Machine Learning (2024) 113:1743–1769
https://doi.org/10.1007/s10994-022-06255-z

IA-NGM: A bidirectional learning method for neural graph matching with feature fusion

Tianxiang Qin[1] · **Shikui Tu**[1] · **Lei Xu**[1]

Received: 28 May 2022 / Revised: 10 August 2022 / Accepted: 19 September 2022 /
Published online: 1 November 2022

Abstract
Most existing deep learning methods for graph matching tasks tend to focus on affinity learning in a feedforward fashion to assist the neural network solver. However, the potential benefits of a direct feedback from the neural network solver to the affinity learning are usually underestimated and overlooked. In this paper, we propose a bidirectional learning method to tackle the above issues. Our method leverages the output of a neural network solver to perform feature fusion on the input of affinity learning. Such direct feedback helps augment the input feature maps of the raw images according to the current solution. A feature fusion procedure is proposed to enhance the raw features with pseudo features that contain deviation information of the current solution from the ground-truth one. As a result, the bidirectional alternation enables the learning component to benefit from the feedback, while keeping the strengths of learning affinity models. According to the results of experiments conducted on five benchmark datasets, our methods outperform the corresponding state-of-the-art feedforward methods.

Keywords Graph matching · Bidirectional learning · Combinatorial optimization · Graph neural networks

Editors: Yu-Feng Li and Prateek Jain.

✉ Shikui Tu
tushikui@sjtu.edu.cn

✉ Lei Xu
leixu@sjtu.edu.cn

Tianxiang Qin
tianxiang_qin@sjtu.edu.cn

[1] Department of Computer Science and Engineering, Shanghai Jiao Tong University, 800 Dongchuan Road, Shanghai 200240, China

1 Introduction

Graph matching (GM) refers to finding node correspondences between two graphs such that the similarity between the matched graphs is maximized. To find node correspondences between two graphs, GM usually leverages node-wise and edge-wise information. When considering node-wise and edge-wise information, GM can be represented in the following Lawler's **q**uadratic **a**ssignment **p**rogramming (QAP) form Lawler (1963):

$$\begin{aligned} J(\mathbf{X}) &= \mathrm{vec}(\mathbf{X})^{\mathbf{T}}\mathbf{K}\mathrm{vec}(\mathbf{X}), \\ \mathbf{X} &\in \{0,1\}^{n\times n}, \mathbf{X}\mathbf{1} = \mathbf{1}, \mathbf{X}^{\mathbf{T}}\mathbf{1} = \mathbf{1} \end{aligned} \tag{1}$$

where $\mathbf{K}$ is the so-called affinity matrix (Leordeanu & Hebert, 2005). Solving the general QAP is known to be NP-hard, and it is difficult to find the global optimal solution for large graphs. GM has been applied in many fields, such as visual tracking, action recognition, robotics, weak-perspective 3-D reconstruction, and so on. For a more comprehensive survey on GM applications, please refer to Vento and Foggia (2013).

Traditional GM methods are mostly developed on predefined, hand-crafted affinity matrix, which is limited to represent real-world data's structure. To tackle this issue, Caetano et al. (2009) introduced free parameters and turned the affinity matrix into a learnable function. Recently, it was developed in Zanfir and Sminchisescu (2018) to learn the affinity in an end-to-end deep learning model, and later deep learning frameworks became popular to build learning models for GM. Typically, a deep graph neural network is constructed to encode the node-wise and edge-wise structural information into the affinity matrix, and the affinity is subsequently used in a GM solver, e.g., Hungarian algorithm. For example, Wang et al. (2019) relaxed GM as a linear assignment problem by employing deep graph embedding models to learn intra-graph and cross-graph affinity functions, while Wang et al. (2021) regarded the affinity matrix as an association graph and transformed GM to a vertex classification problem, solving GM in the Lawler's QAP form directly. However, these methods mainly focus on the design of the feedforward pipeline, and do not notice the potential benefits of a possible feedback from the GM solver to the affinity learning.

In this paper, we take into account not only the assistance from affinity learning to GM solving, but also the feedback benefits from GM solving. Recently, a general bidirectional learning scheme, called IA-DSM, was suggested in Xu (2019), to solve double stochastic matrix (DSM) featured combinatorial tasks like GM. The IA-DSM framework consists of a learning component and an optimization component, and the intermediate discrete solution from the optimization component is fed back into the learning component through a feature enrichment and fusion process. Taking this framework into GM problem solving, Zhao et al. (2021) proposed a bidirectional learning method, IA-GM, to impose the output of the feedforward model into the graph embedding to enhance the affinity learning. Although IA-GM (Zhao et al., 2021) has been demonstrated with promising improvement over the existing GM methods, it still has several limitations. First, IA-GM was implemented by relaxing GM as a linear assignment problem, and whether the bidirectional learning paradigm works for GM in the general Lawler's QAP form is unknown. Second, the performance of IA-GM was mainly evaluated on PASCAL VOC keypoint (Bourdev & Malik, 2009), and the generalization ability to other benchmarks requires further investigations. Third, it deserves more explorations on the effective ways of circulating the information from the optimization component back to the affinity learning.

To address the above issues, we propose a deep bidirectional learning method for solving GM in the Lawler's QAP form, under the IA-DSM framework (Xu, 2019). A new feature fusion technique is employed to enhance the structural patterns using the images which are constructed via the output of the learning component (i.e. current estimation of the solution). In this way, each input sample would bring one similar sample in each bidirectional alternation, helping the model to learn better features. Our main contributions are summarized as follows:

- We propose a deep bidirectional learning method that works for solving GM in the general Lawler's QAP form. Our method employs the DSM given by Sinkhorn algorithm rather than the hard assignment solution after the computation by Hungarian algorithm, and construct image-like input in the feature space for the subsequent feature fusion. In this manner, the constructed features of a node contain information of all possible matching nodes, and enable the learning component to adjust its matching result in the next bidirectional alternation.
- We present a gated feature fusion technique to combine the features of the raw samples in the actual world and their constructed image-like input in the bidirectional alternation. The fused features enforce the learning component to focus more on certain nodes according to the guiding information from the intermediate matching output by Sinkhorn algorithm. In comparisons with Zhao et al. (2021), our bidirectional learning employs a longer feedback path on the features directly.
- We evaluate the proposed method and the existing state-of-the-art ones on four image benchmark datasets and one QAPLIB (Burkard et al., 1997) dataset. Experiments demonstrate that our method is able to improve the matching accuracies consistently and robustly. We also extend IA-GM's bidirectional learning paradigm directly to the QAP form, and empirical analysis indicate that the IA-DSM framework can be flexibly implemented and still has room to improve for solving GM.

2 Related work

2.1 Progresses in learning GM

Traditionally, researchers viewed building the affinity model and finding the solution as two separate steps. They focused on the latter, seeking approximate solution while leaving the affinity model hand-crafted, for example, the elements of the affinity matrix $\mathbf{K}$ are calculated according to the fixed Gaussian kernel with Eucildean distance:

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

where $\mathrm{f}_{ij}, \mathrm{f}_{ab}$ are edges' feature vectors of two graphs respectively. Caetano et al. (2009) first proposed a method to learn the affinity model. Later in 2013, Cho et al. (2013) defined a joint feature map by aligning node-wise and edge-wise similarities into a vectorial form, and introduced weights to all elements of the feature map. However, these predefined methods tend to have limitations in representing real-world data's affinity.

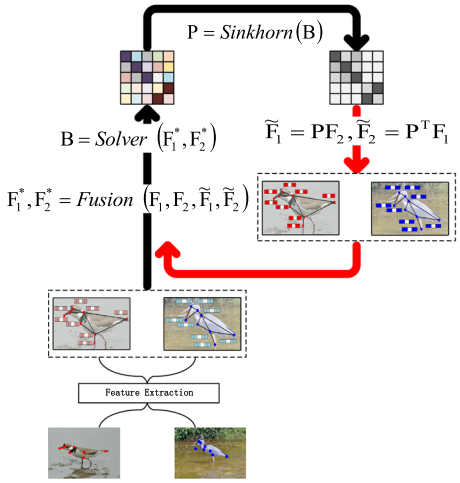
Recently, with the development of deep learning, great progresses have been made in GM. Zhou and De la Torre (2015) proposed a novel closed-form factorization of the

pairwise affinity matrix, making it easier to incorporate global geometric transformation in GM. Meanwhile, there is no need to calculate the affinity matrix explicitly as its structure is decoupled. Then, the end-to-end model proposed by Zanfir and Sminchisescu (2018), which learns an $n^2 \times n^2$ quadratic affinity matrix to guide the GM optimization, presented an efficient way to back-propagate gradients from the loss function to the feature layers. Wang et al. (2019) employed deep graph embedding networks to encode structure affinity into a node-wise affinity matrix so that GM is relaxed as a linear assignment problem, and its devised permutation loss is more powerful than the offset loss used in Zanfir and Sminchisescu (2018). Later, Rolínek et al. (2020) replaced Graph Neural Network (GNN) with SplineCNN (Fey et al., 2018) to process features, pushing the accuracy on PASCAL VOC keypoint up to around 80%. However, these works can not directly deal with the general Lawler's QAP form when individual graph information is unprovided as QAPLIB (Burkard et al., 1997). Therefore, following these work, Wang et al. (2021) proposed Neural Graph Matching (NGM) network by translating the GM task to a vertex classification task to directly solve it in the QAP form, as well as NGM-v2 by using SplineCNN for feature refinement.

