\mathbf{S}}_{ij} = \text{Sinkhorn}(\exp(\hat{\alpha}\ \hat{\mathbf{S}}_{ij})) \tag{29}$$

where the fused two-graph matching $\hat{\mathbf{S}}_{ij}$ is from $\hat{\mathcal{S}}$ and $\exp(\hat{\alpha}\ \cdot\ )$ performs regularization for Sinkhron. Cross-entropy loss in Eq. (16) is applied to each $\bar{\mathbf{S}}_{ij}$ for supervised learning, which is similar to the supervised two-graph matching case in the paper.

TABLE 3
Notation of all our proposed methods and their variants.

model	capability	description
NGM	QAP	Neural Graph Matching model introduced in Sec. 3.2.
NGM+	QAP	enhanced NGN with edge embedding in Sec. 3.3.
NHGM	hypergraph matching	Neural Hyper-Graph Matching model introduced in Sec. 3.4.
NMGM	multi-graph matching	Neural Multi-Graph Matching model introduced in Sec. 3.5.
NGM-V	QAP	NGM by replacing Sinkhorn embedding in Eq. (13) with vanilla embedding in Eq. (12).
NGM-MH	QAP	NGM model with multi-head Sinkhorn embedding (8 multi-head channels).
NGM-SF	multi-graph matching	learned NGM model followed with learning-free spectral fusion.
NGM-GX	QAP	sample X times by Gumbel-Sinkhorn and select the solution with the best objective.

3.6 Further Discussions

3.6.1 Learning of problem structure

The inherent working pattern of our proposed neural solver actually learns the underlying structure of graph matching problems. With the connection between graph matching problem and association graph, the original mathematical form of Lawler's QAP transforms into a trackable structure with modern deep learning models. The problem structure is learned by GNN, resulting in a simplified Linear Assignment Problem solved with differentiable Sinkhorn algorithm. In [57] some combinatorial problems over graphs are considered, where problems simplified by GNN are solved greedily. A similar scheme should generalize to other combinatorial problems, by exploiting the representative power of learning problem structures by deep learning.

3.6.2 Matching-aware embedding

The matching-aware Sinkhorn embedding is proposed to add one-to-one matching constraint at shallower embedding layers. Otherwise, the matching constraint is not considered until the output Sinkhorn layer. Early involvement of matching information has been proven effective for both learning-free (RRWM [2] vs SM [7]) and learning based (PCA-GM vs PIA-GM [17]) methods. We show the importance of Sinkhorn embedding by notable improvement in synthetic tests and ablation study on real-world images, additionally further improvement by introducing multi-head Sinkhorn embedding at the cost of increased computation.

3.6.3 Gumbel sampling for optimization problems

As a common post-processing step, the gap between doubly stochastic matrix $\mathbf{S}$ and permutation matrix $\mathbf{X}$ is fulfilled by Hungarian algorithm [35] in a deterministic manner. From the probabilistic point of view, $\mathbf{S}$ represents a distribution on the space of permutation matrices, and our cross-entropy loss minimizes the distance between probability $\mathbf{S}$ and ground truth distribution $\mathbf{X}^{gt}$. Permutation with the highest probability is selected by Hungarian algorithm.

Such a greedy scheme is empirically successful for matching. However, there might exist better solutions from the distribution, especially when minimizing the objective of combinatorial problems. Therefore, we switch to Gumbel-Sinkhorn [49] by replacing Eq. (14) with

$$\mathbf{s}_{ia}^{(k)} = \exp\Big(\alpha_g(f_c(\mathbf{v}_{ia}^{(k)}) + g)\Big) \tag{30}$$

followed by Sinkhorn algorithm. Note the added g is sampled from standard Gumbel distribution with cumulative distribution function (CDF):

$$G(x) = e^{-e^{-x}} \tag{31}$$

which models the distribution of extreme values from another distribution. By Eq. (30) sparser doubly-stochastic matrices can be sampled from the original distribution and repeated sampling provides a batch of samples. These sparse matrices are followed by Hungarian discretization, and the objective scores are computed and the best-performing solution is chosen as the final solution. Exploration and speed can be balanced by the number of Gumbel samples.

4 Experiments

Experiments are conducted on a Linux workstation with Nvidia RTX8000 (48GB) GPU and Intel Xeon W-3175X CPU @ 3.10GHz with 128GB RAM.

We test our methods for Lawler's QAP in two settings: i) synthetic point registration, which takes affinity matrix/tensor as input, and ii) QAPLIB with large-scale real-world QAP instances where the network learns to minimize the objective score. We also test our methods on Koopmans-Beckmanns QAP in the sense that CNN features of image keypoints are learned and matched on real images. For keypoint matching, matching accuracy is computed as the percentage of correct matchings among all true matchings.

We also perform hypergraph and multiple graph matching tests to evaluate our NHGM and NMGM. Our PyTorch implementation of NGM, NHGM, NMGM and NGM+ involves a three-layer GNN, with graph feature channels $l_1 = l_2 = l_3 = 16$. Other hyperparameters are set as $\alpha = \hat{\alpha} = 20, \lambda_2 = 1, \lambda_3 = 1.5$. We set batch size=8. All methods are trained with SGD and 0.9 Nesterov momentum [58]. Detailed learning rate configurations can be found in the following part of this section. Tab. 3 summarizes all our methods and their variants.

4.1 Synthetic Experiment for QAP Learning

4.1.1 Protocol setting

In the synthetic experiment, sets of random points in the 2D plane are matched by comparison with other competitive learning-free graph matching solvers. For each trial, we generate 10 sets of ground truth points whose coordinates are in the plane $U(0,1) \times U(0,1)$. Synthetic points are distorted by random scaling from $U(1-\delta_s, 1+\delta_s)$ and additive random noise $N(0, \sigma_n^2)$. From each ground truth set, 200 graphs are sampled for training and 100 for testing, resulting in totally 2,000 training samples and 1,000 testing samples in one trial. We assume graph structure is unknown to the GM solver, therefore we construct the reference graph by Delaunay triangulation, and the target graph (may contain outliers) is fully connected. Outliers are also randomly sampled from

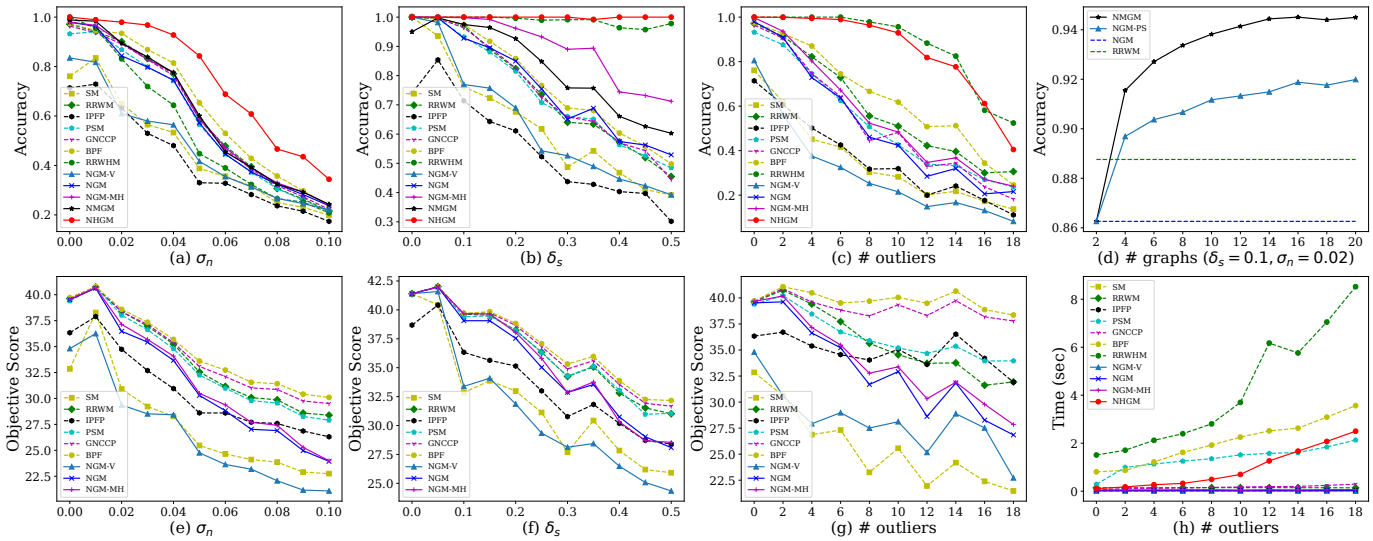


Fig. 6. Synthetic test by varying deformation level σ_n, δ_s, number of outliers/graphs. Note it learns a QAP solver based on the given affinity matrix/tensor, without learning any affinity model. This feature is not supported in GMN [25] and PCA-GM [17] thus they cannot be compared.

$U(0,1) \times U(0,1)$. By default there are 10 inliers without outlier, with $\delta_s = 0.1, \sigma_n = 0$. We construct the same affinity matrix to formulate Lawler's QAP for all methods.

