Clustering-Aware Multiple Graph Matching via Decayed Pairwise Matching Composition

Tianzhe Wang,[†] **Zetian Jiang,**[†] **Junchi Yan**[*]
Shanghai Jiao Tong University
{usedtobe, maple_jzt, yanjunchi}@sjtu.edu.cn

Abstract

Jointly matching of multiple graphs is challenging and recently has been an active topic in machine learning and computer vision. State-of-the-art methods have been devised, however, to our best knowledge there is no effective mechanism that can explicitly deal with the matching of a mixture of graphs belonging to multiple clusters, e.g., a collection of bikes and bottles. Seeing its practical importance, we propose a novel approach for multiple graph matching and clustering. Firstly, for the traditional multi-graph matching setting, we devise a composition scheme based on a tree structure, which can be seen as in the between of two strong multi-graph matching solvers, i.e., MatchOpt (Yan et al. 2015a) and CAO (Yan et al. 2016a). In particular, it can be more robust than MatchOpt against a set of diverse graphs and more efficient than CAO. Then we further extend the algorithm to the multiple graph matching and clustering setting, by adopting a decaying technique along the composition path, to discount the meaningless matching between graphs in different clusters. Experimental results show the proposed methods achieve excellent trade-off on the traditional multi-graph matching case, and outperform in both matching and clustering accuracy, as well as time efficiency.

Introduction

Graph matching (GM) refers to finding node correspondence over two or multiple graphs, based on the affinity information between nodes, edges (Cho, Lee, and Lee 2010; Gold and Rangarajan 1996), and even hyperedges (Lee, Cho, and Lee 2011; Ngoc, Gautier, and Hein 2015; Yan et al. 2015b) over graphs. As a robust tool for structural matching against noise and outliers, graph matching has been applied in various computer vision tasks, and it has also been an intellectual pursuit for devising effective techniques for solving the problem in an approximate manner, as GM, in general, is known NP-hard (Garey and Johnson 1990).

[*]The first two authors contributed equally, whose order of authorship is decided through dice rolling. Junchi Yan is the correspondence author and he is also affiliated with MoE Key Lab of Artificial Intelligence, AI Institute, Shanghai Jiao Tong University. This work is partly supported by NSFC 61972250, the Open Projects Program of National Laboratory of Pattern Recognition.

First, we consider the classic two-graph matching setting involving graph $\mathcal{G}_i$ and $\mathcal{G}_j$, in its quadratic assignment programming (QAP) problem formulation (Loiola et al. 2007), which is also termed by Lawler's QAP (Lawler 1963):

$$J(\mathbf{X}) = \text{vec}(\mathbf{X})^\top \mathbf{K}\text{vec}(\mathbf{X}), \quad \mathbf{X} \in \{0,1\}^{n_i \times n_j}, \quad \mathbf{X}\mathbf{1}_{n_j} = \mathbf{1}_{n_i}, \quad \mathbf{X}^\top \mathbf{1}_{n_i} \leq \mathbf{1}_{n_j} \tag{1}$$

where the (partial) permutation matrix $\mathbf{X}$ indicates node correspondence, and $\mathbf{K} \in \mathbb{R}^{n_i n_j \times n_i n_j}$ is often called affinity matrix (Leordeanu and Hebert 2005). The node-to-node and edge-to-edge affinity information are stored on the diagonal elements and off-diagonal ones, respectively. Apart from these second-order affinity model, there are methods for hypergraph matching (Chertok and Keller 2010; Duchenne et al. 2011; Yan et al. 2015b; Zass and Shashua 2008), which often involve an affinity tensor and are shown more robust against noise, at the cost of higher complexity.

Recently, there are also considerable efforts on multiple graph matching for its practical utility. For real-world applications, one often has to face with many graphs for matching, which also brings opportunities of fusing the information across graphs to achieve more accurate matchings.

Given N graphs and affinity matrix $\mathbf{K}_{ij}$ for each pair $\mathcal{G}_i$ and $\mathcal{G}_j$, (Yan et al. 2015a) shows the following objective:

$$\mathbb{X}^* = \arg\max_{\mathbb{X}} \sum_{i,j=1, i\neq j} \text{vec}(\mathbf{X}_{ij})^\top \mathbf{K}_{ij} \text{vec}(\mathbf{X}_{ij}) \tag{2}$$
$$s.t. \quad \mathbf{X}_{ij}\mathbf{1}_n = \mathbf{1}_n \quad \mathbf{1}_n^\top \mathbf{X}_{ij} = \mathbf{1}_n^\top \quad \mathbf{X}_{ij} = \mathbf{X}_{ji}^\top,$$
$$\mathbf{X}_{ij}, \mathbf{X}_{ji} \in \{0,1\}^{n\times n} \quad \forall i,j = 1,...,N$$

where $\mathbb{X} = \{\mathbf{X}_{ij}\}_{i,j=1}^N$ denotes the matchings for each pair of graphs. Here the assumption is that all graphs contain the same number of nodes and some may be outliers.