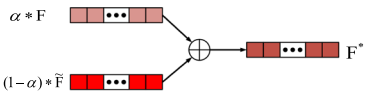
2.2 Bidirectional learning

Bidirectional intelligence was recently reviewed in Xu (2018). In the bidirectional intelligence system, there are two domains defined: A-domain and I-domain. A-domain denotes the **A**ctual-world and I-domain denotes the **I**nner-space. Between the two domains, there are two mappings: A-mapping (from A-domain to I-domain along the inward direction) and I-mapping (from I-domain to A-domain along the outward direction). A general bidirectional learning scheme, called IA-DSM, was first sketched in Xu (2019), under the framework of the system. Featured by an IA-alternation of A-mapping and I-mapping, the IA-DSM is to solve DSM featured combinatorial tasks. For instance, Xu (2019) suggested that traveling salesman problem can be solved in a bidirectional way by employing CNN as A-mapping to obtain a policy and its goodness from the current state, thus guiding the I-mapping for iterative learning. Zhao et al. (2021) provided a GM implementation of the IA-DSM scheme called IA-GM, which follows Wang et al. (2019), relaxing GM as a linear assignment problem. The A-mapping is implemented by an SR-GGNN, and the I-mapping consists of Sinkhorn algorithm and Hungarian algorithm.

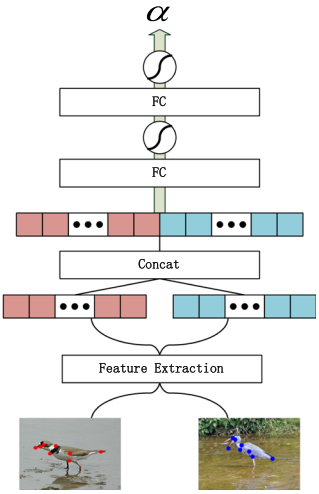
Our work falls in the framework of IA-DSM, but has several differences. Most rencent deep learning GM methods (1) learn affinity matrix from features and (2) find a solution based on the affinity matrix. Zhao et al. (2021) implements the IA-DSM framework by (3) imposing the feedback to graph embedding to enhance the affinity learning, and our method implements the IA-DSM framework by (4) employing the feedback to perform feature fusion. Therefore, the key difference between our work and Zhao et al. (2021) is the bidirectional learning fashion. Unlike Zhao et al. (2021), we lengthen the feedback path to assist the learning component by affecting features directly. Meanwhile, we employ neural network solver of state-of-the-art models and Sinkhorn algorithm as the A-mapping, and we regard pseudo feature generation and feature fusion as the I-mapping. Moreover, we implement IA-DSM to solve GM in the Lawler's QAP form directly while (Zhao et al., 2021) solves GM by relaxing it as a linear assignment problem.



(a) Overview of the proposed method



(b) Feature Fusion



(c) The gating network

Fig. 1 An overview of IA-NGM. **a** The pipeline of IA-NGM for graph matching problem, where *Solver* represents a neural network solver, *Fusion* is implemented by Eq. (10); **b** The *Fusion* process of the pipeline; **c** The gating network for adjusting the fusion weights

3 Methods

3.1 Overview of our method

We propose a bidirectional learning method for solving GM in the general form of Lawler's QAP. Our method is developed under the IA-DSM framework (Xu, 2019) by adopting NGM (Wang et al., 2021) as the network backbone, and thus we call it as IA-NGM. An overview of our method is given in Fig. 1a. IA-NGM consists of a learning component (black arrows) and an optimization component (red arrows) that produces feedback into the learning part. The learning component takes two types of feature maps as input. One type is the primal feature map which is extracted from a real-world image by a CNN-based extractor (e.g., VGG-16 (Simonyan & Zisserman, 2014)). The learning component learns the unary and quadratic affinity across graphs through a neural network solver, and outputs a score matrix **B** whose entries represent the confidences of node correspondences between two graphs. Then Sinkhorn algorithm is performed to get a soft assignment matrix **P**, where its entries indicate how likely the corresponding two nodes are matched to each other. The soft matching solution **P** is used to construct pseudo feature maps for the two input real images, and the pseudo feature maps are circulated back into the learning component via a linear feature fusion module. The fused features are fed into the neural network solver for further learning, and the affinity learning is augmented on the pseudo feature maps which contain the deviation information of the

current matching solution to the ground-truth correspondences. The feedback from the current matching status completes an IA-alternation under the bidirectional intelligence framework (Xu, 2018). The key difference between our IA-alternation and the one in Zhao et al. (2021) is that the affinity learning is more involved in the alternating path, enhancing the representation learning in a more through way.

3.2 Feature extraction and neural network solver

For two input images, we construct two graphs $\mathcal{G}_1, \mathcal{G}_2$ using the annotated key points as graph nodes. We adopt VGG-16 architecture to extract their node features U_1, U_2 from layer *relu4_2*, edge features F_1, F_2 from layer *relu5_1* when using NGM's neural network solver. We use layer *relu4_2* and *relu5_1* to extract init node features and use *relu5_3* to extract global features g when using NGM-v2's solver.

The neural network solver can be viewed as a function as follows:

$$\mathbf{B} = \mathit{Solver}(\mathbf{F}_1^*, \mathbf{F}_2^*), \tag{3}$$

where $\mathbf{B}$ is the score matrix for node-wise correspondence between $\mathcal{G}_1$ and $\mathcal{G}_2$, and $\mathbf{F}_1^*$, $\mathbf{F}_2^*$ represent the features of two graphs, respectively, including node features U_1, U_2, edge features F_1, F_2, and global features g if available. In this paper, we adopt NGM and NGM-v2 (Wang et al., 2021) to learn the score matrix.

3.3 Pseudo feature generation

Following (Adams & Zemel, 2011), Sinkhorn algorithm is computed as follows:

$$\mathbf{S}_{k+1} = \begin{cases} \mathbf{S}_k[\mathbf{1}_n^{\mathbf{T}}\mathbf{S}_k]^{-1}, & k \bmod 2 = 0 \\ [\mathbf{S}_k\mathbf{1}_n^{\mathbf{T}}]^{-1}\mathbf{S}_k, & k \bmod 2 = 1 \end{cases} \tag{4}$$

where $k = 0, 1, 2, \ldots$ denotes the iteration serial number. Then, the soft assignment matrix $\mathbf{P}$ is obtained by Sinkhorn procedure taking the score matrix $\mathbf{B}$ as input,

$$\mathbf{P} = \mathit{Sinkhorn}(\mathbf{B}), \quad \text{i.e., } \mathbf{S}_0 = \mathbf{B}. \tag{5}$$

Then, the hard assignment matrix $\mathbf{X} \in \{0, 1\}^{n \times n}$ is calculated by the Hungarian algorithm:

$$\mathbf{X} = \mathit{Hungarian}(\mathbf{P}). \tag{6}$$

If the matching solution $\mathbf{X}$ is close to optimum, the permuted features of one graph will be similar to those of the other graph. Specifically, define the pseudo node features as:

$$\widetilde{U}_1 = \mathbf{X}U_2, \quad \widetilde{U}_2 = \mathbf{X}^{\mathbf{T}}U_1, \tag{7}$$

where $U_1, U_2 \in \mathbb{R}^{n \times d}$ are primal node features extracted directly from the input images. Generally, we calculate the pseudo overall features as follows:

$$\widetilde{\mathbf{F}}_1 = \mathbf{X}\mathbf{F}_2, \quad \widetilde{\mathbf{F}}_2 = \mathbf{X}^{\mathbf{T}}\mathbf{F}_1, \tag{8}$$

where $\mathbf{F}_1, \mathbf{F}_2$ are primal overall features extracted from the input images. It is noted that the pseudo features depend on the accuracy of the matching solution $\mathbf{X}$. If $\mathbf{X}$ is optimal, the discrepancy between $\widetilde{\mathbf{F}}_i$ and $\mathbf{F}_i$ is minimized. However, at the beginning of the training

process, the intermediate solution $\mathbf{X}$ is not the optimal one, including the correctly matched part and the error part. The matched part does not affect the affinity learning, but the error part would change the affinity.

