IA-GM: A Deep Bidirectional Learning Method for Graph Matching

Kaixuan Zhao, Shikui Tu, Lei Xu
Department of Computer Science and Engineering, Shanghai Jiao Tong University
Centre for Cognitive Machines and Computational Health (CMaCH), Shanghai Jiao Tong University
{zhaokx3, tushikui, leixu}@sjtu.edu.cn

Abstract

Existing deep learning methods for graph matching (GM) problems usually considered affinity learning to assist combinatorial optimization in a feedforward pipeline, and parameter learning is executed by back-propagating the gradients of the matching loss. Such a pipeline pays little attention to the possible complementary benefit from the optimization layer to the learning component. In this paper, we overcome the above limitation under a deep bidirectional learning framework. Our method circulates the output of the GM optimization layer to fuse with the input for affinity learning. Such direct feedback enhances the input by a feature enrichment and fusion technique, which exploits and integrates the global matching patterns from the deviation of the similarity permuted by the current matching estimate. As a result, the circulation enables the learning component to benefit from the optimization process, taking advantage of both global feature and the embedding result which is calculated by local propagation through node-neighbors. Moreover, circulation consistency induces an unsupervised loss that can be implemented individually or jointly to regularize the supervised loss. Experiments on challenging datasets demonstrate the effectiveness of our methods for both supervised learning and unsupervised learning.

Introduction

Many correspondence and similarity learning problems can be naturally represented as graph matching (GM) problems in real world applications such as shape matching (Belongie, Malik, and Puzicha 2002), multimedia retrieval (Chu and Chang 2015), object tracking and categorization (Duchenne, Joulin, and Ponce 2011), etc. GM refers to establishing correspondences between two graphs or among multiple graphs, which are encoded by 1st-order node information and 2nd-order structural information, even higher-order features (Lee, Cho, and Lee 2011). In other words, GM is to find a one-to-one mapping f: $V_B \to V_A$ between the nodes of the two separate graphs such that $(u, v) \in E_B$ iff $(f(u), f(v)) \in E_A$, where $G_A = (V_A, E_A)$ and $G_B = (V_B, E_B)$, with $|V_A| = |V_B| = n$, which is said to be exact GM or isomorphism. However, due to the inevitable noise and deformation in practical applications, there is rarely exactly the same two graphs. People consider the more loose case, *i.e.*, finding the correspondence between vertex sets of given two graphs, so that the nodes and edges error of the two graphs is minimized, which is called the inexact graph matching (Livi and Rizzi 2013) or subisomorphism.

When taking the 2nd-order or edge features into consideration, and suppose $|V_A| = n_1 \leq |V_B| = n_2$, the pair-wise graph matching problem can be formulated as the following Lawler's **q**uadratic **a**ssignment **p**roblem (QAP) (Lawler 1963):

$$J(\mathbf{X}) = \mathbf{x}^T\mathbf{K}\mathbf{x},\ \mathbf{x} = \mathbf{vec}(\mathbf{X}), \tag{1}$$
$$\mathbf{X} \in \{0,1\}^{n_1 \times n_2},\ \mathbf{X}\mathbf{1}_{n_2} = \mathbf{1}_{n_1},\ \mathbf{X}^T\mathbf{1}_{n_1} \preceq \mathbf{1}_{n_2}.$$

where $\mathbf{x}$ denotes a column-wise vectorized replica of $\mathbf{X}$, $\mathbf{K} \in \mathbb{R}^{n_1 n_2 \times n_1 n_2}$ is the so-called affinity matrix, which is constructed as a square symmetric positive matrix. Specifically, $\mathbf{K}_{ia;jb}$ measures the similarity of attributes between the pairs of candidate correspondences $(i, j) \in E_A$ and $(a, b) \in E_B$, where a diagonal element means node-to-node affinity, and a non-diagonal element contains edge-to-edge affinity. $\mathbf{X}$ is called assignment matrix indicating the node correspondence, and it is a permutation matrix. $\mathbf{x} \in \{0,1\}^{n_1 n_2 \times 1}$ denotes a column-wise vectorized replica of $\mathbf{X}$. The general GM by QAP is known to be NP-hard, and it is intractable to acquire a global optimum in the case of large graphs. Therefore, researchers study approximate algorithms to seek inexact solutions.

Traditional GM methods are mostly investigated on predefined affinity models, *e.g.*, the elements of $\mathbf{K}$ is given by a Gaussian kernel with Euclidean distance between the features of nodes and edges. Then, the assignment matrix $\mathbf{X}$ is solved by either a combinatorial optimisation algorithm (Xu and Oja 1990; Conte et al. 2004; Livi and Rizzi 2013) or a learning algorithm (Xu 1994, 1995; Dang and Xu 2000; Xu and King 2001; Liu, Qiao, and Xu 2012). However, predefined hand-crafted structures and affinity functions are limited to capture the inherent structure of a real-world matching problem. A better GM approximation algorithm does not necessarily lead to higher matching accuracy, if the affinity model is inappropriate for the practical GM task with noise and uncertainty. Early methods tackled this issue by learning a set of parameters in the affinity function (Caetano et al. 2009; Leordeanu, Sukthankar, and Hebert 2012).

Recently, deep learning frameworks were adopted in (Zanfir and Sminchisescu 2018) for learning the affinity matrix, forming an end-to-end model for GM. It should be noted that this line of efforts on learning affinity is parallel to the previous research on devising approximate GM algorithms using predefined affinity. Later in (Wang, Yan, and Yang 2019a), deep graph embedding models were employed to learn both intra-graph and cross-graph affinity functions, relaxing a special form of Lawler's QAP as a linear assignment problem that can be efficiently solved. Furthermore, a QAP network was presented to directly learn with the affinity matrix, and then the matching problem became a vertex classification task (Wang, Yan, and Yang 2019b). However, the existing methods of learning GM mainly focus on devising a learning component to assist the combinatorial optimization in a feedforward pipeline, paying little attention to the possible feedback benefits from the optimization component to the learning component.

A new direction is firstly suggested under the bidirectional intelligence framework (Xu 2019b), featured by two key ideas. The first is to alternatively perform a bottom-up learning and a top-down optimizer to benefit from each other. The second is feature enrichment (Xu 2018) that facilitates learning affinity with chessboard-like inputs for solving traveling salesman problem (TSP) and GM. One general scheme IA-DSM is further suggested in (Xu 2019a) to tackle doubly stochastic matrix (DSM) featured combinatorial tasks, including TSP and GM. Here, we concentrates on the GM task to consolidate the IA-DSM scheme by further developing an IA-DSM-GM (or shortly IA-GM) method, with not only the bottom-up and the top-down improved to suit the GM problems but also supervised and unsupervised learning performed individually or jointly.