4.1.2 Peer methods

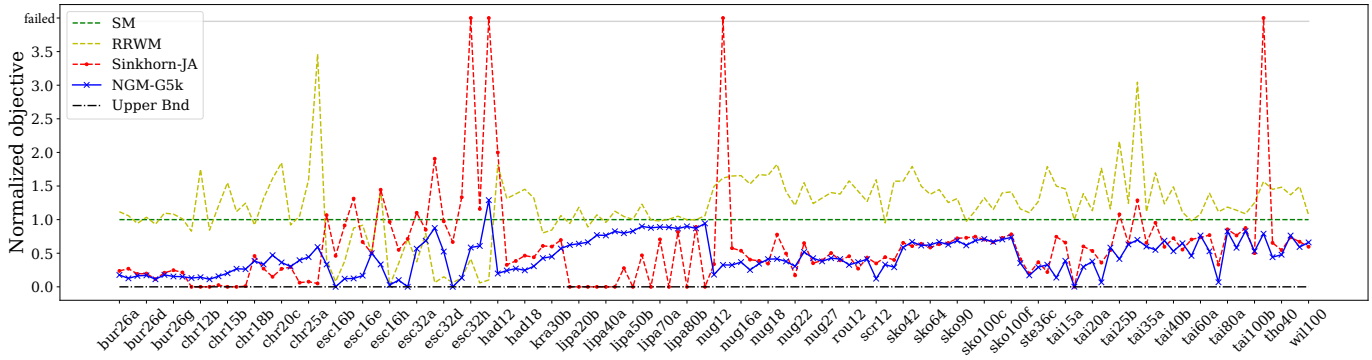
As existing learning methods [17], [25], [31] cannot handle learning with Lawler's QAP with a given affinity matrix, we compare popular learning-free methods: **1) SM** [7] considers graph matching as discovering graph cluster by spectral numerical technique; **2) RRWM** [2] adopts a random-walk view with reweighted jump on graph matching; **3) IPFP** [59] iteratively improves a given solution via integer projection; **4) PSM** [60] improves SM through a probabilistic view; **5) GNCCP** [61] is a novel convex-concave path-following algorithm for graph matching and **6) BPF** [14] improves path following techniques by branch switching, reaching state-of-the-art performance on graph matching. Additionally, hyper-graph matching algorithm **7) RRWHM** [24] extending powerful RRWM to hyper-graph scenarios is also compared. Second-order affinity is integrated into third-order tensor for RRWHM following [24]. In this experiment, second-order affinity is modeled by $K_{ia,jb} = \exp\left((\mathbf{f}_{ij} - \mathbf{f}_{ab})^2/\sigma_2^2\right)$ where $\mathbf{f}_{ij}$ is edge length E_{ij}. We empirically set $\sigma_2 = 5 \times 10^{-7}$ for all experiments. The third-order affinity model follows Eq. (20) with $\sigma_3 = 0.1$.

We re-implement the parallelization-friendly SM, RRWM and RRWHM solvers with GPU. While the other compared methods involve iterative computing and complicated branching, which are not suitable for GPU. Thus the CPU version released by [14] are compared. NGM-V means vanilla NGM without Sinkhorn embedding (see discussion between Eq. (12) and Eq. (13)) and NGM-MH means multi-head Sinkhorn embedding with NGM, by concatenating additional 8 Sinkhorn channels to $\mathbf{m}^{(k)}$ in Eq. (13). The multi-graph matching smoothing technique [29] is adopted for multi-matching baseline NGM-SF, where learned NGM is followed with spectral fusion. The learning rate starts at 10^{-2} decays by 10 every 5,000 steps. Multi-graph matching involves 4 graphs by default. The Hungarian algorithm is used as the common discretization step.

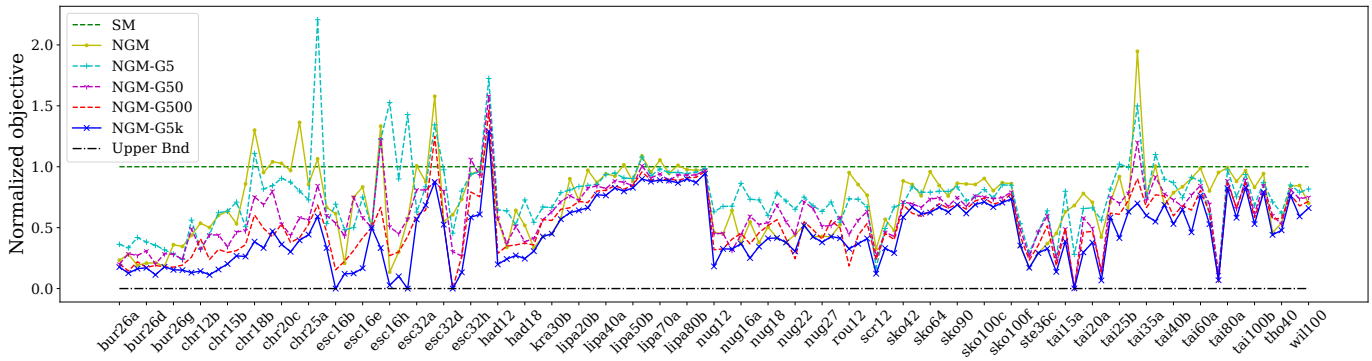
4.1.3 Result and discussions

Fig. 6(a-c) shows our proposed NGM performs comparatively with state-of-the-art solvers in matching accuracy, and can even surpass under severe random scaling Fig. 6(b). Further improvement in accuracy is achieved via multi-head Sinkhorn embedding model NGM-MH. NMGM gains steadily from NGM by fusing multi-matching information, and in Fig. 6(d) we show the improvement in NMGM by introducing more graphs, and the necessity of learning joint matching as NMGM steadily outperforms NGM-SF whose weights are from two-graph NGM. With the third-order affinity, NHGM shows state-of-the-art robustness to noise, scaling and outliers. Compared to learning-free hyper-graph matching RRWHM [24], our NHGM performs comparatively in the precense of outliers as shown in Fig. 6(c). While it performs more robustly to noises and scaling in Fig. 6(a&b). As shown in Fig. 6(h), NGM, NGM-V and NGM-MH are among the fastest graph matching algorithms, and due to efficient GPU parallelization, NHGM is comparatively fast against PSM [60] and more time-efficient compared to state-of-the-art BPF [14]. In contrast, traditional hypergraph matching algorithms e.g. RRWHM [24] are usually much slower than second-order graph matching and unscalable to larger size of problems.

We report the QAP objective score solved by two-graph matching methods in Fig. 6(e-g), where interestingly our learning-based solvers reach relatively low scores compared to their corresponding accuracy. As have been discussed in Sec. 1, the QAP objective may be biased under noisy conditions i.e. the optimal solution to QAP may not correspond to true matching. Our solvers learns to ignore the noisy patterns in input affinity matrix. Such a phenomenon becomes more severe in matching real-world images, and the strength of learning-based solvers becomes more significant, as will be shown in Sec. 4.3.1 in details.



(a) Normalized objective score of our best-performing NGM-G5k against learning-free QAP solvers. Failed instances are plotted at the top of y-axis. NGM-G5K performs comparatively against Sinkhorn-JA [15] on most instances and can even surpass (especially on `bur`, `esc`, `had`, and `tai`).



(b) Normalized objective score comparing different sampling settings. More samples in Gumbel-Sinkhorn guarantee a higher probability of finding better solutions, at the cost of increased computation. NGM always picks the permutation matrix with the highest probability with Sinkhorn and Hungarian algorithms, and performs similarly to NGM-G50 and NGM-G500.

Fig. 7. Normalized objective score on real-world QAPLIB instances (lower score is better). Not all instance labels are shown due to limited space.

The effectiveness of matching-aware Sinkhorn embedding is shown in the accuracy gap between NGM and NGM-V. Further improvement is achieved by multi-head Sinkhorn embedding in NGM-MH, especially with random scaling in Fig. 6(b). As discussed in Sec. 3.6.2, NGM-V without Sinkhorn embedding works in a way similar to SM as the embedding procedure does not consider the assignment constraint, and they also perform closely to each other. On the other hand, by exploiting Sinkhorn embedding, NGM and NGM-MH are conceptually similar to RRWM as all of them try to incorporate the assignment constraint on the fly. Their performances are also very close.

4.2 Learning Real-world QAP Instances

4.2.1 Experiment setting

Our NGM solves the most general Lawler's QAP, which has a wide range of applications beyond vision. Evaluation on QAPLIB [32] is performed to show the capability of NGM on learning the QAP objective score, which should be minimized in QAPLIB (in contrast, objective score is maximized in graph matching). The QAPLIB contains 134 real-world QAP instances from 15 categories, e.g. planning a hospital facility layout [62]. The problem size is defined as $n_1 = n_2 = n$ from Lawler's QAP in Eq. (1), and ranges from 12 to 256. Results are reported on 133 instances with $12 \leq n \leq 150$, as the most challenging `tai256c` is computationally intractable with our testbed (275GB video memory is required for intermediate computing). We set the loss function as the objective score of QAP, keeping the model architecture unchanged. It turns out a self-supervised learning task where the objective is minimized:

$$L_{obj} = \mathrm{vec}(\mathbf{S})^\top \mathbf{K} \mathrm{vec}(\mathbf{S}) \tag{32}$$

where $\mathbf{S}$ is from the output Sinkhorn layer of NGM. Given optimal L_{obj} is reached, the learned $\mathbf{S}$ is a double-stochastically relaxed solution to original QAP. To explore the feasible space, Gumbel-Sinkhorn discussed in Sec. 3.6.3 is adopted during inference.