Since $\mathbf{X}$ by Eq. (6) is a binary permutation matrix, the pseudo feature generation by Eq. (8) is not flexible enough to continuously capture the learning status of the model. Moreover, it does not take into account the uncertainty of the learning process which is large at the beginning. Therefore, we further replace $\mathbf{X}$ in Eq. (8) with the soft assignment matrix $\mathbf{P}$, and get

$$\widetilde{\mathbf{F}}_1 = \mathbf{P}\mathbf{F}_2; \quad \widetilde{\mathbf{F}}_2 = \mathbf{P}^{\mathbf{T}}\mathbf{F}_1. \tag{9}$$

In this way, the constructed pseudo features are more sensitive to deviation of the learning output from the optimal solution, and they are helpful and more effective in guiding the learning process towards the optimal solution.

3.4 Gated feature fusion

We present a linear fusion scheme to integrate the pseudo features back into the learning component, as illustrated in Fig. 1b:

$$\mathbf{F}_1^* = \alpha\mathbf{F}_1 + (1-\alpha)\widetilde{\mathbf{F}}_1, \quad \mathbf{F}_2^* = \alpha\mathbf{F}_2 + (1-\alpha)\widetilde{\mathbf{F}}_2, \tag{10}$$

where we have constrained the sum of the combining coefficients α and $1-\alpha$ to be one. The obtained $\mathbf{F}_1^*, \mathbf{F}_2^*$ are the features augmented with deviation information of the current GM solution from the optimum, and they are fed into Eq. (3) to improve the affinity learning.

In practice, the coefficients can be predefined according to the empirical experience. However, prefixed coefficients may not adapt to the learning dynamics well because feature extractors like SplineCNN may change the distribution of the primal features as learning proceeds. We propose a two fully-connected (FC) layer network with feature channels $l_1 = 1024, l_2 = 32$, which is shown in Fig. 1c, to learn the weight α according to the current state of the input. It is worth noticing that the *Feature Extraction* in Fig. 1a and c is same, but we use all features extracted in Fig. 1a and only use global features in Fig. 1c. The global features g_1, g_2 are computed as the *relu5_3* layer of VGG-16 for the two input graphs $\mathcal{G}_1, \mathcal{G}_2$, as described in Sect. 3.2. Mathematically, the gating network is computed as follows:

$$\alpha = \tanh(\mathrm{FC}(\tanh(\mathrm{FC}(\mathit{concat}(g_1, g_2))))), \tag{11}$$

where *concat* denotes the concatenation operator.

3.5 IA-alternation

With the gated fusion between the primal features and the pseudo ones by Eq. (10), it completes the circle of IA-alternation, i.e.,

$$\mathbf{P}^{(k)} = \mathit{Sinkhorn}(\mathit{Solver}(\mathbf{F}_1^{*(k)}, \mathbf{F}_2^{*(k)})), \tag{12}$$

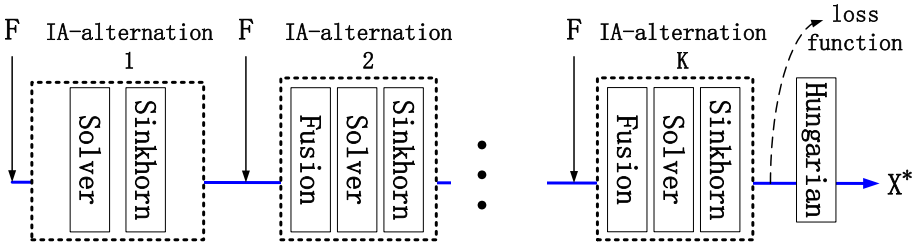


Fig. 2 Unfolded structure of the proposed network during training

$$\mathbf{F}_i^{*(k+1)} = \mathit{Fusion}(\mathit{Construct}(\mathbf{P}^{(k)}, \mathbf{F}_i^{*(k)})), \quad i \in \{1, 2\} \tag{13}$$

where $k = 1, 2, \cdots, K$ represents the serial number of the IA-alternations and *Construct* is implemented by Eq. (9). At the beginning, there is no pseudo features, and we set $\mathbf{F}_1^{*(1)} = \mathbf{F}_1, \mathbf{F}_2^{*(1)} = \mathbf{F}_2$ or equivalently setting $\alpha = 1$ in Eq. (10). The soft assignment matrix $\mathbf{P}^{(k)}$ encodes the cross-graph affinity via the neural network solver, and the gated feature fusion guides the learning in the next alternation to pay more attention to the matching pairs that have large deviation from the optimum in the previous alternation. As a consequence, the IA-alternation benefits the GM solving model from both directions. After the last IA-alternation, the predicted assignment matrix is calculated by Eq. (6) as:

$$\mathbf{X}^* = \mathit{Hungarian}(\mathbf{P}^{(K)}). \tag{14}$$

3.6 Loss function

We leverage the ground-truth correspondences to supervise the learning process. We adopt the permutation loss (Wang et al., 2019), which has been proved effective for graph matching tasks, as the loss function:

$$\mathcal{L} = -\frac{1}{n} \sum_{i,j} \left[x_{ij}^{gt} \ln p_{ij}^{(K)} + (x_{ij}^{gt}) \ln (1 - p_{ij}^{(K)}) \right] \tag{15}$$

where $\mathbf{X}^{gt} = [x_{ij}^{gt}], x_{ij}^{gt} \in \{0, 1\}$ represents the ground-truth assignment matrix, and $\mathbf{P}^{(K)} = [p_{ij}^{(K)}]$ is the soft assignment matrix of the last IA-alternation by Eqs. (12) and (13). We illustrate the computational procedure in Fig. 2. It is noted that, in the end-to-end training, the Hungarian algorithm is only used in the end to get the output matrix, while Sinkhorn is used in each feedback loop to generate the intermediate solution matrix. The loss function by Eq. (15) is applied on the soft assignment matrix output by the K-th Sinkhorn block, before the Hungarian algorithm is activated in the end. In testing, the model follows the blue arrow in Fig. 2 to get $\mathbf{X}^*$, and there is no back propagation involved.

4 Experiments

We evaluate our models on five benchmark datasets. Specifically, in line with Wang et al. (2021), we use PASCAL VOC keypoint and Willow ObjectClass (Cho et al., 2013) as benchmark datasets, to compare with previous closely-related GM methods: (1) *GMN* (Zanfir & Sminchisescu, 2018) learns affinity functions, (2) *PIA-GM* (Wang et al., 2019) learns intra-graph affinity, (3) *PCA-GM* (Wang et al., 2019) learns cross-graph affinity, (4) *IA-GM* (Zhao et al., 2021) implements the IA-DSM framework by relaxing GM to linear, (5) *BBGM* employs SplineCNN (Fey et al., 2018) in GM and (6) *NGM & NGM-v2* (Wang et al., 2021)) solve GM in QAP form directly.

For further exploration on the benefits of different bidirectional learning fashion, we also include IMC-PT-SparseGM[1] and CUB2011 (Wah et al., 2011) as benchmark datasets, to compare with NGM and NGM-v2 (Wang et al., 2021), and IA-GM (Zhao et al., 2021). Our models are named as IA-NGM and IA-NGM-v2, which correspond to the versions of employing the neural network solvers of NGM and NGM-v2 respectively. We set the value of α in feature fusion by Eq. (10) to be 0.6 for IA-NGM according to our empirical experience, and leave it to be learned by the gating learning network in IA-NGM-v2. We set the number of IA-alternation to be 3 for IA-NGM and 2 for IA-NGM-v2, because a big number of IA-alternations would possibly lead to over-smoothing. For fair comparisons with IA-GM, we extend it to be IA-GM-v1 and IA-GM-v2 by modifying the network backbone from the weights-shared residual gated graph neural network (SR-GGNN) to be NGM and NGM-v2 respectively.

In line with Wang et al. (2021), the neural network solvers of IA-NGM and IA-NGM-v2 both use a three-layer GNN with feature channels $l_1 = l_2 = l_3 = 16$ to perform association graph embedding. The maximum iteration number is set to 20 for Sinkhorn algorithm. The initial learning rate of the neural network solver in IA-NGM/IA-NGM-v2 and the gating network for α in Eq. (10) starts at 10^{-3} and decays by 10 every 10,000 steps. The learning rate of the feature extractor is set to 2×10^{-6} in IA-NGM-v2. All experiments are conducted on a Linux workstation with Titan XP GPU.

4.1 Results on PASCAL VOC keypoint

The PASCAL VOC keypoint dataset, containing 20 semantic classes with annotated examples, is challenging as the images differ in size and the number of keypoints varies from 6 to 23. We count the keypoints of samples for each class in Fig. 3. It is observed that some classes contain samples of more than 10 keypoints while others may have only a few.