Our main contributions are summarized as follows:

- We develop an IA-GM method that alternates a top-down GM optimization component and a bottom-up graph structural embedding component, such that the two components can benefit from each other. The top-down implementations suggested in (Xu 2019a) is simply replaced by the Sinkhorn and Hungarian computation. The bottom-up is improved by learning graph neural network instead of an ordinary deep learning, with a feature enrichment and fusion technique to integrate the feedback directly from the optimization component and the similarity matrix between two input graphs.
- This IA-GM solver can be trained not only in an unsupervised way as suggested in (Xu 2019a) but also in a supervised way as those recent studies with help of ground-truth correspondences. With these ways implemented either individually or jointly, one is always benefited from a cycling consistency induced by bidirectional iteration. The consistency is featured by a cross-entropy loss between the assignment estimation through combinatorial optimization and the matching probability via learning.
- This IA-GM method is further extended to be able to learn the features over both directed and undirected graphs. This is fulfilled by introducing a weights-**s**hared **r**esidual **g**ated **g**raph **n**eural **n**etwork (SR-GGNN), where we fuse each node's own features and it's incoming and outgoing edges information into next level.
- Experiment results including ablation studies demonstrate the effectiveness of our presented components, *i.e.*, the SR-GGNN, learning the similarity between the graphs, learning the fusion of extracted similarity with the GM solution feedback. Results on benchmark datasets indicate that our method outperforms peer methods in both supervised and unsupervised learning.

Related Work

Advances in Learning GM

Traditionally, building affinity model and solving the correspondence problem are two independent steps, and the affinity functions are usually hand-crafted. In 2009, (Caetano et al. 2009) first proposed the use of parametric models to learn affinity. More generally, (Cho, Alahari, and Ponce 2013) designed a unified parameterized graph structure learning model, by turning a joint feature map into a vector, where each element denotes node-to-node (or edge-to-edge) similarity. A discriminative score function is given by introducing weights on all elements of this feature map. This formulation includes several learning methods as special cases, e.g., (Torresani, Kolmogorov, and Rother 2008; Leordeanu, Sukthankar, and Hebert 2012). However, such predefined parametric affinity model is prone to the limited flexibility of the affinity metric in the real-world matching.

Recently, a seminal work by (Zanfir and Sminchisescu 2018) provided an end-to-end model that learns an appropriate affinity matrix $\mathbf{K} \in \mathbb{R}^{n_1 n_2 \times n_1 n_2}$ for GM problem, but the affinity learning and GM optimization are still considered in an orthogonal way. Later, deep graph embedding networks were adopted in (Wang, Yan, and Yang 2019a) to capture the high-order structure into a node-wise affinity matrix, relaxing the QAP into a linear assignment problem, and (Yu et al. 2020) proposed a loss function that includes the Hungarian attention mechanism. Regarding the affinity matrix as an association graph for graph embedding, a network solver was proposed in (Wang, Yan, and Yang 2019b) for Lawler's QAP. Although learning has been shown extremely helpful for solving the GM optimization, whether the optimization component can assist the learning part in turn is not investigated. Different from (Zanfir and Sminchisescu 2018; Wang, Yan, and Yang 2019a,b; Yu et al. 2020) which considers learning and optimization in a feedforward pipeline, our method cycles the GM solution to further enrich the structural features, correct or enhance the affinity learning. Therefore, our method can make the learning component and optimization component benefit from each other.

A general bidirectional learning scheme, called IA-DSM, was first sketched in (Xu 2019a) for solving DSM featured combinatorial task, under the framework of bidirectional intelligence system. In the system, two domains are defined: A-domain, which denotes the real bodies or patterns in the ***A****ctual* world, and I-domain, which denotes the ***I****nner coding* domain. The IA-DSM is featured by an alternation of A-mapping (from A-domain into I-domain along inbound direction) and I-mapping (from I-domain into A-domain along outbound direction). Taking traveling salesman problem (TSP) for example, it was suggested in (Xu 2019a) to follow AlphaGo (Silver et al. 2016) to use a Convolutional Neural Network (CNN) to implement the A-mapping. The CNN embeds the current state into a policy and a value, to guide the I-mapping which is implemented by a class of neural networks algorithms for iteratively solving TSP.

Our methods falls in the framework of IA-DSM, but has the following developments. First, there is no detailed implementation of IA-DSM on the GM task in (Xu 2019a), and our method fills in the blank. Actually, our method is the first one to show that the IA-DSM works well for combinatorial tasks. Second, we use graph neural networks (GNN) instead of CNN for the A-mapping. Third, we use the optimization process by the Sinkhorn and Hungarian computation for the I-mapping.

GNN for Graph Representation

For GM, the representation learning of each graph in a pair should be able to influence each other through some mechanism such as sharing weights and cross-graph attention, which is more conducive to learning affinity. Recently, works on GM have used GNNs to encoder the graph features. Graph convolutional network (GCN) module was adopted for learning embedding on individual graphs in (Nowak et al. 2017; Wang, Yan, and Yang 2019a), on the association graph by the affinity matrix for QAP in (Wang, Yan, and Yang 2019b). Cross-graph communication was further considered in (Li et al. 2019; Wang, Yan, and Yang 2019a).

We also use GNN for graph embedding in the learning component. We devise our network, called SR-GGNN, by modifying the gated GNN in (Li et al. 2015), and we fuse each node's own feature and it's incoming and outgoing edges information into next level. This multi-directional information dissemination and merge process is more useful than the graph embedding in (Wang, Yan, and Yang 2019a) to learn higher-order features, especially on directed graphs. For the output for each node, we introduce residual learning to optimize feature representation performance.

Methods

Overview of Our Method

The proposed bidirectional learning method for the task of GM falls in the IA-DSM framework (Xu 2019a), and thus is called as IA-DSM-GM or shortly IA-GM. The scheme sketched in Fig.6 of Ref.(Xu 2019a) is refined and also extended into the one shown in Fig.1(a). It consists of a top-down I-mapping (blue arrows), a bottom-up A-mapping (purple arrows). The top-down optimization component takes the probabilistic assignment matrix $\mathbf{U}_0$ as input and computes the optimal matching matrix $\mathbf{V}^*$ at the output which satisfies the binary constraints as given in Eq.(1). The bottom-up learning component is fed with the pair of graphs extracted from real-world images, learns the node-wise and structure-wise affinity across graphs by a GNN module, fuse all the information by certain network layers, and finally computes a policy, *i.e.*, the probability matrix $\mathbf{P}$ whose entries indicate the confidences of node correspondences between two graphs. The policy $\mathbf{P}$ is used to guide matching procedure of the optimization component effectively searching for the optimal matching against noise. The key difference from those recent studies featured by merely feedforward pipeline is that our bidirectional method further enable the optimization component to assist the learning part. This is achieved by a direct feedback of the estimate GM solution into the graph embedding, and fulfilled by a proposed technique called feature enrichment and fusion.