We train one model for each category, and report the normalized objective score. In consideration of compact and intuitive illustration, the normalized objective score is computed with the upper bound (primal bound) provided by the up-to-date online benchmark[1] and normalized by the baseline solver spectral matching (SM) [7]:

$$norm\_score = \frac{solved\_score - upper\_bnd}{SM\_score - upper\_bnd} \tag{33}$$

Detailed per-instance scores and timing statics are available in Tab. 9&10. Both standard NGM model (NGM) and NGM with different Gumbel sampling numbers (NGM-G`X`, `X` = number of samples) are validated for their performance. The learning rate is initialized at 10^{-4} and decays by 10 every 50,000 steps. Batch size is set to 1 and the regularization of

1. http://anjos.mgi.polymtl.ca/qaplib/inst.html

TABLE 4
Best-performing frequencies on QAPLIB among all instances and all tested solvers. Our NGM-G5k surpasses all competing methods on most categories and best performs on 72 out of 133 instances.

category	bur	chr	els	esc	had	kra	lipa	nug	rou	scr	sko	ste	tai	tho	wil	total
#instances	8	14	1	19	5	3	16	15	3	3	13	3	25	3	2	133
SM [7]	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	1
RRWM [2]	0	0	0	9	0	0	0	0	0	0	0	0	0	0	0	9
Sinkhorn-JA [15]	0	**14**	**1**	1	0	0	**15**	4	**1**	0	1	0	7	1	**1**	46
NGM	0	0	0	1	0	2	0	5	0	0	0	**3**	1	0	0	12
NGM-G5	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	1
NGM-G50	0	0	0	1	0	0	0	1	0	0	0	0	0	0	0	2
NGM-G500	1	0	0	3	0	0	0	1	**1**	0	5	0	1	0	0	12
NGM-G5k	**8**	1	0	**11**	**5**	**3**	1	**10**	**1**	**3**	**7**	2	**17**	**2**	**1**	**72**

TABLE 5
Pearson correlation coefficient r (only for $|r| \geq 0.2$ are shown) between the listed statistics for each problem instance in QAPLIB and the corresponding gap of two methods (see Eq. (34)), which can be seen as the difficulty of the problem (bigger gap more difficult). Higher value denotes increased negative effect of the corresponding statistics to the final solution quality. The correlation between statistics and the difference of two methods is also listed on last row, where higher value denotes NGM-G5k is more sensitive than Sinkhorn-JA and vice versa. Note that relatively small r ($|r| < 0.2$) is usually considered no relation between two statistics.

method	nz	nz/n^4 (=sparsity)	$\mathbf{K}_{std}/\mathbf{K}_{max}$	$d_{min}/\bar{\mathbf{K}}$	$d_{max}/\bar{\mathbf{K}}$	$d_{std}/\bar{\mathbf{K}}$	$d_{max}/\bar{d}$	$d_{std}/\bar{d}$
NGM-G5k	0.130	**0.631**	**0.212**	**-0.222**	**0.230**	**0.286**	**0.230**	**0.280**
Sinkhorn-JA	**0.270**	**0.545**	-0.090	**-0.220**	**0.291**	**0.355**	**0.291**	**0.475**
$gap_{\text{NGM}} - gap_{\text{SJA}}$	-0.184	-0.121	**0.234**	0.071	-0.137	-0.163	-0.137	**-0.288**

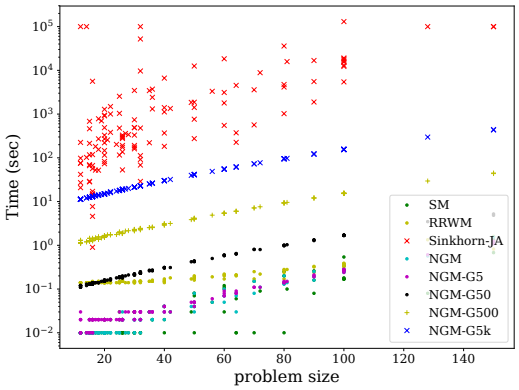


Fig. 8. Inference time (log-scale) of all methods against problem size.

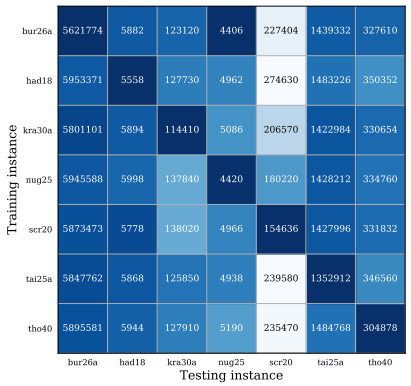


Fig. 9. Generalization test by confusion matrix cross QAP problem instances, where models are learned with instances on y-axis and tested with instances on x-axis. Darker color and lower score correspond to better performance. The tasks are randomly selected from the QAPLIB benchmark. Problem sizes are shown by the numbers in instance names, and our method is insensitive to problem sizes.

Gumbel Sinkhorn $\alpha_g = 1$. Our proposed methods are compared fairly with our GPU implementation of RRWM [2] and SM [7], and results provided in the paper of Sinkhorn-JA [15] (runs on Intel Xeon CPU @ 2.40 GHz). For the problem instances not reported in [15], we assume Sinkhorn-JA fails to reach any feasible solution, as there is no explanation of missing instances in the original paper.

4.2.2 Result and analysis

In Fig. 7(a), our method beats RRWM [2] and SM [7] and is comparative and even superior against state-of-the-art Sinkhorn-JA [15]. As there is no learning-based QAP solver, only non-learning methods are compared. The effectiveness of Gumbel sampling discussed in Sec. 3.6.3 is validated in Fig. 7(b), where Gumbel-based NGM-G5k consistently outperforms deterministic NGM, which always picks the permutation with the highest probability by Hungarian algorithm. The performance of Gumbel-based methods gradually degenerates concerning decreased sampling number, proving more exploration over sampling space guarantees higher expectations on better solutions.

Further evaluation is given in Tab. 4 and Fig. 8. With learning and Gumbel sampling, our NGM-G5k finds the best solution among 72 out of 133 instances, while state-of-the-art learning-free solver Sinkhorn-JA [15] outperforms on 46 instances so that learning-based solvers e.g. NGM can fit a wider range of problems compared to traditional solvers. More importantly, our best-performing model NGM-G5k is of a magnitude faster than Sinkhorn-JA, and adjusting the sampling number of Gumbel method enables balancing between solution quality and computational demand. Finally, we show the generalization ability among different instances in Fig. 9, where NGM-G5k is trained and tested on different randomly picked instances. Our model generalizes soundly to unseen instances with different problem sizes. In conclusion, our learning QAP solvers achieve the best accuracy-speed trade-off on QAPLIB and owns generalizability among different problems.

TABLE 6
Matching accuracy (%) on Pascal VOC Keypoint. Our proposed NGM, NHGM, NGM+ are compared with RRWM [2], GMN [25], PCA-GM [17]. The gray entries denote controlled experiment by replacing NGM with unlearned ImageNet CNN and RRWM solver, among which our joint CNN-solver learning method NGM outperforms. Best results in bold.

method	aero	bike	bird	boat	bottle	bus	car	cat	chair	cow	table	dog	horse	mbike	person	plant	sheep	sofa	train	tv	mean
GMN [25]	31.9	47.2	51.9	40.8	68.7	72.2	53.6	52.8	34.6	48.6	**72.3**	47.7	54.8	51.0	38.6	75.1	49.5	45.0	83.0	86.3	55.3
PCA-GM [17]	40.9	55.0	**65.8**	47.9	76.9	**77.9**	63.5	67.4	33.7	65.5	63.6	61.3	**68.9**	62.8	44.9	77.5	67.4	57.5	**86.7**	**90.9**	63.8
ImgNet-RRWM [2]	16.1	22.3	20.8	21.8	21.3	31.0	23.2	25.4	18.6	20.5	20.6	21.7	18.8	21.9	13.5	28.6	21.7	18.3	50.5	42.8	24.0
ImgNet-NGM solver (ours)	30.8	42.5	44.3	33.8	39.8	52.2	49.2	53.9	27.5	42.4	29.3	49.1	45.1	45.1	24.0	48.3	49.9	29.9	70.2	73.3	44.0
NGM (ours)-RRWM [2]	41.5	54.7	54.3	50.3	67.9	74.3	70.3	60.6	**42.3**	59.1	48.1	57.3	59.1	56.2	40.6	69.6	63.1	52.2	76.3	87.8	59.3
NGM (ours)	50.1	63.5	57.9	53.4	**79.8**	77.1	73.6	68.2	41.1	**66.4**	40.8	60.3	61.9	63.5	45.6	77.1	**69.3**	**65.5**	79.2	88.2	64.1
NHGM (ours)	**52.4**	62.2	58.3	55.7	78.7	77.7	**74.4**	**70.7**	42.0	64.6	53.8	61.0	61.9	60.8	46.8	**79.1**	66.8	55.1	80.9	88.7	64.6
NGM+ (ours)	50.8	**64.5**	59.5	**57.6**	79.4	76.9	**74.4**	69.9	41.5	62.3	68.5	**62.2**	62.4	**64.7**	**47.8**	78.7	66.0	63.3	81.4	89.6	**66.1**

4.2.3 Further discussion

As shown in Tab. 4 and Fig. 7(a), learning-free Sinkhorn-JA [15] and our NGM-G5k performs better on separate categories of QAPLIB, e.g. Sinkhorn-JA performs better on `chr` and `lipa` instances while our method is more powerful on `bur`, `esc`, `nug`, `scr` and `tai`. Some statistical studies are conducted to discover the relation between model behavior and problem patterns, shedding light for future research on both learning-based and learning-free solvers.