The matching accuracies on PASCAL VOC keypoint are reported in Table 1. Our bidirectional methods all outperform the corresponding feedforward baselines, indicating that the feedback of the pseudo features is beneficial to the affinity learning. Specifically, IA-NGM improves the mean accuracy by 2.5% with respect to NGM, while IA-NGM-v2 improves the performance by 0.8% with respect to NGM-v2. Meanwhile, IA-NGM and IA-NGM-v2 outperforms IA-GM-v1 and IA-GM-v2, respectively, indicating that the affinity fusion by Zhao et al. (2021) is not so effective as the proposed feature fusion by Eq. (10). The NGM-v2 based methods are superior to the NGM-based ones, with about 15% increment contributed mainly by using SplineCNN as the feature extractor. The increment 2.5% by IA-NGM is more than 0.8% by IA-NGM-v2, probably because it becomes more difficult

[1] https://github.com/Thinklab-SJTU/ThinkMatch#available-datasets

Fig. 3 Keypoint number distribution of PASCAL VOC dataset. Each of 20 subfigures corresponds to a semantic class. The name of each class is marked at top-right of the subfigure. The x-axis denotes the number of keypoints in a graph, and the y-axis denotes the number of samples containing corresponding number of keypoints

to improve the performance from a very high level. Looking into the detailed accuracies for each class in Table 1, we observe that our bidirectional methods can make the neural network solver learn better on its weak classes while keeping its comparable results on other classes. For example, on the class “chair", IA-NGM-v2 exceeds NGM-v2 by 11.7%; on the class “table", IA-NGM achieves a 29.9% improvement over NGM.

To evaluate the performance on each class, we train and test NGM, IA-NGM, NGM-v2 and IA-NGM-v2 on individual classes, and the results are shown in Table 2. The results are consistent with Table 1. IA-NGM-v2 exceeds NGM-v2 on the class “chair" and “sofa" by over 10%, while NGM-v2’s in-class accuracy is lower than its cross-class accuracy. Some classes can have similar features with other classes so that NGM-v2’s learning is enhanced when trained on all classes, while IA-NGM-v2 already fused the similarity into features so that it has similar in-class and cross-class performances.

Moreover, IA-GM/IA-GM-v2 are comparable to IA-NGM/IA-NGM-v2, respectively, which confirms the effectiveness of the feedback and fusion scheme in Zhao et al. (2021). For the performance on 20 classes, the accuracy gaps between the two methods are mostly small except for the class “table" where IA-NGM-v2 exceeds IA-GM-v2 by 20.3%. We also notice that the accuracies on the class “aero" and “person" remain relatively low, because the images of the two classes have a wide range of keypoints, making it very difficult to learn.

Table 1 Matching accuracy (%) on Pascal VOC keypoint

Method	Aero	Bike	Bird	Boat	Bottle	Bus	Car	Cat	Chair	Cow	Table	Dog	Horse	Mbkie	Person	Plant	Sheep	Sofa	Train	Tv	Mean
GMN	41.6	59.6	60.3	48.0	79.2	70.2	67.4	64.9	39.2	61.3	66.9	59.8	61.1	59.8	37.2	78.2	68.0	49.9	84.2	**91.4**	62.4
PIA-GM	41.5	55.8	60.9	51.9	75.0	75.8	59.6	65.2	33.3	65.9	62.8	62.7	67.7	62.1	42.9	80.2	64.3	59.5	82.7	90.1	63.0
PCA-GM	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
IA-GM	51.6	66.6	63.1	56.0	**80.1**	74.1	68.0	**72.1**	38.9	65.7	73.6	65.1	67.1	**66.1**	44.9	**82.0**	68.1	59.4	78.5	90.9	**66.6**
NGM	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
IA-GM-v1(ours)	**53.6**	59.5	65.2	**56.6**	77.8	76.1	73.5	70.7	42.7	**67.0**	46.5	65.7	65.4	62.9	**47.0**	76.4	68.2	52.4	79.5	88.5	64.8
IA-NGM (ours)	50.5	**67.3**	65.1	53.9	79.8	**78.8**	70.7	70.2	**45.5**	66.6	**74.7**	**66.3**	63.9	63.5	45.5	81.5	**69.7**	46.6	81.0	91.1	**66.6**
BBGM	61.9	71.1	79.7	79.0	87.4	94.0	89.5	80.2	56.8	79.1	64.6	78.9	76.2	75.1	65.2	98.2	77.3	77.0	94.9	**93.9**	79.0
NGM-v2	61.8	71.2	77.6	78.8	87.3	93.6	87.7	79.8	55.4	77.8	89.5	78.8	**80.1**	**79.2**	62.6	97.7	77.7	75.7	96.7	93.2	80.1
IA-GM-v2 (ours)	**62.8**	**73.9**	**80.1**	**79.8**	88.6	**94.9**	**90.0**	**81.9**	63.1	**79.5**	72.1	**80.0**	79.1	78.7	65.7	**98.8**	**78.6**	76.9	**98.4**	93.1	80.8
IA-NGM-v2 (ours)	61.5	73.8	74.0	79.4	**89.1**	94.6	89.7	77.5	**67.1**	77.3	**92.4**	76.8	77.1	77.4	**65.8**	98.5	77.5	**79.5**	96.6	92.3	**80.9**

Best results are in bold

Table 2 In-class performances on certain classes of PASCAL VOC keypoint

Class	NGM	IA-NGM	NGM-v2	IA-NGM-v2
Bottle	74.7	85.8	88.5	**93.3**
Bus	79.4	79.6	92.4	**92.7**
Car	79.4	82.1	**91.7**	89.9
Cat	72.1	65.8	**75.4**	73.2
Chair	41.7	47.4	55.8	**73.1**
Dog	58.8	55.8	**73.5**	72.8
Sofa	72.5	67.9	63.1	**75.9**
Train	89.9	90.9	95.7	**97.4**
Mean	71.0	72.2	79.5	**83.5**

Best results are in bold

Table 3 Matching accuracy (%) on Willow ObjectClass

Method	Car	Duck	Face	m-bike	w-bottle	Mean
GMN	67.9	76.7	99.8	69.2	83.1	79.3
PCA-GM	87.6	83.6	**100.0**	77.6	88.4	87.4
IA-GM	94.1	89.9	**100.0**	83.1	94.1	92.3
NGM	84.2	77.6	99.4	76.8	88.3	85.3
IA-GM-v1 (ours)	95.1	90.3	**100.0**	94.7	94.5	94.9
IA-NGM (ours)	94.4	90.8	**100.0**	95.4	95.7	95.3
BBGM	96.8	89.9	**100.0**	99.8	99.4	97.2
NGM-v2	**97.4**	93.4	**100.0**	98.6	98.3	97.5
IA-GM-v2 (ours)	96.4	93.7	**100.0**	99.8	99.0	97.8
IA-NGM-v2 (ours)	96.4	**96.3**	**100.0**	**100.0**	**100.0**	**98.5**

Best results are in bold

4.2 Results on willow ObjectClass

Willow ObjectClass, containing 5 categories and all instances in same class share 10 distinct image keypoints, is easier than PASCAL VOC keypoint for the GM task. The results are given in Table 3. We see that our bidirectional methods outperform their corresponding feedforward baselines, which is consistent with the results in Table 1. In particular, the mean accuracy of IA-NGM is 10% higher than NGM. Although the mean accuracy of NGM-v2 is already very high at 97.5%, IA-NGM-v2 still improves by a 1% increment to 98.5%. IA-NGM and IA-NGM-v2 are again separately better than IA-GM-v1 and IA-GM-v2, consistently indicating that our gated feature fusion scheme is more effective than the fusion scheme in Zhao et al. (2021).

4.3 Results on IMC-PT-SparseGM

IMC-PT-SparseGM is developed based on Image Matching Challenge dataset (Jin et al., 2021), which contains 16 classes of images of popular landmarks. We use 3 classes for testing and the others for training. Due to the limit of memory, we set the batch size to 2. The results

Table 4 Matching accuracy (%) on IMC-PT-SparseGM

Method	Reichstag	sacre_coeur	st_peters_square	Mean
PCA-GM	73.2	42.9	44.5	53.5
IA-GM	83.5	63.9	78.2	75.2
NGM	97.8	76.1	87.4	87.1
IA-GM-v1 (ours)	97.9	73.3	84.8	85.3
IA-NGM (ours)	98.2	77.2	87.1	87.5
NGM-v2	99.2	78.4	87.6	88.4
IA-GM-v2 (ours)	**99.7**	79.4	87.9	89.0
IA-NGM-v2 (ours)	99.3	**80.9**	**89.1**	**89.8**

Best results are in bold

Table 5 Matching accuracy (%) on CUB2011

Method	PCA-GM	IA-GM	NGM	IA-GM-v1	IA-NGM	NGM-v2	IA-GM-v2	IA-NGM-v2
Accuracy	93.8	95.2	94.0	92.8	94.7	96.3	97.3	**97.4**

Best results are in bold

in Table 4 again confirm the advantages of our bidirectional learning methods over the feed-forward baselines. Both IA-NGM-v2 and IA-GM-v2 outperform NGM-v2, while IA-NGM-v2 achieves the highest mean matching accuracy.