For fair comparisons, we try to keep the configuration details of the learning component and the optimization component, similar to a recent state-of-the-art method (Wang, Yan, and Yang 2019a), i.e., we use GNN for graph embedding and use Sinkhorn layer and Hungarian computation for matching procedure. We also present an improvement by devising a SR-GGNN, as given in Fig.1(c)), to learn the intra-graph and cross-graph affinity, the high-order structure, more efficiently than the GNN in (Wang, Yan, and Yang 2019a). The permutation cross-entropy loss was also adopted to measure the error between predicted DSM and ground truth permutation matrix for supervised training. It should be noted that the development from the feedforward pipeline in (Wang, Yan, and Yang 2019a) to our bidirectional IA-GM can also be generalized to (Wang, Yan, and Yang 2019b; Wang et al. 2020) to directly learn Lawler's QAP. The IA-alternation of learning and optimization will iteratively improve the GM solution, better than the corresponding feedforward one.

Feature Extracting

There exist two ways for keypoints feature extraction. One is shape context (Belongie, Malik, and Puzicha 2002) for the CMU House/Hotel datasets, getting the feature $\mathbf{h}^{(0)}$ as:

$$\mathbf{h}^{(0)} = [h_1, \dots, h_n], \quad h_i = Shape\_Contexts(P_i) \qquad (2)$$

where P_i denotes the i-th node, $Shape\_Contexts(P_i)$ is referred to (Belongie, Malik, and Puzicha 2002) for details.

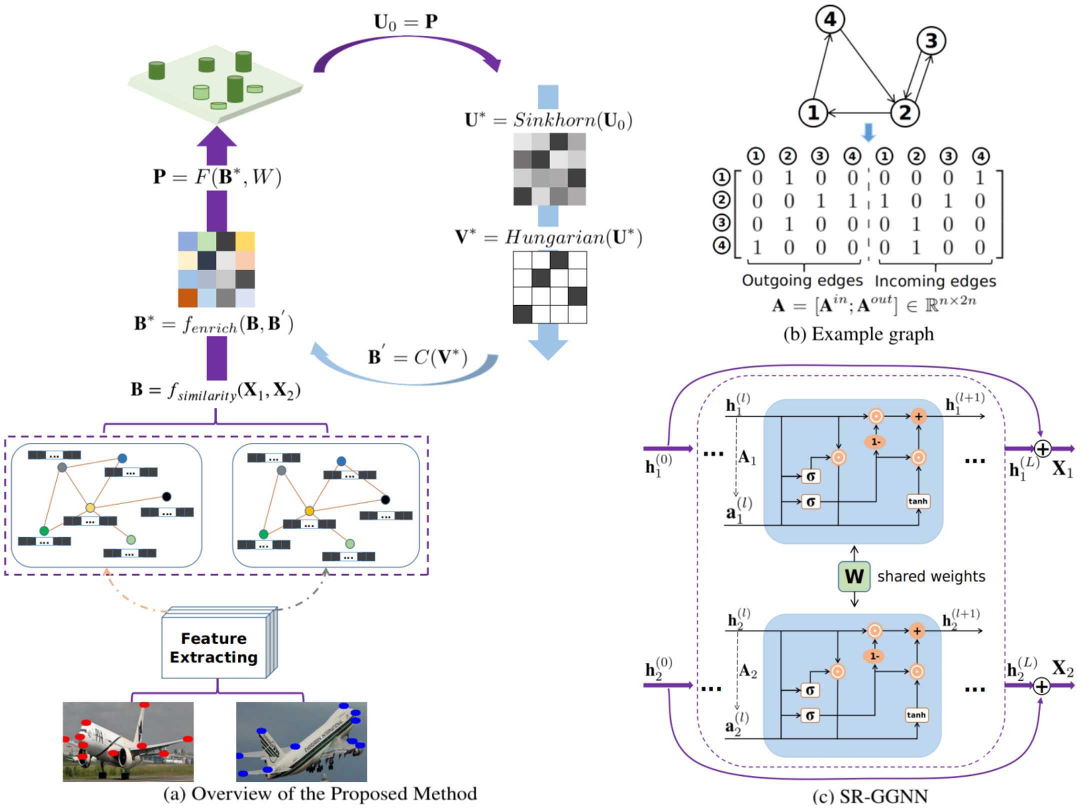
The other way constructs the feature by interpolating on CNN's feature map for PASCAL VOC Keypoints datasets. In line with (Wang, Yan, and Yang 2019a), we adopt the VGG-16 architecture (Simonyan and Zisserman 2014) to extract the node-features $\mathbf{U}$ from the output of *relu4_2* layer and the edge-features $\mathbf{F}$ from *relu5_1* layer, *i.e.*,

$$h_i = f_{cat}(Interp(P_i, \mathbf{U}), Interp(P_i, \mathbf{F})) \qquad (3)$$

where $Interp(P_i, \mathbf{U}/\mathbf{F})$ means bilinear interpolation of point P_i from $\mathbf{U}$ or $\mathbf{F}$, and f_{cat} denotes concatenation.

Representation Learning for Graph Affinity

For the graph embedding module, we use a Siamese network (Bromley et al. 1994), which consists of twin networks, for two input graphs respectively. The parameters between the twin networks are shared, encouraging that two similar graphs should be encoded close to each other in the embedding space. The node embedding is learnt from its node attributes and from neighborhood structure propagation. Specifically, we devise each of the twin networks based



$\mathbf{U}_0 = \mathbf{P}$
$\mathbf{U}^* = Sinkhorn(\mathbf{U}_0)$
$\mathbf{V}^* = Hungarian(\mathbf{U}^*)$
$\mathbf{B}' = C(\mathbf{V}^*)$
$\mathbf{P} = F(\mathbf{B}^*, W)$
$\mathbf{B}^* = f_{enrich}(\mathbf{B}, \mathbf{B}')$
$\mathbf{B} = f_{similarity}(\mathbf{X}_1, \mathbf{X}_2)$
Feature Extracting
Outgoing edges
Incoming edges
$\mathbf{A} = [\mathbf{A}^{in}; \mathbf{A}^{out}] \in \mathbb{R}^{n \times 2n}$
shared weights
W
$\mathbf{h}_1^{(0)}$
$\mathbf{h}_1^{(l)}$
$\mathbf{h}_1^{(l+1)}$
$\mathbf{A}_1$
$\mathbf{a}_1^{(l)}$
$\mathbf{h}_1^{(L)}$
$\mathbf{X}_1$
$\mathbf{h}_2^{(0)}$
$\mathbf{h}_2^{(l)}$
$\mathbf{h}_2^{(l+1)}$
$\mathbf{A}_2$
$\mathbf{a}_2^{(l)}$
$\mathbf{h}_2^{(L)}$
$\mathbf{X}_2$
tanh
1-