For each problem instance, some statistics are summarized from each instance's affinity matrix: problem size n, mean value $\bar{\mathbf{K}}$, minimum value $\mathbf{K}_{min}$, maximum value $\mathbf{K}_{max}$, standard deviation $\mathbf{K}_{std}$, number of zeros nz, mean degree (of association graph) $\bar{d}$, minimum degree d_{min}, maximum degree d_{max} and standard deviation of degree d_{std}. The performance of algorithms is represented by the gap of the solved objective score against upper bound:

$$gap = \frac{solved\_score - upper\_bnd}{solved\_score} \tag{34}$$

It means the percentage of improvement can be made compared to best-known optima (usually solved at extremely high complexity). Pearson correlation coefficients r are computed between each gap and corresponding statistics, additionally some meaningful combinations of the statistics. Items with $|r| \geq 0.2$ are listed in Tab. 5, where positive correlation means a negative effect on solver's performance because a lower gap is better. The correlation between problem statistics and the difference between two methods are also reported.

In the first two columns, the higher sparsity (nz/n^4, proportion of zeros in affinity matrix) makes the problem significantly more challenging for both NGM-G5k and Sinkhorn-JA, so as a larger number of zeros nz. Then the normalized standard deviation $\mathbf{K}_{std}/\mathbf{K}_{max}$ shows some weak negative effect for NGM-G5k, but little correlation to Sinkhorn-JA. In the last five columns both methods are affected by higher degrees in association graph, while Sinkhorn-JA seems more sensitive to the normalized standard deviation of degrees $d_{std}/\bar{\mathbf{K}}, d_{std}/\bar{d}$. In summary, sparsity is the key challenge for both NGM-G5k (where message passing paths are blocked) and Sinkhorn-JA (where it becomes harder to find tight lower bounds). Furthermore, learning with the association graph can restrain the noisy deviation in degrees. Future improvement may be achieved by designing graph learning models with higher capacity, and designing global communication mechanisms against sparse association graphs.

4.3 Real Image for Joint CNN and QAP Learning

Our matching net allows for raw image input, from which a CNN is learned (see Fig. 4). We evaluate semantic keypoint matching on Pascal VOC dataset with Berkeley annotations[2] [63] and Willow ObjectClass dataset [26].

4.3.1 Results on Pascal VOC Keypoint dataset

This natural image dataset [63] consists of 20 instance classes with semantic keypoint labels. We follow the problem setting in [17], where image pairs with inlier positions are fed into the model. We consider it a challenging dataset because instance may vary from its scale, pose and illumination, and the number of keypoints in each image varies from 6 to 23. The shallow learning method HARG-SSVM [26] incorporates a fix-sized reference graph, therefore it is inapplicable to our experiment setting where instances from the same category have different inliers. As our multi-matching mechanism in Sec. 3.5 requires the same number of nodes among graphs, our multi-graph model NMGM is not compared either.

In line with the protocol of the peer methods [17], [25], we filter out poorly annotated images. Then we get 7,020 training samples and 1,682 testing samples. Instances are cropped around their ground truth bounding boxes and resized to 256×256 before fed into the network. As discussed in Sec. 3.2.1, we adopt VGG16 backbone [47] and construct the affinity matrix from the same CNN layers: `relu4_2` for node features and `relu5_1` for edge features. The learning rate starts at 10^{-2} and decays by 10 every 10,000 steps. For two input images, one graph is constructed by Delaunay triangulation and the other is fully-connected. $\sigma_3 = 10^{-4}$ is set for third-order affinity of NHGM.

We compare **GMN** [25], **PCA-GM** [17], by which affinity functions are learned for graph matching. Results in Tab. 6 show that with CNN feature and QAP solver learned jointly, our methods surpass competing methods on most categories, especially best performs in terms of mean accuracy. Specifically, NGM surpasses the state-of-the-art deep graph matching method PCA-GM, and it is worth noting that PCA-GM incorporates explicit modeling on higher-order and cross-graph affinities, while only second-order affinity is considered in our QAP formulation (and third-order for hypergraph matching). NHGM further improves by hypergraph affinities and NGM+ best performs among all methods by exploiting edge embeddings.

2. https://www2.eecs.berkeley.edu/Research/Projects/CS/vision/shape/poselets/voc2011_keypoints_Feb2012.tgz

Fig. 10. Visualization of matching results on 20 Pascal VOC Keypoint categories. Green and red represent correct and incorrect predictions, respectively. As many instances vary a lot in their pose and appearance, this dataset is considered challenging even for deep graph matching.

On the gray methods in Tab. 6, the first entry represents the CNN weights comes from whether pretrained ImageNet classifier (ImgNet) [64] or learned graph matching model (NGM); the second entry means either learning-free solver RRWM or learning-based solver NGM is adopted for QAP. The necessity of learning on both CNN and QAP solver is validated, and our joint CNN and QAP learning NGM best performs among them. Learning with CNN unsurprisingly mitigates the gap between classification task and matching task, while it is worth noting the learned QAP solver is nearly twice accurate against RRWM with ImageNet CNN (44.0% vs 24.0%), showing the robustness against noisy affinity matrices in real-world image matching. The GPU memory cost and running speed during training are listed: NGM (4761MB, 13.4 pairs/s); NGM+ (5441MB, 13.3 pairs/s); and NHGM (9251MB, 10.1 pairs/s), which is comparable to PCA-GM [17] (5165MB, 14.4 pairs/s).

Some visualization of real-world matching result among competing methods can be found in Fig. 10. The error patterns of NGM, NHGM and NGM+ are similar but differ from PCA-GM, and our NHGM and NGM+ improves NGM by correcting some existing errors, e.g. in "aeroplane", "bike" and "motorbike". In general, our proposed methods are more powerful on rigid categories, e.g. aeroplane, bike, car, motorbike, pottedplant and sofa. The node-wise and edge-wise affinity formulation is helpful in modeling structural information, as shown in visualization of "car". In contrast, PCA-GM seems more powerful on non-rigid objects such as birds and horses. Our methods, however, fail when the objects vary a lot in their pose and appearance (see "bird" and "bus"). Additionally, as there exists many non-rigid and pose-varying objects, the effect of adding geometric-based higher-order information in NHGM seems not significant. In contrast, NGM+ provides more convincing result, which is probably due to the increased model capacity with edge embedding.

The generalization ability is further validated by confusion matrices for NGM and NGM+ as shown in Fig. 11. Models are trained only with categories on the x-axis and tested with all categories. The color map is determined by the accuracy in current cell normalized by the highest accuracy in its column. As shown in the confusion matrices, both NGM and NGM+ own some generalization ability between visually similar categories, e.g. chair and sofa, cat and dog. Compared to peer deep graph matching methods [17], [25], NGM fits better on the training categories on the diagonal of confusion matrices. In contrast, higher-capacity NGM+ seems less powerful than NGM when trained with single category, which may be caused by overfitting as the size of training set shrinks by around 20 times.

4.3.2 *Results on Willow ObjectClass*

This natural image dataset covers 5 categories. Each category contains at least 40 images, and all instances in

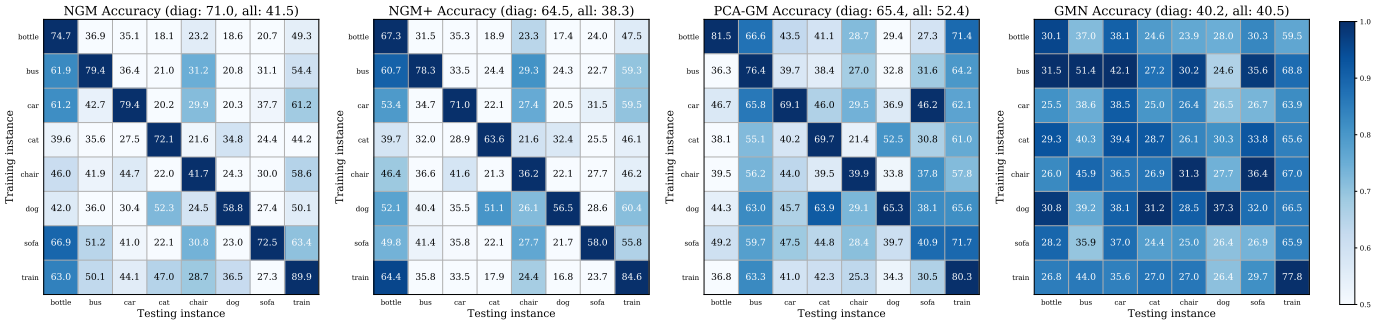


Fig. 11. Generalization study shown as confusion matrix of our proposed NGM and NGM+ in line with [17], [25] on Pascal VOC Keypoint. Models are trained on categories on the y-axis and tested on categories on the x-axis. Darker color denotes relatively better performance in the same column, and the averaged accuracy on both diagonal part (learned during training) and all elements (trained+generalized) of confusion matrices are reported in the brackets over confusion matrices. The train/test split follows the main experiment and eight categories are selected randomly.