Most multi-graph matching solvers (Chen, Guibas, and Huang 2014; Huang and Guibas 2013; Pachauri, Kondor, and Vikas 2013; Wang, Zhou, and Daniilidis 2018) focus on improving the initial matchings obtained by two-graph matchings via the so-called cycle-consistency. In contrast, another line of works tries to combine both affinity and consistency for optimization (Yan et al. 2016a; 2015a).

Nevertheless, all these works assume that all the graphs belong to the same cluster and contain common inliers (with a few outliers) for matching, which is often too ideal. For example, different kinds of fruits are mixed, and matchings of graphs extracted from apples and bananas can be meaningless. Yet existing multi-graph matchings solvers cannot discern such issues as they always enforce one-to-one node matching on two graphs even they are totally different. In this paper, we consider a practically important while surprisingly almost ignored problem in literature: multiple graph matching and clustering whereby the graphs inherently belong to multiple clusters, which is ubiquitous in reality.

First we introduce basic definitions as follows.

Definition 1. *The set of graphs is defined as* $\mathbb{G} = \{\mathcal{G}_1, \ldots, \mathcal{G}_N\}$*, with number of nodes* $|\mathcal{G}_i| = n_i$.

Definition 2. *A cluster partition on* $\mathbb{G}$ *is denoted as* $\mathbf{\Pi} = \{\mathbf{C}_1, \ldots, \mathbf{C}_{n_c}\}$*, where* $\mathbf{C}_i \cap \mathbf{C}_j = \varnothing (\forall i, j = 1, \ldots n_c \cap i \neq j)$*,* $\mathbf{C}_i \neq \varnothing (\forall i = 1, \ldots, n_c)$*,* $\bigcup \mathbf{C}_i = \mathbf{\Pi}$ *and* n_c *is the number of clusters.*

Formally, our problem becomes: given N graphs $\mathbb{G}$ with mixtured clusters in $\mathbf{\Pi}$, and pairwise affinity matrix $\mathbf{K}_{ij}$ for each pair $\mathcal{G}_i$ and $\mathcal{G}_j$, one aims to find a partition $\mathbf{\Pi}^{'} = \{\mathbf{C}_1^{'}, \ldots, \mathbf{C}_{n_c}^{'}\}$ with high clustering quality w.r.t. $\mathbf{\Pi}$, and achieve high matching accuracy for the matching result $\mathbb{X}_k = \{\mathbf{X}_{ij}\}_{\mathcal{G}_i, \mathcal{G}_j \in \mathbf{C}_k}$ within every cluster $\mathbf{C}_k$. We assume the pairwise affinity matrix $\mathbb{K} = \{\mathbf{K}_{ij}\}_{i,j=1}^{N}$ is given.

This paper makes the following main contributions:

i) We formally define the problem of robust matching of a collection of graphs from multiple clusters, such that the meaningful matchings are established within each cluster, and the clusters also need to be automatically identified. Such a joint-clustering-matching setting is practically important while are almost missing in existing literature.

ii) For traditional single-cluster multi-graph matching, we present a solver called tree-based pairwise matching composition (TPMC). It can be seen as in the between of two strong baselines MatchOpt (Yan et al. 2015a) and CAO (Yan et al. 2016a) whereby a maximum spanning tree is built to allow for pairwise matching composition and propagation over the tree. We show in Fig. 3 that it performs competitively in accuracy while being significantly more efficient than one of the best solvers: CAO (Yan et al. 2016a).

iii) For the multi-cluster multi-graph matching setting, we further devise a weighted version (DPMC-C/A) to softly discount the meaningless but unknown inter-cluster matchings, in the hope of improving the intra-cluster matchings. We show in Table 2 and Table 3 the superior performance of our method on the new setting involving graphs from multiple clusters. In particular, our methods outperform CAO (Yan et al. 2016a), which is much more costly than ours.

Figure 1 shows the overall pipeline of our method.

Related Work

The following review focuses on multi-graph matching (MGM), and a more broad survey for graph matching can refer to the survey (Yan et al. 2016b).