(a) Overview of the Proposed Method

(b) Example graph

(c) SR-GGNN

Figure 1: An overview of IA-GM. (a) The pipeline of IA-GM for graph matching problem, where $f_{similarity}$ and f_{enrich} are implemented by Eq.(7) and Eq.(9) respectively; (b) An example graph to explain the construction of adjacency matrix; (c) The structure of the residual gated graph neural network with weights-sharing.

on Gated GNN (GGNN). Initialized at $\mathbf{h}^{(0)}$ by Eq.(2)&(3) the embedding $\mathbf{h}$ after $l+1$ iterations is computed as:

$$\mathbf{h}_i^{(l+1)} = f_{GRU}(\mathbf{h}_i^{(l)}, \mathbf{a}_i^{(l)}), \tag{4}$$
$$\mathbf{a}_i^{(l)} = f_{aggr}(f_{in}(\mathbf{A}_i^{in}, \mathbf{h}_i^{(l)}), f_{out}(\mathbf{A}_i^{out}, \mathbf{h}_i^{(l)}), f_{self}(\mathbf{h}_i^{(l)}))$$

where the adjacency matrix of a graph $\mathbf{A} = [\mathbf{A}^{in}; \mathbf{A}^{out}] \in \mathbb{R}^{n \times 2n}$, $\mathbf{A}^{in}$ and $\mathbf{A}^{out}$ encodes incoming edges and outgoing edges respectively. An example of adjacency matrix is illustrated in Fig.1(b). The separation of incoming and outgoing edges allows the model to deal with both directed and undirected graphs.

It can be observed from Eq.(4) that multi-directional information transmission is enabled for each node. f_{in}, f_{out} and f_{self} are the functions to pass information along the incoming edges of the node, via the outgoing edges, within the node itself, respectively. Here, we adopt the simple linear mapping for the above three message-passing functions. Instead of $\mathbf{a}_i^{(l)} = \sum_{j \to i} \mathbf{h}_j^{(l)}$ in the original GGNN, all the information is aggregated by the function f_{aggr}, which is implemented as a concatenation followed by a fully-connected layer. Then, the aggregated information is fed into a gated recurrent unit (GRU) to update the embedding vector, with the GRU function $f_{GRU}(\mathbf{h}_i^{(l)}, \mathbf{a}_i^{(l)})$ given by:

$$\begin{cases} \mathbf{z}_i^{(l+1)} = \sigma(\mathbf{W}_z \mathbf{h}_i^{(l)} + \mathbf{U}_z \mathbf{a}_i^{(l)}) \\ \mathbf{r}_i^{(l+1)} = \sigma(\mathbf{W}_r \mathbf{h}_i^{(l)} + \mathbf{U}_r \mathbf{a}_i^{(l)}) \\ \widetilde{\mathbf{h}}_i^{(l+1)} = tanh(\mathbf{W}_h(\mathbf{h}_i^{(l)} \odot \mathbf{r}_i^{(l+1)}) + \mathbf{U}_h \mathbf{a}_i^{(l)}) \\ \mathbf{h}_i^{(l+1)} = (1 - \mathbf{z}_i^{(l+1)}) \odot \mathbf{h}_i^{(l)} + \mathbf{z}_i^{(l+1)} \odot \widetilde{\mathbf{h}}_i^{(l+1)} \end{cases} \tag{5}$$

where $\mathbf{z}$ and $\mathbf{r}$ are the update and reset gates, $\odot$ is the Hadamard point-wise multiplication operator.

Taking the idea of residual network (He et al. 2016), we compute the final embedding $\mathbf{x}_i$ for the i-th node by adding a skip connection from $\mathbf{h}^{(0)}$ to $\mathbf{h}^{(L)}$,

$$\mathbf{x}_i = Relu\left(g(\mathbf{h}_i^{(L)}) + \mathbf{W}_s \mathbf{h}_i^{(0)}\right) \tag{6}$$

where g is a fully-connected layer, L is the layer number of GGNN, $\mathbf{W}_s$ is a parameter to match the dimensions via shortcut connections. We call the above procedure by Eq.(4)-(6) as weights **s**hared **r**esidual **g**ated **g**raph **n**eural **n**etwork (SR-GGNN). The details of SR-GGNN are summarized in Fig.1(c). Compared with the embedding network by (Wang, Yan, and Yang 2019a), our SR-GGNN not only enhances the encoding of structure affinity between two graphs into the node-to-node affinity, allowing for more accurate reduction of the affinity matrix $\mathbf{K}$ in Eq.(1) into a linear one, but also allows to process directed graph data and accelerate the training process.

For a graph with n nodes, the graph embedding is given by $\mathbf{X} = [\mathbf{x}_1, \ldots, \mathbf{x}_n]$. Next, we calculate the similarity matrix between two input graphs via cosine similarity or vector inner product:

$$\mathbf{B} = f_{similarity}(\mathbf{\Lambda}_1\mathbf{X}_1, \mathbf{\Lambda}_2\mathbf{X}_2) \tag{7}$$

where $\mathbf{\Lambda}_1, \mathbf{\Lambda}_2 \in \mathbb{R}^{d\times d}$ are learning parameters, and $\mathbf{X}_1$, $\mathbf{X}_2$ are the graph representations of two graphs respectively.

Feature Enrichment and Affinity Fusion

We directly feed the output by the top-down optimization component, *i.e.*, the permutation or matching matrix $\mathbf{V}^*$ for the node correspondence, back into the learning component. In an intermediate training step, the matrix $\mathbf{V}^*$ may be an inexact but close to optimal solution, and the information of how good the current approximation is to the optimal one is helpful for the embedding component to adjust the parameters. In addition to backpropagating the error gradients, we propose a feature enrichment and fusion technique to appropriately exploit this direct feedback, enhancing the graph structure affinity embedding. This can be fulfilled by noticing the following matching consistency:

$$\mathbf{X}_2 \leftarrow (\mathbf{V}^*)^T\mathbf{X}_1; \quad \mathbf{X}_1 \leftarrow \mathbf{V}^*\mathbf{X}_2 \tag{8}$$

where $\mathbf{X}_1, \mathbf{X}_2 \in \mathbb{R}^{n\times d}$ are embeddings of two input graphs with feature dimension d. Based on the correspondence by Eq.(8), we fuse the graph affinity in the embedding space with the one permuted by $\mathbf{V}^*$ in a linear mixture:

$$\mathbf{B}^* = \alpha_1\mathbf{B} + \alpha_2\mathbf{B}^{'} \tag{9}$$

where $\mathbf{B}$ is given by Eq.(7), and $\mathbf{B}^{'}$ is the similarity between the embeddings of the permuted graphs, being computed by replacing $\mathbf{X}_1, \mathbf{X}_2$ with $\mathbf{V}^*\mathbf{X}_2$, $(\mathbf{V}^*)^T\mathbf{X}_1$ in Eq.(7). α_1 and α_2 are the contribution ratios for the two terms. For supervised learning, we trust the learning component ($\mathbf{B}$ term) more when datasets is simpler, and conversely we trust the optimization component ($\mathbf{B}^{'}$ term) more. For unsupervised learning, we set a larger ratio of α_2/α_1 to optimize the solution space iteratively, and more details will be discussed in the experimental section. Finally, $\mathbf{B}^*$ is considered as a chessboard-like image as illustrated in Fig.1(a), and fed into the next network layers for further processing.

The IA Alternation

Our method is featured by an alternation of learning component and optimization component. Since the two components corresponds to A-mapping and I-mapping respectively, we called it IA-alternation following the convention in (Xu 2019b). GM is to optimize an objective, e.g., Eq.(1), with respect to a permutation matrix. Similar to (Wang, Yan, and Yang 2019a), the above graph embedding model allows reducing the second-order affinity matrix $\mathbf{K}$ into a linear one, and thus GM can be subsequently solved efficiently by Sinkhorn normalization and Hungarian computation.

Based on the similarity $\mathbf{B}^*$ computed by Eq.(9), we calculate the probability matrix $\mathbf{P}$, which satisfies $\mathbf{P} = [p_{ij}]$ and $0 \le p_{ij} \le 1, \sum_j p_{ij} = 1$ as follows:

$$\mathbf{P} = F(\mathbf{B}^*, W) = softmax(W \odot \alpha\mathbf{B}^*) \tag{10}$$

where $W \in \mathbb{R}^{d\times d}$ is a parameter to be learned, α is a large super-parameter used to push the probability values towards zero or one. Then, we assign $\mathbf{P}$ to be $\mathbf{U}_0$,

$$\mathbf{U}_0 = \mathbf{P}. \tag{11}$$

$\mathbf{U}_0$ is regarded as the linear affinity matrix, and is taken as an initialization to guide the Sinkhorn iterative computation for searching a good DSM $\mathbf{U}^*$.

Following (Adams and Zemel 2011), the Sinkhorn normalization $\mathbf{S}(\mathbf{X})$ is defined recursively for $1 \le k \le K$ as:

$$\mathbf{S}^k(\mathbf{X}) = \begin{cases} \exp(\mathbf{X}), & \text{if } k = 0 \\ \boldsymbol{\tau}_c(\boldsymbol{\tau}_r(\mathbf{S}^{k-1}(\mathbf{X}))), & otherwise. \end{cases} \tag{12}$$

$$\mathbf{U}^* = Sinkhorn(\mathbf{U}_0) = \mathbf{S}^K(\mathbf{U}_0/\tau) \tag{13}$$

where $\boldsymbol{\tau}_c(\mathbf{X}) = \mathbf{X} \oslash (\mathbf{1}_n\mathbf{1}_n^T\mathbf{X})$ is the column normalization operator and $\boldsymbol{\tau}_r(\mathbf{X}) = \mathbf{X} \oslash (\mathbf{X}\mathbf{1}_n\mathbf{1}_n^T)$ is the row normalization operator, with $\oslash$ denoting the element-wise division. When given the number of iterations K, we can calculate a DSM via Eq.(13). τ is an introduced temperature parameter (Emami and Ranka 2018). In the limit $\tau \to 0$, each element of $\mathbf{U}^*$ is closer to one or zero, *i.e.*, $\mathbf{U}^*$ is closer to a permutation matrix. To satisfy the binary constraints of permutation matrix, we discretize $\mathbf{U}^*$ via Hungarian algorithm, and obtain permutation matrix $\mathbf{V}^*$ as the predicted solution:

$$\mathbf{V}^* = Hungarian(\mathbf{U}^*), \mathbf{V}^* = [v^*_{ij}], i, j = 1, ..., n. \tag{14}$$

It should be noted that the Eq.(11) passes the learning output $\mathbf{P}$ to the input of the matching procedure, while the Eq.(9) feeds the matching output $\mathbf{V}^*$ back to the learning component. This completes a circle of IA-alternation. The matrix $\mathbf{P}$ encodes the affinity between graphs via graph embedding, and it serves like a searching policy in AlphaGo to guide the optimization component. The predicted GM solution $\mathbf{V}^*$ enhances the affinity learning by feeding back the deviation of the similarity permuted by $\mathbf{V}^*$, so that the affinity is learnt more appropriate for the GM optimization to calculate correct correspondence. Therefore, the IA-alternation benefits the algorithm from both sides.

Loss Function

We utilize the ground truth label to supervise the end-to-end training of our method. The objective is to minimize the following permutation cross-entropy loss:

$$\min -\frac{1}{n}\sum_{i,j}\left[u^{gt}_{ij} \ln u^*_{ij} + (1 - u^{gt}_{ij}) \ln(1 - u^*_{ij})\right], \tag{15}$$

where $\mathbf{U}^* = [u^*_{ij}]$, $\mathbf{U}^{gt} = [u^{gt}_{ij}], u^{gt}_{ij} \in \{0, 1\}$ denotes the ground truth permutation matrix. Proposed in (Wang, Yan, and Yang 2019a), the permutation loss has been demonstrated its effectiveness for the combinatorial nature GM, better than the structured max-margin loss (Cho, Alahari, and Ponce 2013) and the pixel offset loss (Zanfir and Sminchisescu 2018).