TABLE 7
Matching accuracy (%) on Willow ObjectClass dataset.

method	# graphs	face	m-bike	car	duck	w-bottle
HARG-SSVM [26]	2	91.2	44.4	58.4	55.2	66.6
GMN [25]	2	99.3	71.4	74.3	82.8	76.7
PCA-GM [17]	2	**100.0**	76.7	84.0	**93.5**	96.9
NGM (ours)	2	99.4	76.8	84.2	77.6	88.3
NHGM (ours)	2	99.9	79.3	86.5	72.2	89.4
NGM+ (ours)	2	97.7	53.5	70.4	54.3	87.3
NGM-SF (ours + [29])	3	99.4	84.6	91.3	83.1	94.6
NMGM (ours)	3	**100.0**	84.8	97.2	88.1	94.9
NGM-SF (ours + [29])	6	99.5	87.3	90.4	82.3	94.3
NMGM (ours)	6	**100.0**	86.0	**97.3**	85.4	**97.7**
NGM-SF (ours + [29])	16	97.2	**90.6**	93.1	83.9	95.5
NMGM (ours)	16	**100.0**	87.7	95.3	86.1	96.0

TABLE 8
Ablation study of NGM+ on Pascal VOC Keypoint.

model	baseline NGM	+ node affinity	+ Sinkhorn embedding
accuracy (rel.)	58.7%	59.6% (+0.9)	64.1% (+4.5)
edge embedding	EGNN(C) [52]	HyperConv [65]	Ours Eq. (18)
accuracy	53.3%	55.9%	**66.1%**

the same class share 10 distinctive image keypoints. We mainly evaluate multi-graph matching learning of NMGM on Willow ObjectClass. Following the protocol in [17], we directly train our methods on the first 20 images and report testing results on the rest. The learning rate starts at 10^{-2} and decays by 10 every 500 steps.

The performance of HARG-SSVM [26], GMN [25] and PCA-GM [17] reported in [17] are listed and compared. Learning-free spectral fusion multi-matching algorithm [29] is also integrated and tested as NGM-SF, by post-processing on NGM pairwise matching. Tab. 7 shows NGM and NHGM performs comparatively to PCA-GM especially on rigid objects, and NMGM surpasses PCA-GM by learning multi-graph information. Learning joint matching on more graphs can achieve further improvement, as NMGM with 6 graphs outperforms 3 graphs. However, the multi-matching capacity seems saturated as there is little improvement by involving 16 graphs. With no learning on multi-graph matching, little improvement is observed for NGM-SF from 3 graphs to 6 graphs and 16 graphs, except the accuracy on motorbikes. The WILLOW dataset is relatively small for deep graph matching as there are only $5 \times 20 = 100$ training images. With higher model capacity, NGM+ suffers overfitting with training loss $= 0$ but it performs poorly on test data.

4.3.3 Ablation study on Pascal VOC Keypoint

Ablation study is performed on our proposed modules in NGM+ and experimental results on Pascal VOC Keypoint dataset are shown in Tab. 8. The baseline model is built following NGM introduced in Sec. 3.2, however node affinity is ignored in model input, i.e. $\mathbf{v}_{ia}^{(0)} = 1$ and Sinkhorn embedding (Sec. 3.6.2) is excluded. The effectiveness of node affinity, Sinkhorn embedding, and NGM+'s edge embedding are evaluated by adding these components successively to the model. Based on the NGM model with node affinity and Sinkhorn embedding (64.1%), we also test other novel edge-embedding schemes including EGNN(C) [52] and HyperConv [65], and it is discovered not all edge-embedding models are suitable for learning with the complicated association graph.

5 Conclusion

We have presented a novel neural graph matching network. There are three main highlights: i) The first graph matching network directly learning Lawlers QAP which is general with a wide range of applications e.g. on QAPLIB beyond visual matching. This is in contrast to many existing works that can only take separate graphs as input. ii) The first deep network for hypergraph matching which involves third-order edges. iii) The first network for deep learning of multiple graph matching. Extensive experimental results on synthetic and real-world data show the state-of-the-art performance of our approach. In particular, it shows the notable cost-efficiency advantages against learning-free methods. The source code will be made publicly available. The future work will explore the more scalable approach for handling large-scale QAP problem. For graph matching, it indicates larger graph and more graphs for matching.

Acknowledgments

The work is partially supported by National Key Research and Development Program of China (2018AAA0100704, 2016YFB1001003), NSFC (61972250, U19B2035), STCSM (18DZ1112300), and the Open Project Program of the National Laboratory of Pattern Recognition (NLPR).

References

[1] M. R. Garey and D. S. Johnson, *Computers and Intractability; A Guide to the Theory of NP-Completeness*, 1990.
[2] M. Cho, J. Lee, and K. M. Lee, "Reweighted random walks for graph matching," in *ECCV*, 2010.
[3] S. Gold and A. Rangarajan, "A graduated assignment algorithm for graph matching," *TPAMI*, 1996.
[4] M. A. Fischler and R. C. Bolles, "Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography," *Commun. ACM*, 1981.

TABLE 9
Per-instance results by objective score (the smaller the better) of solved score and inference time on QAPLIB.