For multi-graph matching, cycle-consistency has been a popular regularizer which is based on the basic fact that node correspondence between G_i and G_j shall be consistent with another two-step matching from G_i to G_k and G_k to G_j, which is derived from the intermediate G_k: $\mathbf{X}_{ij} = \mathbf{X}_{ik}\mathbf{X}_{kj}$. Such a consistency has been an effective and popular side information as adopted in many multi-graph matching solvers.

A major line of works admit the pairwise matchings between each pair of graphs as putative input, and then post-smoothing is performed to recover cycle-consistency solutions. Among them, the spectral techniques are widely used (Kim et al. 2012; Pachauri, Kondor, and Vikas 2013; Huang and Guibas 2013). Further improvements are made in the later works (Chen, Guibas, and Huang 2014; Zhou, Zhu, and Daniilidis 2015; Wang, Zhou, and Daniilidis 2018) to handle more flexible matching settings. In (Leonardos, Zhou, and Daniilidis 2017), a decentralized version of the spectral multi-graph matching (Pachauri, Kondor, and Vikas 2013) is devised. While (Hu, Thibert, and Guibas 2018) presents a theoretically sound greedy construction method for partitioning the graphs that allows for separate and parallel matching in each partition. Note in this work, it is assumed that all the graphs still fall into one single cluster, and the partition is enforced in brute-force. In (Yu et al. 2018), incremental matching of graph sequence is addressed, as inspired by (Hu, Thibert, and Guibas 2018).

Different from the above works separating affinity based matching and post-smoothing using cycle-consistency, the work (Yan et al. 2016a; 2014) combines both affinity and consistency for optimization. The authors show that their strategy achieves state-of-the-art performance in terms accuracy over post-smoothing based methods (Pachauri, Kondor, and Vikas 2013; Chen, Guibas, and Huang 2014). Hence we mainly compare with (Yan et al. 2016a) in our experiments.

However, the consistency cannot be directly used in the setting considered in this paper, as the matchings across different clusters are meaningless, so for the derived consistency. Hence we take a different solution, and perhaps the most related work to this paper is (Yan et al. 2015a; 2013), whereby a central reference graph G_r is chosen, and it spans a set of basis $\{\mathbf{X}_{rk}\}_{k=1}^{N}$ encoding two-graph matchings such that other matching can be represented by the cycle: $\mathbf{X}_{ij} = \mathbf{X}_{ri}^{\top}\mathbf{X}_{rj}$. In this paper, we dismiss the concept of hub graph and devise a more flexible matching approach which can handle multiple clusters robustly.

Multi-graph Matching and Clustering

Preliminaries

First we follow the tradition in previous multi-graph matching literature (Yan et al. 2015a; 2016a; Yu et al. 2018) which define the term *supergraph* as follows.

Definition 3. *Given matchings* $\{\mathbf{X}_{ij}\}$ *and affinity matrices* $\{\mathbf{K}_{ij}\}$*, the score matrix is defined as:*

$$\mathbf{S}_{ij} = \left(\text{vec}(\mathbf{X}_{ij})^{\top}\mathbf{K}_{ij}\text{vec}(\mathbf{X}_{ij}) - S_{ij}^{min}\right) / (S_{ij}^{max} - S_{ij}^{min})$$

where $S_{ij}^{max}(S_{ij}^{min})$ *denotes the sum of* $n_i \times n_j$ *largest (smallest) elements in* $\mathbf{K}_{ij}$ *which can ensure the resulting normalized* $\mathbf{S}_{ij}$ *fall into* $[0, 1]$.

Definition 4. *A supergraph* $\mathcal{H}$ *is an undirected complete graph induced by graph set* $\mathbb{G}$ *and pairwise matching* $\mathbb{X}$.

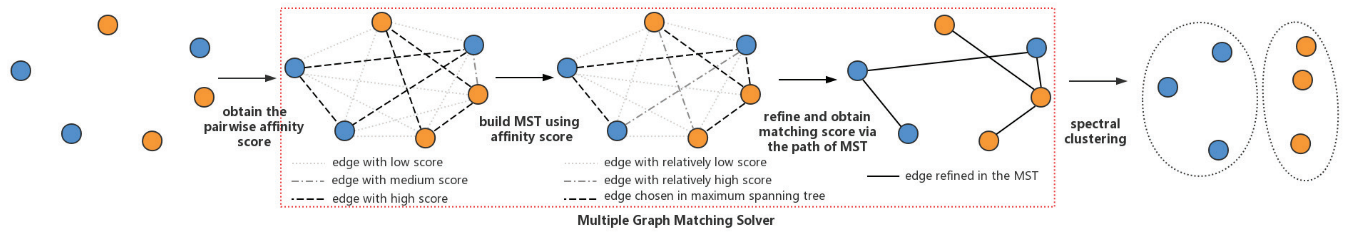

obtain the pairwise affinity score
build MST using affinity score
refine and obtain matching score via the path of MST
spectral clustering
edge with low score
edge with medium score
edge with high score
edge with relatively low score
edge with relatively high score
edge chosen in maximum spanning tree
edge refined in the MST
Multiple Graph Matching Solver