4.4 Results on CUB2011

The CUB2011 dataset by Wah et al. (2011) contains 11788 images of 200 subcategories belonging to birds, 5994 for training and 5794 for testing. Detailed annotations are provided for each image, including its subcategory label, 15 part locations, 312 binary attributes and 1 bounding box. Unlike other datasets we used, all images of CUB2011 are regarded as one class only. The matching accuracies are reported in Table 5. We can find that the bidirectional method IA-NGM-v2 improves the accuracy by about 1.1% over NGM-v2, while IA-NGM is slightly better than NGM.

To summarize the results on the four image datasets, i.e., PASCAL VOC keypoint, Willow ObjectClass, IMC-PT-SparseGM, and CUB2011, our bidirectional learning methods IA-NGM and IA-NGM-v2 are both able to improve the matching performance with respect to their baselines NGM or NGM-v2. Generally, the NGM/NGM-v2 based formulations of GM in the form of Lawler's QAP is relatively more powerful than the relaxed linear assignment version and IA-NGM-v2 is the best for all four datasets.

4.5 Results on QAPLIB

We also evaluate our bidirectional learning method for pure QAP tasks on QAPLIB benchmark,[2] in comparisions with RRWM (Cho et al., 2010), SM (Leordeanu & Hebert, 2005),

[2] https://www.opt.math.tugraz.at/qaplib/

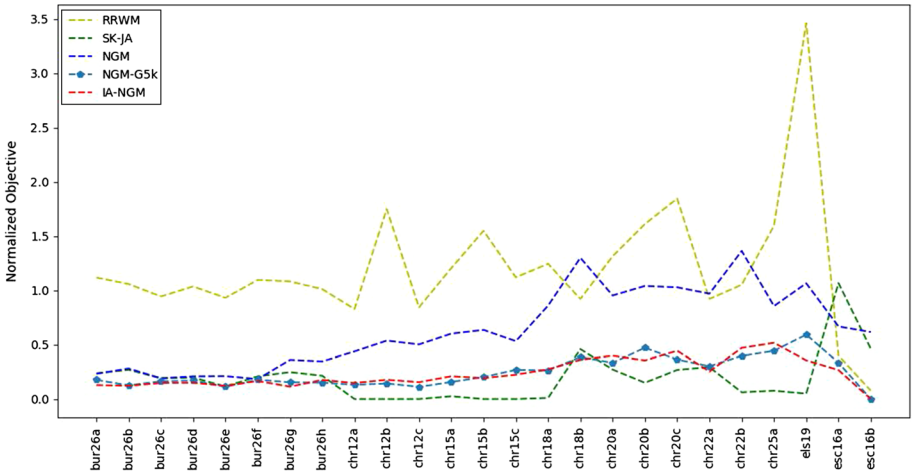


Fig. 4 Normalized objective score (lower is better) of different methods on selected QAP instances due to the space limit. The instances are sorted by alphabet order

SK-JA (Kushinsky et al., 2019) and NGM/NGM-G5k (Wang et al., 2021). As the affinity matrix of pure QAP instances are constructed using Kronecker product rather than features of images, our feature fusion method is not applicable. Therefore, we just try one sharping method like Eq. (9), regarding the input matrix as feature map. The final solution is obtained using the original input after three IA-alternations during training. Due to the limit of memory, we can only run experiments on 100 instances, and the settings are different from NGM-G5k's suggested settings. We train one model for each category, and use the normalized objective score (lower is better) as the evaluation metric, i.e.,

$$norm\_score = \frac{solved\_score - upper\_bnd}{SM\_score - upper\_bnd}, \tag{16}$$

where the upper bound (primal bound) is provided by the up-to-date online benchmark, and the baseline solver is given by SM (Leordeanu & Hebert, 2005).

In spite of these difficulties, our method still gets lower objective scores on 45 instances than NGM-G5k according to Table 10 in the Appendix 1. Examples of the results are visualized in Fig. 4. It is noted that the result on kra32 is abnormal. We have checked the data, and find that the data might be problematic, because the ground truth objective score cannot be obtained by the ground truth solution and two input matrices. Meanwhile, although the data of ste36a has the same problem, its result is normal. We also found the orders of two input matrice of the kronecker product of several instances are reversed which seems have little influence on the result.

4.6 Study on variants of our method

In this section, we investigate the performances of different variants of our bidirectional method, to explain why we choose the linear mixture as our feature fusion technique for NGM's neural network solver and why we choose the soft assignment matrix $\mathbf{P}$ for feature fusion rather than the hard assignment matrix $\mathbf{X}$.

Table 6 Matching accuracy (%) on PASCAL VOC keypoint when choosing different numbers of IA-alternation

Number of IA-alternation	$K = 0$ (baseline)	$K = 1$	$K = 2$	$K = 3$
Mean accuracy	64.46	65.62	66.61	66.02

Table 7 Matching accuracy (%) on PASCAL VOC keypoint

Method	Number of IA-alternation	α	Mean
NGM (baseline)	$K = 1$	–	63.18
Concatenation	$K = 2$	–	63.87
Linear mixture	$K = 2$	0.6	65.20
Linear mixture	$K = 3$	0.6	65.70
Linear mixture	$K = 4$	0.6	65.03
Linear mixture	$K = 5$	0.6	65.32
Linear mixture	$K = 6$	0.6	**65.97**
Linear mixture	$K = 7$	0.6	25.13
Linear &concat	$K = 2$	0.6	64.61

Best results are in bold

Table 6 provides the results of experiments on the choice of the number of IA-alternations of IA-NGM. To avoid the over-smoothing problem, we only vary the number of IA-alternations from 2 to 4. We observe that the model of 3 IA-alternations achieves the highest performances.

4.6.1 Exploration on IA-alternation and feature fusion technique

We have tried several ways of feature fusion, including the linear mixture, concatenation, and the combination of these two approaches. The mean matching accuracies are reported in Table 7, which also includes the results when the number of IA-alternation varies. We observe that most feature fusion models improve the performance except for the case of 7 IA-alternations and linear mixture, which leads to the over-smoothing problem. The input feature is fused in every IA-alternation, and after too many times of IA-alternation, the features of nodes and edges tend to be close to each other due to the GNN used in the neural network solver, where all the nodes share the same transition function and the same output function (Uwents et al., 2011). From the second and the third row, we can see that the linear mixture outperforms the concatenation by 1.33%, and from the third and the last row, we can find that the linear mixture also outperforms the combination method. Therefore, we choose the linear mixture for feature fusion in IA-NGM, and also in IA-NGM-v2 for consistency.

Meanwhile, the choice of α in Eq. (10) is important, as it affects the influence of pseudo features on affinity learning. Inappropriate value may cause the fused features to deviate from the original correct one. As the learning proceeds, the uncertainty conveyed in pseudo features gradually changes. Therefore, a fixed α may limit the performance of the model. As in Table 8, the case of determining α via a learnable gating network can adapt the learning dynamics well and outperforms the case of fixing α at a certain constant.

Table 8 Matching accuracy (%) on four image datasets when choosing different fusion weights while the number of IA-alternations is $K = 2$

α in Eq. (10)	PASCAL VOC	Willow	IMC-PT-SparseGM	CUB2011
1	80.2	97.5	88.4	96.3
0.5	78.4	96.5	84.8	97.0
0.75	79.5	**98.5**	86.4	96.8
0.8	75.7	97.6	86.1	97.3
Gating learning network	**80.9**	**98.5**	**89.8**	**97.4**

Best results are in bold. Notice that $\alpha = 1$ acutally degenerates our algorithm to NGM-v2

Table 9 Average accuracy on PASCAL VOC of IA-NGM using **P** or **X** for feature fusion

epoch	1	2	3	4	5	6	7	8	9	10
X	33.6	52.6	51.0	53.1	55.1	56.1	56.8	59.4	59.4	55.9
P	62.3	64.5	63.9	65.3	64.5	64.5	64.9	64.8	65.6	66.3

4.6.2 Choice of feedback matrix

Table 9 shows the average testing accuracies of the first ten epochs on PASCAL VOC keypoint of IA-NGM using the soft assignment matrix **P** or the hard assignment matrix **X** for feature fusion. It is found that IA-NGM using **P** learns faster and better than IA-NGM using **X**. As a result, we choose **P** for feature fusion in IA-NGM and also in IA-NGM-v2.

5 Conclusion

We have proposed a deep bidirectional learning method named IA-NGM for graph matching problem. Our method is featured by an IA-alternation of a learning component for affinity learning and an optimization component that produces feedback via feature fusion into the affinity learning. The feedback is leveraged to construct the feature maps by a gated feature fusion of not only raw features directly extracted from images, but also pseudo features that convey the deviation information of the current optimization output from the ground-truth optimum. We devise a gating network to instruct the feature fusion based on the global features of two graphs, and the gating network adapts to the learning dynamics well for improved performance. Experimental results on benchmark datasets indicate that our method can fit two different neural network solvers appropriately, and demonstrate that the bidirectional learning scheme is effective to further improve the matching accuracies. Last but not least, we also find that our bidirectional learning fashion can be applied effectively on pure QAP tasks of QAPLIB benchmark, and there is still room for further improvement.