In addition to the above supervised loss, our method can also be extended to work without supervision. This nature is induced by a cycling consistency between the affinity learning output and matching optimization output during the IA-alternation. Specifically, we maximize the index alignment

Accuracy(%) \ Class / Method	aero	bike	bird	boat	bottle	bus	car	cat	chair	cow	table	dog	horse	mbike	person	plant	sheep	sofa	train	tv	**mean**
GMN	37.24	51.34	49.71	46.60	76.34	70.73	64.47	61.32	34.41	55.53	60.10	56.24	57.72	52.89	33.96	73.60	62.24	42.26	**83.39**	89.48	57.98
PIA-GM	49.68	62.07	58.88	55.93	76.49	70.54	68.51	69.72	38.30	60.28	49.28	61.07	61.60	61.71	**45.03**	79.50	64.60	54.35	82.13	90.03	62.97
PIA-GM(SR-GGNN)	49.84	**66.92**	59.61	54.23	76.82	**76.95**	68.22	70.23	38.42	62.67	44.14	64.97	66.75	57.76	44.62	81.32	66.32	52.65	79.70	89.85	63.60
PCA-GM	50.23	61.32	60.52	54.19	77.87	73.65	**68.88**	69.98	**39.86**	63.01	52.48	63.90	62.94	64.52	44.08	82.03	**68.23**	51.90	77.50	**90.93**	63.90
Ours	**51.57**	66.60	**63.08**	**56.01**	**80.07**	74.05	68.01	**72.06**	38.88	**65.69**	**73.64**	**65.05**	**67.12**	**66.08**	44.94	**82.01**	68.12	**59.38**	78.46	90.88	**66.58**
PCA-GM*	51.67	61.98	68.24	55.65	**80.82**	71.64	74.43	66.49	38.45	59.90	73.44	59.07	56.97	60.49	43.76	**88.38**	58.78	53.78	**90.21**	90.66	65.24
Ours*	**53.91**	**67.73**	**68.79**	**60.15**	80.28	**75.06**	**76.96**	**72.00**	**40.23**	**65.55**	**79.56**	**65.36**	**66.16**	**67.43**	**46.28**	87.39	**65.41**	**58.51**	89.36	**90.68**	**68.84**

Table 1: Supervised learning on Pascal Keypoints datasets. Methods with superscript * denote in-class training and testing, otherwise cross-class. The best results for each semantic class and average accuracy are marked in bold.

by cross-entropy between probability matrix $\mathbf{P}$ from the learning component and matching matrix $\mathbf{V}^*$ by the optimization component, *i.e.*, aligning v^*_{ij} with p_{ij}:

$$\max \sum_{i,j} v^*_{ij} \ln p_{ij} + R(\theta) \tag{16}$$

where $R(\theta) = -\|\theta\|^2$ is a regularization term over the parameters θ. This comes from Equation (7) of Ref.(Xu 2019a) and is similar to the AlphaGo Zero's cross-entropy loss between the neural network's move probability and the tree search policy (Silver et al. 2017). It is different from maximizing the correlation between the confidence matrix and its binary version by the power method (Leordeanu, Sukthankar, and Hebert 2012). It is noted that the Eq.(15) and Eq.(16) can be optimized individually or jointly.

Experiments

In this section, we evaluate our model on benchmark datasets. Consistent with the literature, we define the matching accuracy as $\frac{1}{n}(\mathbf{V}^* \wedge \mathbf{V}^{gt})$, where $\mathbf{V}^* \in \{0,1\}^{n\times n}$ is the GM solution, $\wedge$ is the Logical AND, and $\mathbf{V}^{gt}$ is the ground truth permutation matrix.

Supervised Learning

We use PASCAL VOC dataset with keypoints annotation (Bourdev and Malik 2009) for the case of supervised learning. The dataset contains 20 semantic classes with 3510 pairs of annotated examples for training and 841 pairs for testing. PASCAL Keypoints is a challenging dataset because there are not only differences in angle, size and style between the paired images, and but also inaccurate keypoints information and varying number of keypoints from one class to another. For fair comparisons, we construct training and testing sets according to the following two scenarios:

- *cross-class*: Each pair of data used for training and testing is randomly selected from all 20 semantic classes.
- *in-class*: For each class, we build a random subset for training, while the rest are for testing.

We include for comparisons previous closely-related GM methods: GMN (Zanfir and Sminchisescu 2018), intra-graph affinity based GM (PIA-GM) and cross-graph affinity based GM (PCA-GM) (Wang, Yan, and Yang 2019a), and PIA-GM(SR-GGNN). The PIA-GM(SR-GGNN) is implemented by replacing the deep graph embedding module GCN with our SR-GGNN module, while keeping other components of PIA-GM unchanged. It is included as a baseline to evaluate the effectiveness of our SR-GGNN component.

The supervised version of our method is implemented by Eq.(15) Here, we provide the details for several key parts. Pretrained VGG16 is used to extract features for affinity learning. In practice, we find that the layer number L of the modified GRU computation by Eq.(4) is best set at $L = 1$, because stacked multi-layer network may cause over-smoothing problem for the propagation of node embedding. The more stacked layers, the more neighbors of the node will be involved, and then the final representations of all nodes tend to be consistent. The parameters $\mathbf{\Lambda}_1$ and $\mathbf{\Lambda}_2$ in Eq.(7) are initialized as $\mathbf{I} + \boldsymbol{\varepsilon}$, where $\mathbf{I}$ an identity matrix, $\boldsymbol{\varepsilon}$ is a random perturbation matrix with entries generated from a uniform distribution $U[-1/\sqrt{d}, 1/\sqrt{d}]$, and d is the feature dimension. For Eq.(10), we initialize W as $\mathbf{1}_{n\times n} + \boldsymbol{\varepsilon}$ with $\mathbf{1}_{n\times n}$ being an $n \times n$ all-one matrix, and set $\alpha = 40$ to increase the difference between probability values.

We set $\alpha_1 = 0.75$ and $\alpha_2 = 1.25$ in Eq.(9) to trust $\mathbf{B}'$ term more because of this complex datasets of high variation and noise. In practice, we found that values satisfying $\alpha_1 \in [0.5, 1.5]$, $\alpha_1 + \alpha_2 = 2$, actually lead to a small variation in accuracy. We set the number of IA-iterations to 4 for VOC datasets. In Eq.(13), technically the temperature parameter $\tau \to 0$ should be close to zero as much as possible, but in practice, the too-low temperature usually leads to high variations in gradients. Thus, we use $\tau = 0.05$ as default hyperparameter. Delaunay triangulation is adopted to build graphs.

The results of supervised learning are shown in Table 1. It can be observed that our method is the best for most semantic classes and also achieves the highest average accuracy, under either cross-class or in-class scenarios. As expected, the in-class scenario is easier than the cross-class one. Also, we notice that PIA-GM is improved to be close to PCA-GM when using our SR-GGNN module instead of using GCN for intra-graph affinity learning in (Wang, Yan, and Yang 2019a). It implies that our SR-GGNN is more effective in graph structure embedding. The slight advantage of PCA-GM over PIA-GM(SR-GGNN) indicates that it is still useful to use cross-graph information. Comparing with PCA-GM, our proposed method achieved a 2.7% improvement for cross-class, and 3.6% improvement for in-class.