Figure 1: The proposed pipeline (TPMC or DPMC-C/A) of solving multi-graph clustering and matching problem. Blue and orange nodes denote different clusters e.g. object car vs. dog. A maximum spanning tree (see Definition 6) is built on the supergraph (see Definition 4) whereby the edge weights denote the overall matching scores between two graphs (nodes). The edges with higher weight (often within the same cluster) can be more likely chosen in maximum spanning tree.

Its nodes, edges and edge weights denote graphs, pairwise matchings, and pairwise matchings score, respectively: $\mathcal{H} = \{V = \{\mathcal{G}_1, \dots, \mathcal{G}_N\}, E = \mathbb{X}, W = \{S_{ij}\}_{i,j=1}^{N,N}\}$.

Definition 5. *Given a supergraph $\mathcal{H}$, a path from $\mathcal{G}_i$ to $\mathcal{G}_j$ is denoted as $\mathcal{P}_{ij}$, which is a set containing the edges from $\mathcal{G}_i$ to $\mathcal{G}_j$. Specifically, $\mathcal{P}_{ij} = \{\mathbf{X}_{ik_1}, \mathbf{X}_{k_1k_2}, \dots, \mathbf{X}_{k_sj}\}$.*

Definition 6. *Given a supergraph $\mathcal{H}$, a spanning tree $\mathcal{T} = \{V = \{\mathcal{G}_1, \dots, \mathcal{G}_N\}, E = \mathbb{X}' \subsetneq \mathbb{X}\}$ is a sub-supergraph, which contains $N-1$ edges and satisfies that every two vertices has exactly one path in $\mathcal{T}$. Moreover, the spanning tree with the highest accumulated pairwise affinity score over all of its edges is called maximum spanning tree (MST).*

We first discuss a simple pipeline for general discussion. First, we use off-the-shelf multi-graph matching method, such as MatchOpt (Yan et al. 2015a) and CAO (Yan et al. 2016a), to obtain the pairwise affinity score matrix. The matching score between the two graphs is commonly assumed proportional to their similarity. Then we can use spectral clustering (Ding 2004) to obtain the final clustering result given the pairwise score matrix as input.

On the one hand, for multi-graph matching with multiple clusters, we extend Eq. 2 into the following form:

$$\begin{aligned}
\mathbb{X}^* = \arg\max_{\mathbb{X},\mathbf{\Pi}} & \sum_{\mathcal{G}_i,\mathcal{G}_j \in C_k, i\neq j} \mathrm{vec}(\mathbf{X}_{ij})^\top \mathbf{K}_{ij} \mathrm{vec}(\mathbf{X}_{ij}) \quad (3)\\
& + \alpha \sum_{\mathcal{G}_i \in C_k, \mathcal{G}_j \in C_l, k\neq l} \mathrm{vec}(\mathbf{X}_{ij})^\top \mathbf{K}_{ij} \mathrm{vec}(\mathbf{X}_{ij})\\
& s.t. \quad \mathbf{X}_{ij}\mathbf{1}_n = \mathbf{1}_n \quad \mathbf{1}_n^\top \mathbf{X}_{ij} = \mathbf{1}_n^\top\\
& \mathbf{X}_{ij} = \mathbf{X}_{ji}^\top \in \{0,1\}^{n\times n} \quad \forall i,j = 1,...,N
\end{aligned}$$

where we use the constant penalty coefficient α to weight the inter-cluster affinity score as they are in fact meaningless for matching. When $\alpha = 1$, it equals to Eq. 2 and in the ideal case i.e., the clusters are given, one can safely set $\alpha = 0$. However, the challenge is that the true clusters are unknown hence we may still have to consider all the pairwise matching terms for optimization. To fulfill that goal, we will show a soft decaying based approach to fulfill such a goal.

Simplified Analysis. One basic assumption is that intra-cluster graphs can obtain higher pairwise affinity score than inter-cluster graphs, and we analyze this idea as follows.