Appendix 1

Detailed results on QAPLIB

See Table 10.

Table 10 Per-instance results by objective score (lower is better) on QAPLIB

Instance	Score									
	Upper	SM	RRWM	SK-JA	NGM	NGM-G5	NGM-G50	NGM-G500	NGM-G5k	IA-NGM
bur26a	5426670	6533340	6663181	5688893	5684628	5828287	5650343	5650343	5621774	**5568979**
bur26b	3817852	4690772	4741283	4053243	4063246	4110185	4066986	3943769	3927943	**3924336**
bur26c	5426795	6537412	6474996	5639665	5638641	5892078	5728542	5671528	5608065	**5591449**
bur26d	3821225	4649645	4678974	3985052	3994147	4137709	4077089	3965083	3962317	**3945937**
bur26e	5386879	6711029	6619788	5539241	5666202	5856035	5655088	5638619	**5536142**	5548485
bur26f	3782044	4723824	4814298	3979071	3954977	4082092	4050459	3949064	3949711	**3939088**
bur26g	10117172	12168111	12336830	10624776	10855165	10694274	10694272	10477104	10433439	**10351077**
bur26h	7098658	8753694	8772077	7453329	7670546	7493907	7493907	7417478	**7348866**	7389910
chr12a	9552	50732	43624	**9552**	27556	32670	29780	20138	14940	15620
chr12b	9742	46386	73860	**9742**	29396	21528	21528	24734	14984	16244
chr12c	11156	57404	50130	**11156**	34344	31602	31560	22174	16346	18332
chr15a	9896	77094	90870	**11616**	50272	51746	39414	31478	20442	23982
chr15b	7990	77430	115556	**7990**	52082	52066	31936	28546	22048	21332
chr15c	9504	64198	70738	**9504**	38568	48318	34932	26548	24190	21758
chr18a	11098	94806	115328	**11948**	83026	53612	50814	41254	33124	33944
chr18b	1534	4054	3852	2690	4810	4330	3430	3056	2504	**2442**
chr20a	2192	11154	13970	**4624**	10728	9516	8382	6626	5178	5758
chr20b	2298	9664	14168	**3400**	9962	8522	8196	5426	5766	4910
chr20c	14142	112406	195572	**40464**	115128	103040	66182	66382	49770	57716
chr22a	6156	16732	15892	9258	16410	15394	10778	10160	9348	**8832**
chr22b	6194	13294	13658	**6634**	15876	11882	10330	9156	9006	9528
chr25a	3796	21526	32060	**5152**	18950	16704	13758	13162	11648	12948
els19	17212548	33807116	74662642	**18041490**	34880280	53830864	31247564	28600336	27029748	23111772
esc16a	68	98	80	100	88	84	86	82	78	**76**
esc16b	**292**	318	294	304	308	310	306	296	**292**	**292**

Table 10 (continued)

Instance	Score									
	Upper	SM	RRWM	SK-JA	NGM	NGM-G5	NGM-G50	NGM-G500	NGM-G5k	IA-NGM
esc16c	160	276	204	266	184	216	210	186	174	**170**
esc16d	16	48	44	58	40	32	40	26	**20**	24
esc16e	28	52	50	44	48	46	42	38	**32**	**32**
esc16g	26	44	52	52	50	48	48	38	**32**	34
esc16h	996	1292	**1002**	1282	1036	1448	1146	1076	1004	1010
esc16i	14	54	28	36	26	50	32	26	**18**	**18**
esc16j	8	22	18	18	16	28	16	14	**8**	10
esc32e	2	68	6	46	42	32	22	**2**	**2**	**2**
esc32g	6	36	**10**	46	28	30	14	14	**10**	**10**
had12	1652	1894	2090	–	1790	1808	1792	1722	**1700**	1706
had14	2724	3310	3494	2916	2922	3098	2940	2928	**2866**	2904
had16	3720	4390	4646	3978	4150	4070	4064	3960	3902	**3814**
had18	5358	6172	6540	5736	5780	5950	5670	5662	**5558**	5626
had20	6922	8154	8550	7464	7334	7592	7430	7362	7300	**7252**
kra30a	88900	148690	136830	125290	**114410**	128920	122660	122780	**114410**	117550
kra30b	91420	150760	141550	126980	**118130**	130940	128880	124590	**118130**	119700
kra32	88700	145310	148730	128120	121340	133290	129080	125530	120930	*23668*
lipa20a	3683	3956	3940	**3683**	3929	3904	3891	3864	3853	3849
lipa20b	27076	36502	38236	**27076**	33907	34970	33902	33815	33125	33516
lipa30b	151426	198434	201775	**151426**	192356	191633	191034	189057	187607	188505
lipa40a	31538	32736	32686	**31538**	32666	32658	32521	32504	32454	32472
lipa40b	476581	628272	647295	**476581**	616656	620456	610699	604766	601848	603248
lipa50b	1210244	1589128	1591109	**1210244**	1543264	1552533	1530231	1531102	1523856	1522173
lipa60a	107218	109861	110468	**108456**	110094	110068	109884	109748	109595	109637

Table 10 (continued)

Instance	Score									
	Upper	SM	RRWM	SK-JA	NGM	NGM-G5	NGM-G50	NGM-G500	NGM-G5k	IA-NGM
lipa60b	2520135	3303961	3300291	**2520135**	3269504	3246252	3231895	3219131	3208501	3204439
lipa70a	169755	173649	173569	**172504**	173862	173577	173419	173218	173220	173114
lipa70b	4603200	6055613	6063182	**4603200**	5978316	5986328	5889358	5908729	5890161	5883777
lipa80a	253195	258345	258608	**257395**	258402	258108	257997	257767	257663	257752
lipa80b	7763962	10231797	10223697	**7763962**	10173155	10094877	10059724	10007233	9983040	10002951
lipa90a	360630	367384	367370	366649	367193	367085	366926	366795	**366508**	366540
lipa90b	12490441	16291267	16514577	**12490441**	16194745	16232027	16176035	16143880	16076956	16116343
nug12	578	886	1038	682	720	772	716	676	634	**604**
nug14	1014	1450	1720	–	1210	1308	1212	1156	1156	**1154**
nug15	1150	1668	2004	1448	1482	1500	**1312**	1360	1318	1314
nug16a	1610	2224	2626	1940	**1836**	2140	1882	1888	**1836**	1842
nug16b	1240	1862	2192	1492	1580	1696	1610	1468	**1396**	1466
nug17	1732	2452	2934	2010	2004	2256	2110	2062	**1980**	2010
nug18	1930	2688	3188	**2192**	2312	2384	2340	2322	2242	2248
nug20	2570	3450	4174	3254	**2936**	3260	3174	3068	**2936**	2988
nug21	2438	3702	4228	3064	2916	3346	3132	2954	2916	**2892**
nug22	3596	5896	6382	**3988**	4616	5092	4616	4160	4298	4300
nug24	3488	4928	5720	4424	**4234**	4568	4514	4278	**4234**	4264
nug25	3744	5332	5712	**4302**	4420	4822	4788	4510	4420	4442
nug27	5234	7802	8626	6244	6332	6860	6546	6276	**6208**	**6208**
nug28	5166	7418	8324	6298	**6128**	6764	6332	6424	**6128**	6288
nug30	6124	8956	10034	**7242**	7608	7666	7780	7530	7294	7370
rou12	235528	325404	377168	276446	321082	301728	275876	**252156**	264898	258836
rou15	354210	489350	546526	**390810**	469592	453260	430000	413294	403872	399952
rou20	725522	950018	1010554	823298	897348	876282	867682	845494	**817776**	832346

Table 10 (continued)