Next, we extend our experiments on directed graphs. We use the Cars and Motorbikes pairs (Leordeanu, Sukthankar, and Hebert 2012) to compare with other directed GM methods. This dataset consists of 30 pairs of car images and 20 pairs of motorbike images (60 percent for training and 40 percent for testing), the number of inliers for each pair ranges from 15 to 52. This dataset is simpler than the PASCAL VOC Keypoints so that the original features space has rich information to learn the correspondence. Therefore, we set $\alpha_1 = 1.25$ and $\alpha_2 = 0.75$ in Eq.(9) to trust $\mathbf{B}$ term more. As shown in Tab.2, compared with PM (Zass and Shashua 2008), SMAC (Cour, Srinivasan, and Shi 2007), IPFP-S (Leordeanu, Hebert, and Sukthankar 2009), RRWM (Cho, Lee, and Lee 2010) and FGM-D (Zhou and De la Torre 2015), our method achieved the highest average accuracy. It should be noted that our method does not additionally compute the directional edge affinity, but only use the basic information, *i.e.*, the direction of the edges.

Accuracy(%) / Method / Dataset	PM	SMAC	IPFP-S	RRWM	FGM-D	Ours
Cars	28.2	74.4	81.0	89.6	89.7	**94.8**
Motorbikes	36.2	82.6	81.2	92.1	**95.5**	92.6
mean	32.2	78.5	81.1	90.9	92.6	**93.7**

Table 2: Directed GM on Cars/Motorbikes datasets.

Unsupervised Learning

For unsupervised learning, our network is implemented via Eq.(16). We perform our method on the CMU House/Hotel datasets (House: 111 images and Hotel: 101 images) for comparisons with existing unsupervised learning methods. We conduct experiments under the cross-class scenario. Since the CMU House/Hotel datasets are simpler than the VOC datasets, the contribution of feature enrichment $\mathbf{B}'$ by Eq.(9) is reliable. Then, we increase the ratio of α_2/α_1, and set $\alpha_1 = 0.5$ and $\alpha_2 = 1.5$ to balance the best performance. The remaining hyperparameter settings are consistent with supervised learning. Table 3 summarizes the experiment results, where the results of other methods come directly from (Leordeanu, Sukthankar, and Hebert 2012). It can be observed that our method outperforms the existing unsupervised methods for GM. Matching examples can be seen in Fig.2. Therefore, our unsupervised method is also very effective to solve real world GM problems.

Accuracy(%) / Dateset / Method	house	hotel	**mean**
(Caetano et al. 2009), sup	<84	<87	<85.5
(Leordeanu and Hebert 2008), sup	**99.8**	94.8	97.3
(Leordeanu, Sukthankar, and Hebert 2012), unsup	**99.8**	94.8	97.3
Ours, unsup	99.5	**97.8**	**98.7**

Table 3: Unsupervised learning on CMU House/Hotel datasets. (sup: supervised; unsup: unsupervised)

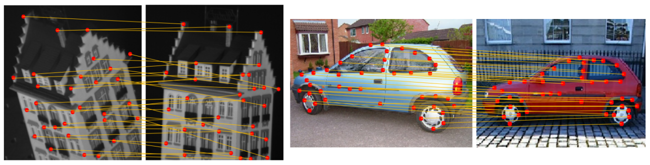


Figure 2: Examples of matching on the different datasets.

Ablation Studies

We evaluate the roles of the key parts of our model by adding them one-by-one to a simple baseline. Here, the baseline is a feedforward pipeline that replaces the GCN module in PIA-GM (Wang, Yan, and Yang 2019a) with a simple fully-connected (FC) layer. Then, we replace the FC layer with SR-GGNN module for graph embedding by Eq.(4)-(6), activate the direct feedback in feature enrichment and fusion by setting $\alpha_2 > 0$ in Eq.(9), activate the parameter learning in Eq.(7) for similarity, activate the parameter learning in Eq.(10) for the probability matrix. According to the results shown in Table 4, activating the direct feedback in the part of feature enrichment and fusion contributes a high gain (3.06%) to the accuracy. Actually, if comparing to the accuracy in Table 1, the gain from GCN to SR-GGNN is 0.63%, much smaller than 3.06%, indicating that the direct feedback to assist the affinity learning is indeed the main factor for performance improvement over the feedforward pipelines.

SR-GGNN	IA-alternation			Accuracy(%)	Gain(%)
	Feature Enrichment	Similarity Learning	Probability matrix learning		
N	N	N	N	58.95	–
Y	N	N	N	63.04	↑ **4.09**
Y	Y	N	N	66.10	↑ **3.06**
Y	Y	Y	N	66.32	↑ 0.22
Y	Y	Y	Y	**66.58**	↑ 0.26

Table 4: Results of ablation studies. "Y" means the learning for the column is activated, while "N" means not.

Conclusion

We have proposed a deep bidirectional learning method named IA-GM for graph matching problem. Our method is featured by an IA-alternation between an optimization component for matching procedure and a learning component for graph affinity. Existing methods for learning GM usually use the learning component to assist optimization in a feed-forward pipeline. In contrast, our method imposes a direct feedback of the output of the matching procedure into the graph embedding of the learning component, so that affinity learning is also enhanced. This is fulfilled by a feature enrichment and fusion technique, which exploits the deviation of the similarity permuted by the current matching estimate. Ablation studies verify that such direct feedback indeed contributes the most to the performance improvement. Moreover, our model are able to be trained without supervision, and to be extended on directed graphs. Experiments on benchmark datasets verify the advantages of our method over the existing related methods.

Acknowledgements

This work was supported by National Science and Technology Innovation 2030 Major Project (2018AAA0100700) of the Ministry of Science and Technology of China, and Shanghai Municipal Science and Technology Major Project (2021SHZDZX0102). Shikui Tu and Lei Xu are corresponding authors.

References

Adams, R. P.; and Zemel, R. S. 2011. Ranking via sinkhorn propagation. *arXiv preprint arXiv:1106.1925* .

Belongie, S.; Malik, J.; and Puzicha, J. 2002. Shape matching and object recognition using shape contexts. *IEEE transactions on pattern analysis and machine intelligence* 24(4): 509–522.

Bourdev, L.; and Malik, J. 2009. Poselets: Body part detectors trained using 3d human pose annotations. In *2009 IEEE 12th International Conference on Computer Vision*, 1365–1372. IEEE.

Bromley, J.; Guyon, I.; LeCun, Y.; Säckinger, E.; and Shah, R. 1994. Signature verification using a" siamese" time delay neural network. In *Advances in neural information processing systems*, 737–744.

Caetano, T. S.; McAuley, J. J.; Cheng, L.; Le, Q. V.; and Smola, A. J. 2009. Learning graph matching. *IEEE transactions on pattern analysis and machine intelligence* 31(6): 1048–1058.