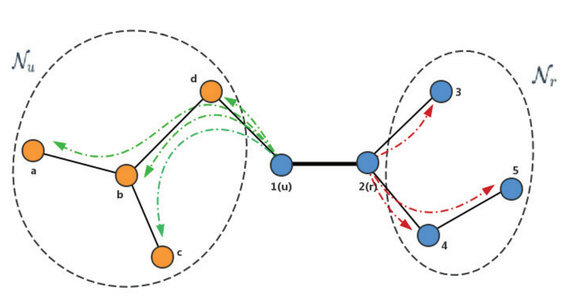
Proposition 1. *For graphs from multiple clusters, the expectation of the affinity score from graph pairs in the same cluster, will be higher than the expectation score of those from different clusters, given the clusters are well separated.*

Proof. Readers are referred to the experiment part for the details of how to construct a synthetic dataset for analysis. Here we give a quick study without loss of generality.

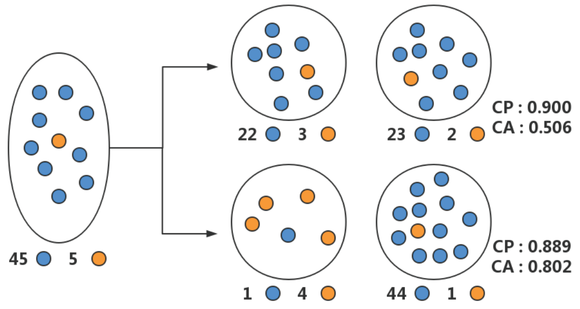
Given two graphs as cluster centers from two clusters a, b and their weighted adjacency matrices $\mathbf{E}_a, \mathbf{E}_b$, whose elements $q_{ij}^{a_0}, q_{ij}^{b_0}$ are generated randomly from Gaussian $U(0,1)$. Using the center graph, a new graph in one cluster a can be generated by adding Gaussian perturbation to the reference graph to get its edge weights: $q_{ij}^{a_1} = q_{ij}^{a_0} + N(0,\upsilon)$. Since the only part of expectation in overall affinity score that varies from inter-cluster and intra-cluster is the affinity score of the same edge pair, e.g., affinity score of edge (i,j) in graph pair (G_a, G_b), we only need to consider the affinity score of same edge pair. Then for the pair of graphs in the same cluster a, this affinity score is bounded by $\mathbb{E}_a = \mathbb{E}[\exp(-\frac{(x-y)^2}{\sigma^2})]$, where $x, y \sim N(0,\upsilon)$, whereas the expectation score of the graph pairs across the clusters a and b is bounded by $\mathbb{E}_b = \mathbb{E}[\exp(-\frac{(x+z-y)^2}{\sigma^2})]$, where $x, y \sim U(0,1), z \sim N(0,\upsilon)$. Obviously, $\mathbb{E}_a > \mathbb{E}_b$ as $\upsilon \to 0$, which exactly indicates the proposition. □

Under this circumstance, traditional multi-graph matching solvers or pairwise matching by traversing all pairs may still obtain distinctive affinity score matrix which can be used to build an appropriate maximum spanning tree on supergraph (see Fig. 1). This observation is important to devise an effective and tailored method for matching graphs with multiple clusters. We further discuss two methods MatchOpt (Yan et al. 2015a) and CAO (Yan et al. 2016a) under the supergraph view, which are mostly related to ours.

Specifically, MatchOpt (Yan et al. 2015a) takes a reference node r on the supergraph and generates every matching $\mathbf{X}_{ru}$ between r and another node u to be updated. Such a single-hub updating scheme inherently can not well handle the mixtured multi-graph matching setting as there are multiple clusters of graphs for matching. The other solver CAO (Yan et al. 2016a) takes a more distributed policy which builds composition pairwise matching to boost the overall



(a) matching updating along maximum spanning tree



(b) counter example for clustering purity

Figure 2: (a) our tree based method: the edge updated in this iteration is $u = 1, r = 2$. $\mathcal{N}_u, \mathcal{N}_r$ are showed. Green and red dashed lines denote $\mathbf{F}_{iu}$, $\mathbf{F}_{ir}$, respectively. (b) counter example of clustering purity. The bottom partition is better than the upper one. CA is more distinctive than CP in this case.

accuracy. It takes a much higher time cost and still cannot handle the mutli-cluster multi-graph matching setting. In summary, MatchOpt and CAO are based on star-shape structure and fully-distributed structure for pairwise matching propagation, respectively. Seeing their pros and cons, we now propose a tree based mechanism which can be seen as in the between for the trade-off of accuracy and time cost. Perhaps more importantly, it is obvious that the star-like model can be fragile to the data with multi-clusters, while we believe the tree-structure is a more robust solution. While the fully distributed design adopted in CAO cannot effectively capture the underlying clustering structure. In the following, we first present our tree-structured multi-graph matching solver, and then its tailored version for the multi-graph mutli-cluster setting is described in detail.