instance	score									time (sec)							
	Upper	SM [7]	RRWM [2]	SK-JA [15]	NGM	NGM-G5	NGM-G50	NGM-G500	NGM-G5k	SM	RRWM	SK-JA	NGM	NGM -G5	NGM -G50	NGM -G500	NGM -G5k
bur26a	5426670	6533340	6663181	5688893	5684628	5828287	5650343	5650343	**5621774**	0.02	0.15	309.9	0.02	0.02	0.19	1.85	19.34
bur26b	3817852	4690772	4741283	4053243	4063246	4110185	4066986	3943769	**3927943**	0.01	0.15	191.7	0.02	0.02	0.19	1.88	19.21
bur26c	5426795	6537412	6474996	5639665	5638641	5892078	5728542	5671528	**5608065**	0.01	0.15	136.9	0.02	0.02	0.19	1.90	18.74
bur26d	3821225	4649645	4678974	3985052	3994147	4137709	4077089	3965083	**3962317**	0.02	0.16	276.6	0.02	0.02	0.18	1.89	18.78
bur26e	5386879	6711029	6619788	5539241	5666202	5856035	5655088	5638619	**5536142**	0.01	0.16	52.9	0.03	0.02	0.19	1.87	18.62
bur26f	3782044	4723824	4814298	3979071	3954977	4082092	4050459	**3949064**	3949711	0.01	0.15	173.6	0.02	0.02	0.20	1.86	18.59
bur26g	10117172	12168111	12336830	10624776	10855165	10694274	10694272	10477104	**10433439**	0.01	0.15	292.8	0.02	0.02	0.19	1.87	18.55
bur26h	7098658	8753694	8772077	7453329	7670546	7493907	7493907	7417478	**7348866**	0.01	0.15	330.4	0.02	0.02	0.18	1.87	18.64
chr12a	9552	50732	43624	**9552**	27556	32670	29780	20138	14940	0.01	0.14	75.7	0.01	0.01	0.12	1.15	11.26
chr12b	9742	46386	73860	**9742**	29396	21528	21528	24734	14984	0.01	0.14	75.1	0.01	0.01	0.12	1.16	11.34
chr12c	11156	57404	50130	**11156**	34344	31602	31560	22174	16346	0.01	0.14	97.9	0.01	0.02	0.12	1.12	11.20
chr15a	9896	77094	90870	**11616**	50272	51746	39414	31478	20442	0.01	0.14	683.6	0.01	0.02	0.13	1.26	12.60
chr15b	7990	77430	115556	**7990**	52082	52066	31936	28546	22048	0.01	0.14	461.9	0.01	0.01	0.13	1.26	12.61
chr15c	9504	64198	70738	**9504**	38568	48318	34932	26548	24190	0.01	0.14	214.1	0.01	0.02	0.13	1.25	12.59
chr18a	11098	94806	115328	**11948**	83026	53612	50814	41254	33124	0.01	0.14	781.5	0.01	0.02	0.14	1.37	14.00
chr18b	1534	4054	3852	2690	4810	4330	3430	3056	**2504**	0.01	0.14	52.1	0.01	0.02	0.14	1.48	13.97
chr20a	2192	11154	13970	**4624**	10728	9516	8382	6626	5178	0.02	0.14	1285.8	0.01	0.02	0.16	1.51	15.02
chr20b	2298	9664	14168	**3400**	9962	8522	8196	5426	5766	0.02	0.14	911.3	0.01	0.02	0.16	1.66	14.96
chr20c	14142	112406	195572	**40464**	115128	103040	66182	66382	49770	0.02	0.14	945.0	0.01	0.02	0.15	1.68	14.93
chr22a	6156	16732	15892	**9258**	16410	15394	10778	10160	9348	0.02	0.14	1488.4	0.01	0.02	0.17	1.75	16.88
chr22b	6194	13294	13658	**6634**	15876	11882	10330	9156	9006	0.02	0.14	1005.3	0.01	0.02	0.17	1.76	16.10
chr25a	3796	21526	32060	**5152**	18950	16704	13758	13162	11648	0.03	0.15	2553.2	0.01	0.02	0.19	2.04	17.93
els19	17212548	33807116	74662642	**18041490**	34880280	53830864	31247564	28600336	27029748	0.01	0.14	700.0	0.01	0.02	0.15	1.47	14.49
esc16a	68	98	80	100	88	84	86	82	**78**	0.00	0.14	12.8	0.01	0.02	0.14	1.27	13.00
esc16b	292	318	294	304	308	310	306	296	**292**	0.00	0.14	4.6	0.01	0.02	0.13	1.29	12.97
esc16c	160	276	204	266	184	216	210	186	**174**	0.01	0.14	7.7	0.01	0.01	0.13	1.30	13.00
esc16d	16	48	44	58	40	32	40	26	**20**	0.01	0.14	14.6	0.01	0.02	0.13	1.30	12.95
esc16e	28	52	50	44	48	46	42	38	**32**	0.00	0.14	13.5	0.01	0.02	0.14	1.31	13.00
esc16f	0	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	0.00	0.14	0.9	0.01	0.02	0.14	1.30	12.98
esc16g	26	44	52	52	50	48	48	38	**32**	0.00	0.14	17.1	0.01	0.02	0.14	1.30	13.04
esc16h	996	1292	**1002**	1282	1036	1448	1146	1076	1004	0.00	0.14	15.1	0.01	0.02	0.13	1.29	12.95
esc16i	14	54	28	36	26	50	32	26	**18**	0.02	0.14	5625.6	0.01	0.02	0.13	1.31	12.97
esc16j	8	22	18	18	16	28	16	14	**8**	0.01	0.14	13.0	0.01	0.02	0.14	1.30	13.04
esc32a	130	426	**240**	456	428	316	370	308	298	0.01	0.15	91.8	0.02	0.03	0.23	2.25	22.91
esc32b	168	460	400	416	424	412	404	**356**	368	0.00	0.15	28.9	0.01	0.03	0.24	2.36	22.87
esc32c	642	770	**650**	886	844	814	754	802	754	0.00	0.15	112.4	0.02	0.02	0.25	2.38	22.95
esc32d	200	360	**224**	356	288	356	328	284	284	0.00	0.14	68.4	0.02	0.02	0.23	2.41	22.84
esc32e	2	68	6	46	42	32	22	**2**	**2**	0.02	0.15	9661.4	0.02	0.03	0.23	2.42	22.88
esc32g	6	36	**10**	46	28	30	14	14	**10**	0.01	0.15	52135.2	0.02	0.03	0.24	2.35	22.82
esc32h	438	602	**506**	-	592	592	612	568	534	0.00	0.15	-	0.02	0.02	0.23	2.45	22.79
esc64a	116	254	**124**	276	250	248	244	220	200	0.01	0.20	225.8	0.03	0.08	0.65	6.00	61.55
esc128	64	202	**78**	-	238	302	282	266	242	0.08	1.34	-	0.57	0.61	3.48	29.53	297.84
had12	1652	1894	2090	-	1790	1808	1792	1722	**1700**	0.01	0.14	-	0.01	0.02	0.11	1.13	11.28
had14	2724	3310	3494	2916	2922	3098	2940	2928	**2866**	0.01	0.14	102.2	0.01	0.01	0.12	1.20	12.13
had16	3720	4390	4646	3978	4150	4070	4064	3960	**3902**	0.01	0.14	56.7	0.01	0.02	0.13	1.27	12.88
had18	5358	6172	6540	5736	5780	5950	5670	5662	**5558**	0.01	0.15	271.4	0.01	0.02	0.14	1.37	13.90
had20	6922	8154	8550	7464	7334	7592	7430	7362	**7300**	0.01	0.14	328.4	0.01	0.02	0.15	1.51	14.97
kra30a	88900	148690	136830	125290	**114410**	128920	122660	122780	**114410**	0.01	0.14	491.6	0.01	0.03	0.21	2.22	21.36
kra30b	91420	150760	141550	126980	**118130**	130940	128880	124590	**118130**	0.01	0.14	489.9	0.02	0.02	0.21	2.28	21.35
kra32	88700	145310	148730	128120	121340	133290	129080	125530	**120930**	0.01	0.15	479.6	0.02	0.03	0.23	2.42	22.96
lipa20a	3683	3956	3940	**3683**	3929	3904	3891	3864	3853	0.01	0.14	271.1	0.01	0.02	0.15	1.64	14.89
lipa20b	27076	36502	38236	**27076**	33907	34970	33902	33815	33125	0.01	0.14	73.3	0.01	0.02	0.15	1.65	15.04
lipa30a	13178	13861	13786	**13178**	13841	13756	13742	13660	13631	0.01	0.15	191.9	0.02	0.02	0.22	2.24	21.18
lipa30b	151426	198434	201775	**151426**	192356	191633	191034	189057	187607	0.03	0.15	160.5	0.02	0.03	0.22	2.28	21.35
lipa40a	31538	32736	32686	**31538**	32666	32658	32521	32504	32454	0.01	0.14	183.2	0.02	0.03	0.31	3.11	30.09
lipa40b	476581	628272	647295	**476581**	616656	620456	610699	604766	601848	0.04	0.17	369.3	0.04	0.04	0.31	3.11	30.10
lipa50a	62093	64070	64162	**62642**	64100	63886	63816	63705	63671	0.01	0.16	275.2	0.03	0.05	0.42	4.20	41.04
lipa50b	1210244	1589128	1591109	**1210244**	1543264	1552533	1530231	1531102	1523856	0.08	0.20	763.5	0.08	0.05	0.44	4.02	41.25
lipa60a	107218	109861	110468	**108456**	110094	110068	109884	109748	109595	0.03	0.17	551.5	0.05	0.07	0.58	5.46	54.70
lipa60b	2520135	3303961	3300291	**2520135**	3269504	3246252	3231895	3219131	3208501	0.10	0.22	1796.2	0.10	0.09	0.58	5.39	55.37
lipa70a	169755	173649	173569	**172504**	173862	173577	173419	173218	173220	0.01	0.17	565.8	0.08	0.11	0.81	7.19	72.61
lipa70b	4603200	6055613	6063182	**4603200**	5978316	5986328	5889358	5908729	5890161	0.15	0.25	3592.8	0.15	0.11	0.80	7.11	72.90
lipa80a	253195	258345	258608	**257395**	258402	258108	257997	257767	257663	0.01	0.17	1023.4	0.13	0.14	1.01	9.38	94.93
lipa80b	7763962	10231797	10223697	**7763962**	10173155	10094877	10059724	10007233	9983040	0.20	0.27	4158.0	0.20	0.14	0.99	9.29	95.20
lipa90a	360630	367384	367370	366649	367193	367085	366926	366795	**366508**	0.08	0.24	1889.5	0.16	0.20	1.31	12.06	122.14
lipa90b	12490441	16291267	16514577	**12490441**	16194745	16232027	16176035	16143880	16076956	0.26	0.33	5544.5	0.25	0.21	1.34	12.01	122.29
nug12	578	886	1038	682	720	772	716	676	**634**	0.01	0.14	11.4	0.01	0.02	0.11	1.12	11.29
nug14	1014	1450	1720	-	1210	1308	1212	**1156**	**1156**	0.01	0.14	-	0.01	0.01	0.13	1.20	12.10
nug15	1150	1668	2004	1448	1482	1500	**1312**	1360	1318	0.01	0.14	69.6	0.01	0.02	0.13	1.24	12.56
nug16a	1610	2224	2626	1940	**1836**	2140	1882	1888	**1836**	0.01	0.14	118.7	0.01	0.01	0.13	1.31	13.01
nug16b	1240	1862	2192	1492	1580	1696	1610	1468	**1396**	0.01	0.15	66.8	0.01	0.01	0.14	1.45	12.96
nug17	1732	2452	2934	2010	2004	2256	2110	2062	**1980**	0.01	0.14	181.6	0.01	0.02	0.14	1.49	13.44
nug18	1930	2688	3188	**2192**	2312	2384	2340	2322	2242	0.01	0.15	155.2	0.01	0.02	0.14	1.58	13.88
nug20	2570	3450	4174	3254	**2936**	3260	3174	3068	**2936**	0.01	0.15	146.7	0.01	0.02	0.16	1.68	14.87
nug21	2438	3702	4228	3064	**2916**	3346	3132	2954	**2916**	0.01	0.14	256.8	0.01	0.02	0.16	1.74	15.52
nug22	3596	5896	6382	**3988**	4616	5092	4616	4160	4298	0.01	0.14	382.6	0.01	0.02	0.16	1.73	16.09
nug24	3488	4928	5720	4424	**4234**	4568	4514	4278	**4234**	0.01	0.14	202.6	0.01	0.02	0.18	1.76	17.34
nug25	3744	5332	5712	**4302**	4420	4822	4788	4510	4420	0.01	0.14	478.7	0.01	0.03	0.18	1.97	17.95
nug27	5234	7802	8626	6244	6332	6860	6546	6276	**6208**	0.01	0.14	360.3	0.01	0.03	0.20	2.12	19.34
nug28	5166	7418	8324	6298	**6128**	6764	6332	6424	**6128**	0.01	0.14	339.6	0.01	0.03	0.21	2.16	19.96
nug30	6124	8956	10034	**7242**	7608	7666	7780	7530	7294	0.01	0.14	330.7	0.02	0.03	0.22	2.32	21.28
rou12	235528	325404	377168	276446	321082	301728	275876	**252156**	264898	0.01	0.14	41.9	0.01	0.01	0.12	1.31	11.35
rou15	354210	489350	546526	**390810**	469592	453260	430000	413294	403872	0.01	0.14	66.2	0.01	0.01	0.13	1.44	12.49
rou20	725522	950018	1010554	823298	897348	876282	867682	845494	**817776**	0.01	0.14	115.1	0.01	0.02	0.15	1.66	14.97
scr12	31410	71392	95134	45334	44400	38228	42014	40908	**36292**	0.01	0.14	20.8	0.01	0.01	0.12	1.29	11.33
scr15	51140	104308	101714	74632	81344	77376	75746	75224	**68768**	0.02	0.14	117.1	0.01	0.02	0.13	1.45	12.62
scr20	110030	263058	350528	171260	182882	212432	175534	171666	**154636**	0.01	0.14	220.8	0.01	0.02	0.16	1.68	14.96