Instance	Score									
	Upper	SM	RRWM	SK-JA	NGM	NGM-G5	NGM-G50	NGM-G500	NGM-G5k	IA-NGM
scr12	31410	71392	95134	45334	44400	38228	42014	40908	**36292**	37248
scr15	51140	104308	101714	74632	81344	77376	75746	75224	68768	**66018**
scr20	110030	263058	350528	171260	182882	212432	175534	171666	**154636**	163832
sko81	90998	112838	118372	105246	107588	108404	105776	105418	104710	**104678**
sko90	115534	140840	148784	133818	137402	136624	134472	133056	132942	**132334**
ste36a	9526	30030	33294	17938	16648	20452	19506	18320	16768	**16480**
ste36b	15852	176526	193046	47616	**43248**	63332	58752	53426	**43248**	45614
ste36c	8239110	24530792	28908062	14212212	**12988352**	15819606	15824300	14055568	**12988352**	13891604
tai12a	224416	318032	392004	**245012**	259014	284106	280362	272610	255158	263332
tai12b	39464925	96190153	124497790	81727424	65138752	53817560	53707944	50142456	**47252044**	47375184
tai15a	388214	514304	571952	**171272**	467812	488870	450056	450586	436968	433954
tai15b	51765268	702925159	702292926	**52585356**	495479040	234006816	53608232	52900096	52871608	52786588
tai17a	491812	669712	738566	598716	630644	608616	594994	574052	**544754**	555042
tai20a	703482	976236	1012228	849082	896518	884076	838930	831358	806382	**802270**
tai20b	122455319	394836310	602903767	220470588	237607744	275857568	160806544	159867648	**140704160**	148618400
tai25b	344355646	764920942	1253946482	798113083	730775168	773628544	638959616	600683456	**518647040**	548680704
tai30b	637117113	1008164383	1766978330	1114514832	1359600384	1193240320	1082828032	972068480	896379008	**893980928**
tai35b	283315445	454981851	574511546	446783959	455718176	472176896	440893216	415469760	**377687744**	409314048
tai40b	637250948	1165811212	1423772477	1019672934	1053339520	1096534400	1022449792	952554880	**917498816**	941171712
tai50b	458821517	796553600	790688128	696556852	764856128	767252544	718563200	676292800	**614638528**	646608448
tai60a	7205962	8614998	8731620	**8243624**	8596094	8449862	8397220	8341650	8281996	8261384
tai60b	608215054	1089964672	1279537664	978843717	994559424	902721984	942202560	909268288	862969152	**856783040**
tai64c	1855928	5893540	6363888	3189566	5703540	2348902	2324478	**2072176**	2133738	2135026
tai80a	13499184	15665790	16069786	15352662	15648708	15608216	15417076	15383582	15283138	**15247172**
tai80b	818415043	1338090880	1410723456	1215586531	1275809408	1209990912	1159786624	1163362432	**1120577408**	1135197312

Table 10 (continued)

Instance	Score									
	Upper	SM	RRWM	SK-JA	NGM	NGM-G5	NGM-G50	NGM-G500	NGM-G5k	IA-NGM
tho30	149936	230828	267194	202844	187062	207424	198456	196072	**185622**	190176
Instance	Time									
	SM	RRWM	SK-JA	NGM	NGM-G5	NGM-G50	NGM-G500	NGM-G5k	IA-NGM	
bur26a	0.02	0.15	309.9	0.02	0.02	0.19	1.85	19.34	5.57	
bur26b	0.01	0.15	191.7	0.02	0.02	0.19	1.88	19.21	4.00	
bur26c	0.01	0.15	136.9	0.02	0.02	0.19	1.9	18.74	5.72	
bur26d	0.02	0.16	276.6	0.02	0.02	0.18	1.89	18.78	3.99	
bur26e	0.01	0.16	52.9	0.03	0.02	0.19	1.87	18.62	4.00	
bur26f	0.01	0.15	173.6	0.02	0.02	0.2	1.86	18.59	3.99	
bur26g	0.01	0.15	292.8	0.02	0.02	0.19	1.87	18.55	3.99	
bur26h	0.01	0.15	330.4	0.02	0.02	0.18	1.87	18.64	5.67	
chr12a	0.01	0.14	75.7	0.01	0.01	0.12	1.15	11.26	3.29	
chr12b	0.01	0.14	75.1	0.01	0.01	0.12	1.16	11.34	3.30	
chr12c	0.01	0.14	97.9	0.01	0.02	0.12	1.12	11.2	3.32	
chr15a	0.01	0.14	683.6	0.01	0.02	0.13	1.26	12.6	3.40	
chr15b	0.01	0.14	461.9	0.01	0.01	0.13	1.26	12.61	3.50	
chr15c	0.01	0.14	214.1	0.01	0.02	0.13	1.25	12.59	3.42	
chr18a	0.01	0.14	781.5	0.01	0.02	0.14	1.37	14	3.57	
chr18b	0.01	0.14	52.1	0.01	0.02	0.14	1.48	13.97	3.50	
chr20a	0.02	0.14	1285.8	0.01	0.02	0.16	1.51	15.02	3.76	
chr20b	0.02	0.14	911.3	0.01	0.02	0.16	1.66	14.96	3.63	
chr20c	0.02	0.14	945	0.01	0.02	0.15	1.68	14.93	3.62	
chr22a	0.02	0.14	1488.4	0.01	0.02	0.17	1.75	16.88	3.75	
chr22b	0.02	0.14	1005.3	0.01	0.02	0.17	1.76	16.1	3.80	

Table 10 (continued)

Instance	Time								
	SM	RRWM	SK-JA	NGM	NGM-G5	NGM-G50	NGM-G500	NGM-G5k	IA-NGM
chr25a	0.03	0.15	2553.2	0.01	0.02	0.19	2.04	17.93	4.96
els19	0.01	0.14	700	0.01	0.02	0.15	1.47	14.49	3.54
esc16a	0	0.14	12.8	0.01	0.02	0.14	1.27	13	3.38
esc16b	0	0.14	4.6	0.01	0.02	0.13	1.29	12.97	3.44
esc16c	0.01	0.14	7.7	0.01	0.01	0.13	1.3	13	3.51
esc16d	0.01	0.14	14.6	0.01	0.02	0.13	1.3	12.95	3.51
esc16e	0	0.14	13.5	0.01	0.02	0.14	1.31	13	3.44
esc16g	0	0.14	17.1	0.01	0.02	0.14	1.3	13.04	3.50
esc16h	0	0.14	15.1	0.01	0.02	0.13	1.29	12.95	3.43
esc16i	0.02	0.14	5625.6	0.01	0.02	0.13	1.31	12.97	3.44
esc16j	0.01	0.14	13	0.01	0.02	0.14	1.3	13.04	3.45
esc32e	0.02	0.15	9661.4	0.02	0.03	0.23	2.42	22.88	6.88
esc32g	0.01	0.15	52135.2	0.02	0.03	0.24	2.35	22.82	7.11
had12	0.01	0.14	–	0.01	0.02	0.11	1.13	11.28	3.40
had14	0.01	0.14	102.2	0.01	0.01	0.12	1.2	12.13	3.36
had16	0.01	0.14	56.7	0.01	0.02	0.13	1.27	12.88	3.46
had18	0.01	0.15	271.4	0.01	0.02	0.14	1.37	13.9	3.53
had20	0.01	0.14	328.4	0.01	0.02	0.15	1.51	14.97	3.61
kra30a	0.01	0.14	491.6	0.01	0.03	0.21	2.22	21.36	5.92
kra30b	0.01	0.14	489.9	0.02	0.02	0.21	2.28	21.35	6.20
kra32	0.01	0.15	479.6	0.02	0.03	0.23	2.42	22.96	6.71
lipa20a	0.01	0.14	271.1	0.01	0.02	0.15	1.64	14.89	3.91
lipa20b	0.01	0.14	73.3	0.01	0.02	0.15	1.65	15.04	3.62
lipa30b	0.03	0.15	160.5	0.02	0.03	0.22	2.28	21.35	6.02

Table 10 (continued)

Instance	Time								
	SM	RRWM	SK-JA	NGM	NGM-G5	NGM-G50	NGM-G500	NGM-G5k	IA-NGM
lipa40a	0.01	0.14	183.2	0.02	0.03	0.31	3.11	30.09	9.54
lipa40b	0.04	0.17	369.3	0.04	0.04	0.31	3.11	30.1	9.58
lipa50b	0.08	0.2	763.5	0.08	0.05	0.44	4.02	41.25	17.55
lipa60a	0.03	0.17	551.5	0.05	0.07	0.58	5.46	54.7	30.21
lipa60b	0.1	0.22	1796.2	0.1	0.09	0.58	5.39	55.37	30.48
lipa70a	0.01	0.17	565.8	0.08	0.11	0.81	7.19	72.61	52.39
lipa70b	0.15	0.25	3592.8	0.15	0.11	0.8	7.11	72.9	52.54
lipa80a	0.01	0.17	1023.4	0.13	0.14	1.01	9.38	94.93	83.89
lipa80b	0.2	0.27	4158	0.2	0.14	0.99	9.29	95.2	83.87
lipa90a	0.08	0.24	1889.5	0.16	0.2	1.31	12.06	122.14	134.81
lipa90b	0.26	0.33	5544.5	0.25	0.21	1.34	12.01	122.29	134.63
nug12	0.01	0.14	11.4	0.01	0.02	0.11	1.12	11.29	3.28
nug14	0.01	0.14	–	0.01	0.01	0.13	1.2	12.1	3.34
nug15	0.01	0.14	69.6	0.01	0.02	0.13	1.24	12.56	3.45
nug16a	0.01	0.14	118.7	0.01	0.01	0.13	1.31	13.01	3.42
nug16b	0.01	0.15	66.8	0.01	0.01	0.14	1.45	12.96	3.43
nug17	0.01	0.14	181.6	0.01	0.02	0.14	1.49	13.44	3.47
nug18	0.01	0.15	155.2	0.01	0.02	0.14	1.58	13.88	3.52
nug20	0.01	0.15	146.7	0.01	0.02	0.16	1.68	14.87	3.65
nug21	0.01	0.14	256.8	0.01	0.02	0.16	1.74	15.52	3.79
nug22	0.01	0.14	382.6	0.01	0.02	0.16	1.73	16.09	3.88
nug24	0.01	0.14	202.6	0.01	0.02	0.18	1.76	17.34	4.59
nug25	0.01	0.14	478.7	0.01	0.03	0.18	1.97	17.95	5.08
nug27	0.01	0.14	360.3	0.01	0.03	0.2	2.12	19.34	4.08