Maximum Spanning Tree Multi-Graph Matching

In this subsection, we show a method which assumes all the graphs belong to a single cluster. There are multiple paths from $\mathcal{G}_i$ to $\mathcal{G}_j$, each of which can yield a composition $\bar{\mathbf{X}}_{ij}$ as shown in Eq. 4 (Yan et al. 2015a). Therefore we simplify the supergraph into a spanning tree $\mathcal{T}$, where any two graphs are connected by exactly one path. Let $\mathcal{P}_{ij}$ denote the unique path from $\mathcal{G}_i$ to $\mathcal{G}_j$ on $\mathcal{H}$, and $\bar{\mathbf{X}}_{ij}$ denote the composition matching yielded along $\mathcal{P}_{ij}$. The objective then becomes:

$$\mathbb{X}^* = \arg\max_{\mathbb{X}} \sum_{i,j=1, i\neq j} \text{vec}(\bar{\mathbf{X}}_{ij})^\top \mathbf{K}_{ij} \text{vec}(\bar{\mathbf{X}}_{ij}) \tag{4}$$
$$s.t. \quad \bar{\mathbf{X}}_{ij} = \mathbf{X}_{ik_1}\mathbf{X}_{k_1k_2}\dots\mathbf{X}_{k_sj}; \mathbf{X}_{ik_1}, \mathbf{X}_{k_tk_{t+1}}, \mathbf{X}_{k_sj} \in \mathcal{P}_{ij}$$

As higher affinity score for a pairwise matching often indicates better matching accuracy, we hereby use Kruskal's Algorithm (Kruskal 1956) to generate a maximum spanning tree (MST) to reserve the edges with the highest affinity score on the supergraph. Then, we update the total $N-1$ edges on $\mathcal{T}$ by a certain order. In each iteration, we fix the other $N-2$ edges and optimize the chosen edge. We can further derive the following objective by dropping some constant terms from Eq. 4 which consists of the right part ($\mathcal{N}_r$) and left part ($\mathcal{N}_u$) of $\mathcal{G}_u$, as sketched in Fig. 2(b):

$$\begin{aligned} J_{tree}(\bar{\mathbf{X}}_{ur}) =& J(\bar{\mathbf{X}}_{ur}) + \sum_{f\in\mathcal{N}_r} J(\bar{\mathbf{X}}_{uf}) + \sum_{f\in\mathcal{N}_u} J(\bar{\mathbf{X}}_{rf}) \\ =& J(\mathbf{X}_{ur}) + \sum_{f\in\mathcal{N}_r} J(\mathbf{X}_{ur}\mathbf{X}_{rk_1}\dots\mathbf{X}_{k_sf}) \\ &+ \sum_{f\in\mathcal{N}_u} J(\mathbf{X}_{ru}\mathbf{X}_{uk_1}\dots\mathbf{X}_{k_sf}) \end{aligned} \tag{5}$$

where $J(\bar{\mathbf{X}}_{ij}) = \text{vec}(\bar{\mathbf{X}}_{ij})^\top \mathbf{K}_{ij} \text{vec}(\bar{\mathbf{X}}_{ij})$, and the right part (see Fig. 2(a)) $\mathcal{N}_r = \{f | f \neq r \cap |\mathcal{P}_{f,r}| < |\mathcal{P}_{f,u}|\}$, so for the left part $\mathcal{N}_u = \{f | f \neq u \cap |\mathcal{P}_{f,u}| < |\mathcal{P}_{f,r}|\}$.

Consider all $f \in \mathcal{N}_r$, $\bar{\mathbf{X}}_{uf}$ can be decomposed into $\mathbf{X}_{ur}\bar{\mathbf{X}}_{rf}$. Let $\mathbf{F}_{fr} = \bar{\mathbf{X}}_{fr} \otimes \mathbf{I}$, we will have:

$$\text{vec}(\bar{\mathbf{X}}_{uf}) = (\bar{\mathbf{X}}_{fr} \otimes \mathbf{I})\text{vec}(\mathbf{X}_{ur}) = \mathbf{F}_{fr}\text{vec}(\mathbf{X}_{ur}) \tag{6}$$

Likewise, for all $f \in \mathcal{N}_u$, we can represent $\text{vec}(\bar{\mathbf{X}}_{rf})$ with $\text{vec}(\mathbf{X}_{ur})$ by a permutation matrix $\mathbf{M}$:

$$\text{vec}(\bar{\mathbf{X}}_{rf}) = \mathbf{F}_{fu}\text{vec}(\mathbf{X}_{ru}) = \mathbf{F}_{fu}\mathbf{M}\text{vec}(\mathbf{X}_{ur}) \tag{7}$$