TABLE 10
Continued per-instance results by objective score (the smaller the better) of solved score and inference time on QAPLIB.

instance	score									time (sec)							
	Upper	SM [7]	RRWM [2]	SK-JA [15]	NGM	NGM-G5	NGM-G50	NGM-G500	NGM-G5k	SM	RRWM	SK-JA	NGM	NGM -G5	NGM -G50	NGM -G500	NGM -G5k
sko42	15812	20770	23612	19058	20192	19274	19340	19218	**18716**	0.03	0.18	1342.5	0.03	0.03	0.31	3.16	32.01
sko49	23386	29616	34548	**27160**	28712	28584	27718	27238	27554	0.03	0.17	1849.2	0.04	0.05	0.42	3.86	39.80
sko56	34458	44594	49650	40954	42182	42476	41218	**40474**	40684	0.05	0.19	3318.1	0.05	0.06	0.50	4.89	48.94
sko64	48498	60878	65540	**55738**	60368	58274	57594	56392	56222	0.06	0.21	4533.6	0.07	0.09	0.63	6.01	61.82
sko72	66256	82156	89264	**76332**	79716	78956	78092	77320	76870	0.09	0.22	8845.2	0.11	0.11	0.80	7.61	77.14
sko81	90998	112838	118372	105246	107588	108404	105776	105418	**104710**	0.10	0.28	15863.8	0.14	0.15	1.03	9.58	97.28
sko90	115534	140840	148784	133818	137402	136624	134472	133056	**132942**	0.16	0.32	16796.6	0.19	0.19	1.28	12.15	122.47
sko100a	152002	185738	184854	176626	180972	176336	174818	173980	**172810**	0.18	0.33	18370.8	0.26	0.27	1.70	15.80	155.41
sko100b	153890	185366	189502	177398	180774	177600	177832	176538	**175588**	0.17	0.33	15432.1	0.26	0.24	1.70	15.59	155.32
sko100c	147862	178710	188756	**169566**	175740	172940	171062	170784	169806	0.17	0.38	13000.4	0.27	0.27	1.68	15.29	155.69
sko100d	149576	181328	186086	**170648**	175096	173170	172968	171586	170816	0.17	0.33	17350.9	0.26	0.26	1.72	15.27	155.58
sko100e	149150	180062	192342	171656	176010	175448	172216	171232	**170958**	0.17	0.37	16240.4	0.26	0.28	1.69	15.27	155.24
sko100f	149036	177518	189284	171296	173552	173194	171608	171232	**169986**	0.17	0.37	19155.6	0.25	0.27	1.69	15.33	155.17
ste36a	9526	30030	33294	17938	**16648**	20452	19506	18320	16768	0.02	0.15	2415.2	0.02	0.03	0.26	2.54	26.21
ste36b	15852	176526	193046	47616	**43248**	63332	58752	53426	**43248**	0.02	0.16	3718.0	0.02	0.03	0.27	2.59	26.32
ste36c	8239110	24530792	28908062	14212212	**12988352**	15819606	15824300	14055568	**12988352**	0.02	0.15	1312.1	0.02	0.03	0.27	2.61	26.42
tai12a	224416	318032	392004	**245012**	259014	284106	280362	272610	255158	0.01	0.14	27.1	0.01	0.01	0.11	1.13	11.38
tai12b	39464925	96190153	124497790	81727424	65138752	53817560	53707944	50142456	**47252044**	0.01	0.14	225.1	0.01	0.03	0.12	1.15	11.35
tai15a	388214	514304	571952	471272	467812	488870	450056	450586	**436968**	0.01	0.14	28.2	0.01	0.02	0.13	1.26	12.51
tai15b	51765268	702925159	702292926	**52585356**	495479040	234006816	53608232	52900096	52871608	0.01	0.14	29.0	0.02	0.01	0.13	1.24	12.56
tai17a	491812	669712	738566	598716	630644	608616	594994	574052	**544754**	0.01	0.14	52.4	0.01	0.02	0.14	1.42	13.94
tai20a	703482	976236	1012228	849082	896518	884076	838930	831358	**806382**	0.01	0.14	82.6	0.01	0.02	0.15	1.63	14.91
tai20b	122455319	394836310	602903767	220470588	237607744	275857568	160806544	159867648	**140704160**	0.02	0.15	489.9	0.01	0.02	0.15	1.62	14.89
tai25a	1167256	1485502	1536172	**1341104**	1393248	1427052	1405630	1364288	1352912	0.02	0.14	116.0	0.02	0.02	0.19	1.92	18.03
tai25b	344355646	764920942	1253946482	798113083	730775168	773628544	638959616	600683456	**518647040**	0.02	0.14	1040.0	0.01	0.02	0.19	1.93	17.95
tai30a	1818146	2210304	2305048	2072218	**2065706**	2208012	2125220	2101282	**2065706**	0.03	0.15	175.3	0.01	0.03	0.21	2.27	21.35
tai30b	637117113	1008164383	1766978330	1114514832	1359600384	1193240320	1082828032	972068480	**896379008**	0.03	0.15	3464.2	0.02	0.02	0.22	2.23	21.32
tai35a	2422002	3030184	3100748	2820060	2886132	2874582	2861898	2826342	**2786748**	0.03	0.15	221.1	0.03	0.03	0.27	2.55	25.35
tai35b	283315445	454981851	574511546	446783959	455718176	472176896	440893216	415469760	**377687744**	0.03	0.15	3440.6	0.03	0.03	0.26	2.53	25.36
tai40a	3139370	3825396	3985684	**3547918**	3610604	3755024	3672066	3660256	3610604	0.04	0.16	1121.6	0.04	0.04	0.31	2.90	30.20
tai40b	637250948	1165811212	1423772477	1019672934	1053339520	1096534400	1022449792	952554880	**917498816**	0.04	0.15	6646.7	0.04	0.03	0.31	2.91	29.92
tai50a	4938796	6078426	6203546	**5569952**	5891066	5788660	5704692	5714682	5677282	0.07	0.19	1418.5	0.04	0.05	0.43	4.08	41.38
tai50b	458821517	796553600	790688128	696556852	764856128	767252544	718563200	676292800	**614638528**	0.08	0.18	12552.0	0.04	0.05	0.43	3.98	41.11
tai60a	7205962	8614998	8731620	**8243624**	8596094	8449862	8397220	8341650	8281996	0.11	0.21	3121.1	0.07	0.08	0.60	5.40	55.35
tai60b	608215054	1089964672	1279537664	978843717	994559424	902721984	942202560	909268288	**862969152**	0.12	0.20	18385.7	0.07	0.07	0.60	5.40	55.34
tai64c	1855928	5893540	6363888	3189566	5703540	2348902	2324478	**2072176**	2133738	0.01	0.21	373.4	0.08	0.08	0.64	6.00	61.70
tai80a	13499184	15665790	16069786	15352662	15648708	15608216	15417076	15383582	**15283138**	0.20	0.28	4745.2	0.14	0.14	0.97	9.31	95.09
tai80b	818415043	1338090880	1410723456	1215586531	1275809408	1209990912	1159786624	1163362432	**1120577408**	0.25	0.24	35995.4	0.13	0.15	1.00	9.24	94.92
tai100a	21052466	24176962	24446982	23787764	24077728	23977908	23840360	23759392	**23644528**	0.34	0.39	5447.5	0.27	0.28	1.67	15.38	156.06
tai100b	1185996137	1990209280	2192130048	**1589275900**	1853681152	1736472576	1680677888	1654308224	1612020992	0.54	0.34	130312.5	0.24	0.27	1.69	15.22	156.20
tai150b	498896643	662657408	755505920	-	653429440	641556480	637699712	629012864	**628349568**	1.35	1.60	-	1.55	1.25	5.20	44.96	433.41
tho30	149936	230828	267194	202844	187062	207424	198456	196072	**185622**	0.01	0.14	739.1	0.02	0.03	0.21	2.13	21.38
tho40	240516	375154	440146	314070	313026	323808	311780	318188	**304878**	0.02	0.15	1407.0	0.03	0.04	0.30	2.97	30.18
tho150	8133398	10000616	10689758	**9508422**	9702946	9720102	9649596	9569826	9557766	0.68	0.82	99778.2	1.12	1.09	4.86	43.62	443.22
wil50	48816	56588	60420	54030	55390	54962	54552	54086	**53418**	0.04	0.18	1867.0	0.05	0.04	0.43	4.02	42.11
wil100	273038	305030	307258	**292118**	295418	299162	297012	295952	294172	0.17	0.34	12315.5	0.28	0.28	1.68	15.43	154.60

[5] Z. Zhang, "Iterative point matching for registration of free-form curves and surfaces," *IJCV*, 1994.

[6] E. M. Loiola, N. M. de Abreu, P. O. Boaventura-Netto, P. Hahn, and T. Querido, "A survey for the quadratic assignment problem," *EJOR*, 2007.

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

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

[9] C. Edwards, "A branch and bound algorithm for the koopmans-beckmann quadratic assignment problem," in *Combinatorial optimization II*. Springer, 1980, pp. 35–52.

[10] A. P. Punnen and S. N. Kabadi, "A linear time algorithm for the koopmans–beckmann qap linearization and related problems," *Discrete Optimization*, vol. 10, no. 3, pp. 200–209, 2013.

[11] G. Erdoğan and B. Tansel, "A branch-and-cut algorithm for quadratic assignment problems based on linearizations," *Computers & Operations Research*, vol. 34, no. 4, pp. 1085–1106, 2007.