Cho, M.; Alahari, K.; and Ponce, J. 2013. Learning Graphs to Match. In *IEEE International Conference on Computer Vision*.

Cho, M.; Lee, J.; and Lee, K. M. 2010. Reweighted random walks for graph matching. In *European conference on Computer vision*, 492–505. Springer.

Chu, W.-T.; and Chang, F.-C. 2015. A privacy-preserving bipartite graph matching framework for multimedia analysis and retrieval. In *Proceedings of the 5th ACM on International Conference on Multimedia Retrieval*, 243–250.

Conte, D.; Foggia, P.; Sansone, C.; and Vento, M. 2004. Thirty years of graph matching in pattern recognition. *International journal of pattern recognition and artificial intelligence* 18(03): 265–298.

Cour, T.; Srinivasan, P.; and Shi, J. 2007. Balanced graph matching. In *Advances in Neural Information Processing Systems*, 313–320.

Dang, C.; and Xu, L. 2000. A barrier function method for the nonconvex quadratic programming problem with box constraints. *Journal of Global Optimization* 18(2): 165–188.

Duchenne, O.; Joulin, A.; and Ponce, J. 2011. A graph-matching kernel for object categorization. In *2011 International Conference on Computer Vision*, 1792–1799. IEEE.

Emami, P.; and Ranka, S. 2018. Learning permutations with sinkhorn policy gradient. *arXiv preprint arXiv:1805.07010* .

He, K.; Zhang, X.; Ren, S.; and Sun, J. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 770–778.

Lawler, E. L. 1963. The quadratic assignment problem. *Management science* 9(4): 586–599.

Lee, J.; Cho, M.; and Lee, K. M. 2011. Hyper-graph matching via reweighted random walks. In *CVPR 2011*, 1633–1640. IEEE.

Leordeanu, M.; and Hebert, M. 2008. Smoothing-based optimization. In *2008 IEEE Conference on Computer Vision and Pattern Recognition*, 1–8. IEEE.

Leordeanu, M.; Hebert, M.; and Sukthankar, R. 2009. An integer projected fixed point method for graph matching and map inference. In *Advances in neural information processing systems*, 1114–1122.

Leordeanu, M.; Sukthankar, R.; and Hebert, M. 2012. Unsupervised learning for graph matching. *International journal of computer vision* 96(1): 28–45.

Li, Y.; Gu, C.; Dullien, T.; Vinyals, O.; and Kohli, P. 2019. Graph matching networks for learning the similarity of graph structured objects. *arXiv preprint arXiv:1904.12787* .

Li, Y.; Tarlow, D.; Brockschmidt, M.; and Zemel, R. 2015. Gated graph sequence neural networks. *arXiv preprint arXiv:1511.05493* .

Liu, Z.-Y.; Qiao, H.; and Xu, L. 2012. An extended path following algorithm for graph-matching problem. *IEEE transactions on pattern analysis and machine intelligence* 34(7): 1451–1456.

Livi, L.; and Rizzi, A. 2013. The graph matching problem. *Pattern Analysis and Applications* 16(3): 253–283.

Nowak, A.; Villar, S.; Bandeira, A. S.; and Bruna, J. 2017. A note on learning algorithms for quadratic assignment with graph neural networks. *stat* 1050: 22.

Silver, D.; Huang, A.; Maddison, C. J.; Guez, A.; Sifre, L.; Van Den Driessche, G.; Schrittwieser, J.; Antonoglou, I.; Panneershelvam, V.; Lanctot, M.; et al. 2016. Mastering the game of Go with deep neural networks and tree search. *nature* 529(7587): 484.

Silver, D.; Schrittwieser, J.; Simonyan, K.; Antonoglou, I.; Huang, A.; Guez, A.; Hubert, T.; Baker, L.; Lai, M.; Bolton, A.; et al. 2017. Mastering the game of go without human knowledge. *Nature* 550(7676): 354–359.

Simonyan, K.; and Zisserman, A. 2014. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556* .

Torresani, L.; Kolmogorov, V.; and Rother, C. 2008. Feature Correspondence Via Graph Matching: Models and Global Optimization. In *European Conference on Computer Vision-eccv*.

Wang, R.; Yan, J.; and Yang, X. 2019a. Learning combinatorial embedding networks for deep graph matching. In *Proceedings of the IEEE International Conference on Computer Vision*, 3056–3065.

Wang, R.; Yan, J.; and Yang, X. 2019b. Neural Graph Matching Network: Learning Lawler's Quadratic Assignment Problem with Extension to Hypergraph and Multiple-graph Matching.

Wang, T.; Liu, H.; Li, Y.; Jin, Y.; Hou, X.; and Ling, H. 2020. Learning Combinatorial Solver for Graph Matching. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*.

Xu, L. 1994. Combinatorial optimization neural nets based on a hybrid of Lagrange and transformation approaches. In *Proceedings of World Congress on Neutral Networks*, 399–404.

Xu, L. 1995. On the hybrid LT combinatorial optimization: new U-shape barrier, sigmoid activation, least leaking energy and maximum entropy. In *Proc. ICONIP*, volume 95, 309–312.

Xu, L. 2018. Deep bidirectional intelligence: AlphaZero, deep IA-search, deep IA-infer, and TPC causal learning. In *Applied Informatics*, volume 5, 1–38. Springer.

Xu, L. 2019a. Deep IA-BI and Five Actions in Circling. In *LNCS: Proc. 2019 Intelligent Science and Big Data Engineering (Oct.17-20)*, 1–21. Springer.

Xu, L. 2019b. An Overview and Perspectives On Bidirectional Intelligence: Lmser Duality, Double IA Harmony, and Causal Computation. *IEEE/CAA Journal of Automatica Sinica* 6(4): 865–893.

Xu, L.; and King, I. 2001. A PCA approach for fast retrieval of structural patterns in attributed graphs. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)* 31(5): 812–817.

Xu, L.; and Oja, E. 1990. Improved simulated annealing, Boltzmann machine, and attributed graph matching. In *Lecture Notes in Computer Sciences 412 : Neural Networks*, volume 412, 151–160. Springer Berlin Heidelberg.

Yu, T.; Wang, R.; Yan, J.; and Li, B. 2020. Learning deep graph matching with channel-independent embedding and Hungarian attention. In *ICLR2020*.

Zanfir, A.; and Sminchisescu, C. 2018. Deep learning of graph matching. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2684–2693.

Zass, R.; and Shashua, A. 2008. Probabilistic graph and hypergraph matching. In *2008 IEEE Conference on Computer Vision and Pattern Recognition*, 1–8. IEEE.

Zhou, F.; and De la Torre, F. 2015. Factorized graph matching. *IEEE transactions on pattern analysis and machine intelligence* 38(9): 1774–1789.