Then, the objective can be rewritten into:

$$\begin{aligned} J_{tree}(\mathbf{X}_{ur}) = \text{vec}(\mathbf{X}_{ur})^\top \Bigg(\mathbf{K}_{ur} + \sum_{f\in\mathcal{N}_r} \mathbf{F}_{fr}^\top \mathbf{K}_{uf} \mathbf{F}_{fr} \\ + \sum_{f\in\mathcal{N}_u} \mathbf{M}^\top \mathbf{F}_{fu}^\top \mathbf{K}_{rf} \mathbf{F}_{fu} \mathbf{M} \Bigg) \text{vec}(\mathbf{X}_{ur}) \end{aligned} \tag{8}$$

The resulting sub-problem equals to a standard QAP formulation and can be readily solved using off-the-shelf two-graph matching solver e.g., RRWM (Cho, Lee, and Lee 2010). We call the above algorithm as **tree based pairwise matching composition (TPMC)**. In the following, we will further develop a tailored approach against multiple clusters of graphs based on the above analysis.

Algorithm 1: Decayed Pairwise Matching Composition for Multi-cluster Multi-graph Matching

```
Input: affinity matrix $\{\mathbf{K}_{ij}\}_{i,j=1}^{N}$, pairwise matching $\mathbb{X}$
       obtained by a certain means e.g., two-graph
       matching solver RRWM (Cho, Lee, and Lee
       2010) or multi-graph matching solver CAO (Yan
       et al. 2016a), and the resulting affinity score
       $\{\mathbf{S}_{ij}\}_{i,j=1}^{N}$.
1  Use $\{\mathbf{S}_{ij}\}_{i,j=1}^{N}$ to generate a MST $\mathcal{T}$ and update order
     $\mathcal{O}_{upd}$ by Kruskal's Algorithm (Kruskal 1956).
2  if use Constant-Decay then
3      Set $\gamma(e) = c$ (0.7) for all $e \in \mathcal{T}$ and calculate
         $\gamma(\mathcal{P}_{ij})$.
4  else if use Weighted-Decay then
5      Set $\gamma_{min}$(default=0.3), $\gamma_{max}$(default=0.9).
6      Set $\gamma(e)$ following Eq. 10 and calculate $\gamma(\mathcal{P}_{ij})$.
7  for i = 1 ... iterMax do
8      for each edge $\mathbf{X}_{ur}$ in $\mathcal{O}_{upd}$ do
9          Obtain $\mathbf{F}_{fr}$ and $\mathbf{F}_{fu}$ using depth-first search on
             $\mathcal{T}$.
10         Refine the matching $\mathbf{X}_{ur}$ by solving Eq. 9 with
             a pairwise matching solver e.g. RRWM.
11     for each pair of $\mathcal{G}_i$ and $\mathcal{G}_j$ in $\mathcal{H}$ do
12         Set $\mathbf{X}_{ij} = \bar{\mathbf{X}}_{ij}$ via $\mathcal{P}_{ij}$ to update each pairwise
             matching.
Output: Optimized matching $\mathbb{X}$.
```

Decayed Pairwise Matching Composition for Multi-cluster Multi-graph Matching

Unknowing the cluster label of each graph makes the objective difficult to optimize. We circumvent the explicit objective in Eq. 3 and resort to the maximum spanning tree $\mathcal{T}$ that leads to an implicit and (conceptually) soft-α version as an approximation to Eq. 3. The hope is that, two intra-cluster graphs are adjacent or close on $\mathcal{T}$, while those from different clusters have a longer path in $\mathcal{T}$. In particular, the inter-cluster matching is meaningless whose contribution to the overall objective in Eq. 8 shall be suppressed.

As such, we introduce a decaying mechanism to penalize the pairs believed belonging to different clusters. Specifically, we modify the objective $J_{tree}(\mathbf{X}_{ur})$ into:

$$\text{vec}(\mathbf{X}_{ur})^{\top}\left(\mathbf{K}_{ur} + \sum_{f\in\mathcal{N}_r}\gamma(\mathcal{P}_{rf})\mathbf{F}_{fr}^{\top}\mathbf{K}_{uf}\mathbf{F}_{fr} + \sum_{f\in\mathcal{N}_u}\gamma(\mathcal{P}_{uf})\mathbf{M}^{\top}\mathbf{F}_{fu}^{\top}\mathbf{K}_{rf}\mathbf{F}_{fu}\mathbf{M}\right)\text{vec}(\mathbf{X}_{ur}) \tag{9}$$

where $\gamma(\mathcal{P}_{ij}) = \prod_{e\in\mathcal{P}_{ij}}\gamma(e) \in [0,1]$ is the decay rate.