[12] T. Dokeroglu and A. Cosar, "A novel multistart hyper-heuristic algorithm on the grid for the quadratic assignment problem," *Engineering Applications of Artificial Intelligence*, vol. 52, pp. 10–25, 2016.

[13] P. Hahn, T. Grant, and N. Hall, "A branch-and-bound algorithm for the quadratic assignment problem based on the hungarian method," *European Journal of Operational Research*, vol. 108, no. 3, pp. 629 – 640, 1998.

[14] T. Wang, H. Ling, C. Lang, and S. Feng, "Graph matching with adaptive and branching path following," *TPAMI*, 2017.

[15] Y. Kushinsky, H. Maron, N. Dym, and Y. Lipman, "Sinkhorn algorithm for lifted assignment problems," *SIAM Journal on Imaging Sciences*, vol. 12, no. 2, pp. 716–735, 2019.

[16] A. Nowak, S. Villar, A. Bandeira, and J. Bruna, "Revised note on learning quadratic assignment with graph neural networks," in *DSW*, 2018.

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

[18] E. L. Lawler, "The quadratic assignment problem," *Management Science*, 1963.

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

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

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

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

[23] R. Zass and A. Shashua, "Probabilistic graph and hypergraph matching," in *CVPR*, 2008.

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

[25] A. Zanfir and C. Sminchisescu, "Deep learning of graph matching," in *CVPR*, 2018.

[26] M. Cho, K. Alahari, and J. Ponce, "Learning graphs to match," in *ICCV*, 2013.

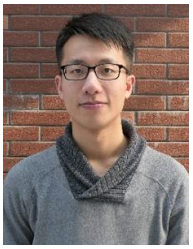
[27] J. Yan, J. Wang, H. Zha, X. Yang, and S. Chu, "Consistency-driven alternating optimization for multigraph matching: A unified approach," *TIP*, vol. 24, no. 3, pp. 994–1009, 2015.

[28] Y. Chen, L. Guibas, and Q. Huang, "Near-optimal joint object matching via convex relaxation," in *ICML*, 2014.

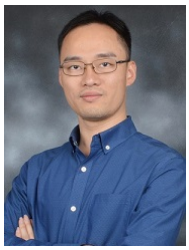
[29] D. Pachauri, R. Kondor, and S. Vikas, "Solving the multi-way matching problem by permutation synchronization," in *NIPS*, 2013.

[30] Q. Wang, X. Zhou, and K. Daniilidis, "Multi-image semantic matching by mining consistent features," in *CVPR*, 2018.

[31] Z. Zhang and W. Lee, "Deep graphical feature learning for the feature matching problem," in *ICCV*, 2019.
[32] R. Burkard, S. Karisch, and F. Rendl, "Qaplib – a quadratic assignment problem library," *Journal of Global optimization*, 1997.
[33] E. Maset, F. Arrigoni, and A. Fusiello, "Practical and efficient multi-view matching," in *ICCV*, 2017, pp. 4568–4576.
[34] F. Scarselli, M. Gori, A. C. Tsoi, M. Hagenbuchner, and G. Monfardini, "The graph neural network model," *IEEE Transactions on Neural Networks*, vol. 20, no. 1, pp. 61–80, 2008.
[35] R. Burkard, M. DellAmico, and S. Martello, *Assignment Problems*. SIAM, 2009.
[36] J. Yan, C. Li, Y. Li, and G. Cao, "Adaptive discrete hypergraph matching," *IEEE Transactions on Cybernetics*, 2018.
[37] Q. Ngoc, A. Gautier, and M. Hein, "A flexible tensor block coordinate ascent scheme for hypergraph matching," in *CVPR*, 2015.
[38] J. Yan, Y. Tian, H. Zha, X. Yang, Y. Zhang, and S. Chu, "Joint optimization for consistent multiple graph matching," in *ICCV*, 2013.
[39] J. Yan, Y. Li, W. Liu, H. Zha, X. Yang, and S. Chu, "Graduated consistency-regularized optimization for multi-graph matching," in *ECCV*, 2014.
[40] T. Yu, J. Yan, W. Liu, and B. Li, "Incremental multi-graph matching via diversity and randomness based graph clustering," in *ECCV*, 2018.
[41] L. Torresani, V. Kolmogorov, and C. Rother, "Feature correspondence via graph matching: Models and global optimization," in *ECCV*, 2008.
[42] T. Caetano, J. McAuley, L. Cheng, Q. Le, and A. J. Smola, "Learning graph matching," *TPAMI*, vol. 31, no. 6, pp. 1048–1058, 2009.
[43] M. Leordeanu, R. Sukthankar, and M. Hebert, "Unsupervised learning for graph matching," *IJCV*, 2012.
[44] M. Leordeanu, A. Zanfir, and C. Sminchisescu, "Semi-supervised learning and optimization for hypergraph matching," in *ICCV*, 2011.
[45] R. Adams and R. Zemel, "Ranking via sinkhorn propagation," *arXiv:1106.1925*, 2011.
[46] T. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," *ICLR*, 2017.
[47] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," in *ICLR*, 2014.
[48] F. Zhou and F. Torre, "Factorized graph matching," *IEEE PAMI*, 2016.
[49] G. Mena, D. Belanger, S. Linderman, and J. Snoek, "Learning latent permutations with gumbel-sinkhorn networks," *ICLR*, 2018.
[50] R. Santa Cruz, B. Fernando, A. Cherian, and S. Gould, "Visual permutation learning," *TPAMI*, 2018.
[51] F. Serratosa, A. Solé-Ribalta, and X. Cortés, "Automatic learning of edit costs based on interactive and adaptive graph recognition," in *GbR*, 2011.
[52] L. Gong and Q. Cheng, "Exploiting edge features for graph neural networks," in *CVPR*, 2019.
[53] S. Bai, F. Zhang, and P. H. S. Torr, "Hypergraph convolution and hypergraph attention," 2019.
[54] Y. Feng, H. You, Z. Zizhao, R. Ji, and Y. Gao, "Hypergraph neural networks," *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, pp. 3558–3565, 07 2019.
[55] X. Zhou, M. Zhu, and K. Daniilidis, "Multi-image matching via fast alternating minimization," in *ICCV*, 2015.
[56] C. Ionescu, O. Vantzos, and C. Sminchisescu, "Matrix backpropagation for deep networks with structured layers," in *ICCV*, 2015.
[57] E. Khalil, H. Dai, Y. Zhang, B. Dilkina, and L. Song, "Learning combinatorial optimization algorithms over graphs," in *NIPS*, 2017.
[58] I. Sutskever, J. Martens, G. Dahl, and G. Hinton, "On the importance of initialization and momentum in deep learning," in *Proceedings of the 30th International Conference on International Conference on Machine Learning - Volume 28*, ser. ICML13. JMLR.org, 2013, p. III1139III1147.
[59] M. Leordeanu, M. Hebert, and R. Sukthankar, "An integer projected fixed point method for graph matching and map inference," in *NIPS*, 2009.
[60] A. Egozi, Y. Keller, and H. Guterman, "A probabilistic approach to spectral graph matching," *TPAMI*, 2013.
[61] Z. Liu, H. Qiao, and L. Xu, "An extended path following algorithm for graph-matching problem," *TPAMI*, pp. 1451–1456, 2012.
[62] P. M. Hahn and J. Krarup, "A hospital facility layout problem finally solved," *Journal of Intelligent Manufacturing*, vol. 12, no. 5-6, pp. 487–496, 2001.
[63] L. Bourdev and J. Malik, "Poselets: Body part detectors trained using 3d human pose annotations," in *ICCV*, 2009.
[64] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *CVPR*, 2009.
[65] J. Jiang, Y. Wei, Y. Feng, J. Cao, and Y. Gao, "Dynamic hypergraph neural networks," in *IJCAI*, 2019, pp. 2635–2641.



Runzhong Wang is currently a PhD Candidate with Department of Computer Science and AI Institute, Shanghai Jiao Tong University. He obtained B.E. in Electrical Engineering from Shanghai Jiao Tong University. He serves a reviewer for CVPR 2020. His research interests include machine learning and computer vision. He open-sourced and maintains the deep graph matching solver available at: https://github.com/Thinklab-SJTU/PCA-GM



Junchi Yan (M'10) is currently an Associate Professor with Department of Computer Science and Engineering, and AI Institute, Shanghai Jiao Tong University, China. Before that, he was a Senior Research Staff Member and Principal Scientist with IBM Research – China, where he started his career in April 2011. He obtained the Ph.D. in Electrical Engineering from Shanghai Jiao Tong University. He received the ACM China Doctoral Dissertation Nomination Award and China Computer Federation Doctoral Dissertation Award. His research interests include machine learning and computer vision. He serves as an Associate Editor for IEEE ACCESS, Area Chair for ICPR 2020, CVPR 2021 and Senior PC for CIKM 2019.

Xiaokang Yang (M'00-SM'04-F'19) received the B. S. degree from Xiamen University, in 1994, the M. S. degree from Chinese Academy of Sciences in 1997, and the Ph.D. degree from Shanghai Jiao Tong University in 2000. He is currently a Distinguished Professor of School of Electronic Information and Electrical Engineering, Shanghai Jiao Tong University, Shanghai, China. His research interests include visual signal processing and communication, media analysis and retrieval, and pattern recognition. He serves as an Associate Editor of IEEE Transactions on Multimedia, IEEE Signal Processing Letters. He is a Fellow of IEEE.