Table 10 (continued)

Instance	Time								
	SM	RRWM	SK-JA	NGM	NGM-G5	NGM-G50	NGM-G500	NGM-G5k	IA-NGM
nug28	0.01	0.14	339.6	0.01	0.03	0.21	2.16	19.96	4.17
nug30	0.01	0.14	330.7	0.02	0.03	0.22	2.32	21.28	4.50
rou12	0.01	0.14	41.9	0.01	0.01	0.12	1.31	11.35	3.25
rou15	0.01	0.14	66.2	0.01	0.01	0.13	1.44	12.49	3.41
rou20	0.01	0.14	115.1	0.01	0.02	0.15	1.66	14.97	3.63
scr12	0.01	0.14	20.8	0.01	0.01	0.12	1.29	11.33	3.26
scr15	0.02	0.14	117.1	0.01	0.02	0.13	1.45	12.62	3.36
scr20	0.01	0.14	220.8	0.01	0.02	0.16	1.68	14.96	3.59
sko81	0.1	0.28	15863.8	0.14	0.15	1.03	9.58	97.28	90.56
sko90	0.16	0.32	16796.6	0.19	0.19	1.28	12.15	122.47	134.68
ste36a	0.02	0.15	2415.2	0.02	0.03	0.26	2.54	26.21	7.90
ste36b	0.02	0.16	3718	0.02	0.03	0.27	2.59	26.32	7.85
ste36c	0.02	0.15	1312.1	0.02	0.03	0.27	2.61	26.42	7.80
tai12a	0.01	0.14	27.1	0.01	0.01	0.11	1.13	11.38	3.29
tai12b	0.01	0.14	225.1	0.01	0.03	0.12	1.15	11.35	3.30
tai15a	0.01	0.14	28.2	0.01	0.02	0.13	1.26	12.51	3.39
tai15b	0.01	0.14	29	0.02	0.01	0.13	1.24	12.56	3.44
tai17a	0.01	0.14	52.4	0.01	0.02	0.14	1.42	13.94	3.49
tai20a	0.01	0.14	82.6	0.01	0.02	0.15	1.63	14.91	3.63
tai20b	0.02	0.15	489.9	0.01	0.02	0.15	1.62	14.89	3.62
tai25b	0.02	0.14	1040	0.01	0.02	0.19	1.93	17.95	4.85
tai30b	0.03	0.15	3464.2	0.02	0.02	0.22	2.23	21.32	4.53
tai35b	0.03	0.15	3440.6	0.03	0.03	0.26	2.53	25.36	5.56
tai40b	0.04	0.15	6646.7	0.04	0.03	0.31	2.91	29.92	6.71

Table 10 (continued)

Instance	Time								
	SM	RRWM	SK-JA	NGM	NGM-G5	NGM-G50	NGM-G500	NGM-G5k	IA-NGM
tai50b	0.08	0.18	12552	0.04	0.05	0.43	3.98	41.11	11.76
tai60a	0.11	0.21	3121.1	0.07	0.08	0.6	5.4	55.35	19.80
tai60b	0.12	0.2	18385.7	0.07	0.07	0.6	5.4	55.34	19.79
tai64c	0.01	0.21	373.4	0.08	0.08	0.64	6	61.7	23.16
tai80a	0.2	0.28	4745.2	0.14	0.14	0.97	9.31	95.09	53.07
tai80b	0.25	0.24	35995.4	0.13	0.15	1	9.24	94.92	53.14
tho30	0.01	0.14	739.1	0.02	0.03	0.21	2.13	21.38	4.49

Best results are in bold

Author contributions Conceptualization: TQ, ST; Methodology: TQ, ST, LX; Formal analysis and investigation: TQ, ST; Writing - original draft preparation: TQ, ST, LX; Writing - review and editing: TQ, ST, L X; Funding acquisition: ST, LX; Supervision: ST, LX.

Funding This work was supported by The National Key Research and Development Program (2018AAA0100700) of the Ministry of Science and Technology of China, and Shanghai Municipal Science and Technology Major Project (2021SHZDZX0102).

Data availability The data and materials used in this article is fully available.

Code availability The codes are available at https://github.com/CMACH508/IA-NGM.

Declarations

Conflict of interest The authors have no conflicts of interests to declare that are relevant to the content of this article.

Ethics approval Not Applicable.

Consent to participate Not Applicable.

Consent for publication Not Applicable.

References

Adams, R.P., & Zemel, R.S. (2011). *Ranking via sinkhorn propagation*. arXiv preprint arXiv:1106.1925.

Bourdev, L., & Malik, J. (2009). Poselets: Body part detectors trained using 3d human pose annotations. In *International Conference on Computer Vision* (pp. 1365–1372).

Burkard, R. E., Karisch, S. E., & Rendl, F. (1997). QAPLIB-a quadratic assignment problem library. *Journal of Global Optimization, 10*(4), 391–403.

Caetano, T. S., McAuley, J. J., Cheng, L., Le, Q. V., & Smola, A. J. (2009). Learning graph matching. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 31*(6), 1048–1058.

Cho, M., Alahari, K., & Ponce, J. (2013). Learning graphs to match. In *Proceedings of the IEEE International Conference on Computer Vision* (pp. 25–32).

Cho, M., Lee, J., & Lee, K.M. (2010). Reweighted random walks for graph matching. In *European Conference on Computer Vision* (pp. 492–505).

Fey, M., Lenssen, J.E., Weichert, F., & Müller, H. (2018). Splinecnn: Fast geometric deep learning with continuous b-spline kernels. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (pp. 869–877).

Jin, Y., Mishkin, D., Mishchuk, A., Matas, J., Fua, P., Yi, K.M., & Trulls, E. (2021). Image matching across wide baselines: From paper to practice. In *International Journal of Computer Vision*, 517–547.

Kushinsky, Y., Maron, H., Dym, N., & Lipman, Y. (2019). Sinkhorn algorithm for lifted assignment problems. *SIAM Journal on Imaging Sciences, 12*(2), 716–735.

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

Leordeanu, M., & Hebert, M. (2005). A spectral technique for correspondence problems using pairwise constraints. *Tenth IEEE International Conference on Computer Vision*. https://doi.org/10.1109/ICCV.2005.20

Rolínek, M., Swoboda, P., Zietlow, D., Paulus, A., Musil, V., & Martius, G. (2020). Deep graph matching via blackbox differentiation of combinatorial solvers. In *European Conference on Computer Vision* (pp. 407–424).

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

Uwents, W., Monfardini, G., Blockeel, H., Gori, M., & Scarselli, F. (2011). Neural networks for relational learning: an experimental comparison. *Machine Learning, 82*(3), 315–349.

Vento, M., & Foggia, P. (2013). Graph matching techniques for computer vision. *Image Processing: Concepts, Methodologies, Tools, and Applications*, 381–421. IGI Global.

Wah, C., Branson, S., Welinder, P., Perona, P., & Belongie, S. (2011). *The caltech-ucsd birds-200-2011 dataset* .

Wang, R., Yan, J., & Yang, X. (2019). Learning combinatorial embedding networks for deep graph matching. In *Proceedings of the IEEE/CVF International Conference on Computer Vision* (pp. 3056–3065).

Wang, R., Yan, J., & Yang, X. (2021). Neural graph matching network: Learning lawler's quadratic assignment problem with extension to hypergraph and multiple-graph matching. *IEEE Transactions on Pattern Analysis and Machine Intelligence*.

Xu, L. (2018). Deep bidirectional intelligence: Alphazero, deep IA-search, deep IA-infer, and TPC causal learning. *Applied Informatics, 5*, 1–38.

Xu, L. (2019). Deep IA-BI and five actions in circling. In *International Conference on Intelligent Science and Big Data Engineering* (pp. 1–21).

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

Zhao, K., Tu, S., & Xu, L. (2021). IA-GM: A deep bidirectional learning method for graph matching. *Proceedings of the AAAI Conference on Artificial Intelligence, 35,* 3474–3482.

Zhou, F., & De la Torre, F. (2015). Factorized graph matching. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 38*(9), 1774–1789.