We suggest two mechanisms for $\gamma(e)$. The first is setting constant $\gamma(e) \in [0,1]$. The other is a weight-decay version:

$$\gamma(e) = \lambda\gamma_{max} + (1-\lambda)\gamma_{min}, \tag{10}$$

$$\lambda = \frac{\mathbf{S}_e - \min_{t\in\mathcal{T}}(\mathbf{S}_t)}{\max_{t\in\mathcal{T}}(\mathbf{S}_t) - \min_{t\in\mathcal{T}}(\mathbf{S}_t)}, 0 < \gamma_{min} < \gamma_{max} \leq 1$$

The detailed processes of this algorithm can be seen in Alg. 1 and the algorithms based on the above two variants are abbreviated as DPMC-C and DPMC-A, respectively.

Experiments

Protocols for Peer Methods, Metrics and Datasets.

We mainly compare two competitive multi-graph matching solvers i.e., MatchOpt (Yan et al. 2015a) and CAO-C (and its approximate but efficient variant CAO-PC) (Yan et al. 2016a) as the source code is publicly available. Moreover, CAO is one of the state-of-the-art multi-graph matching solvers and in our experiments it outperforms many consistency smoothing methods e.g. (Chen, Guibas, and Huang 2014; Zhou, Zhu, and Daniilidis 2015) hence for clarity we mainly compare it in our experiments.

Our experiments need to evluate both matching accuracy and clustering quality. We first employ **Cluster Purity** and **Rand Index** as well-defined metrics for data clustering.

1) Clustering Purity (CP): average of the portion of true positive class in each cluster, given by (Schütze, Manning, and Raghavan 2008): $Purity = \frac{1}{N}\sum_{k=1}^{K}\max_{i\in\{1,\ldots,K'\}}|\mathcal{W}_k \cap \mathcal{C}_i|$, where $\mathcal{W}_k$ is the learned index set belonging to cluster k, $\mathcal{C}_i$ is the real index set of graphs belonging to class i, N is the total number of graphs. Higher purity indicates more concentration in each cluster.

2) Rand Index (RI): By treating the labels as a clustering ground truth, RI can be used as clustering accuracy (the higher the better), measuring the similarity between the clustering and real labels (Rand 1971): $RI = \frac{n_{11}+n_{00}}{n} \in [0,1]$, where n_{11} is the number of graph pairs that are in the same cluster with the same label, and n_{00} is the number of pairs that are in different clusters with different labels.

However, these two metrics have their own limitations when evaluating some extreme case, i.e., a high score not always indicates good performance.

i) Clustering Purity: This metric is not able to deal with unbalanced data in certain cases, whereby a bad clustering result may lead to good CP score, as shown in Fig. 2(b).

ii) Rand Index: It is within [0,1], while has different lower bounds for different cases e.g., it can never be lower than 0.8 when $n_c \times n_g = 10 \times 10$. Hence it may not be a consistent metric for evaluating clustering performance across different numbers of cluster and cluster sizes.

Therefore we further propose two metrics: **Affinity Contrast (AC)** and **Clustering Accuracy (CA)**.

For convenience, we use $\mathcal{A}$, $\mathcal{B}$, ... to denote the ground-truth clusters, $\mathcal{A}'$, $\mathcal{B}'$, ... for the clusters derived by the above methods and $\mathcal{G}_a$, $\mathcal{G}_b$, $\mathcal{G}_{a'}$, $\mathcal{G}_{b'}$,... as a single graph.

3) Affinity Contrast (AC), quotient of intra-cluster average score and inter-cluster average score, measures the relative discrimination between intra-cluster and inter-cluster:

$$\text{AC} = \frac{\sum_{\mathcal{A}}\sum_{\mathcal{G}_a\neq\mathcal{G}_b\in\mathcal{A}}\mathbf{S}_{ab}}{\sum_{\mathcal{A}}\sum_{\mathcal{G}_a\neq\mathcal{G}_b\in\mathcal{A}}1} \times \frac{\sum_{\mathcal{A}\neq\mathcal{B}}\sum_{\mathcal{G}_a\in\mathcal{A},\mathcal{G}_b\in\mathcal{B}}1}{\sum_{\mathcal{A}\neq\mathcal{B}}\sum_{\mathcal{G}_a\in\mathcal{A},\mathcal{G}_b\in\mathcal{B}}\mathbf{S}_{ab}} \tag{11}$$

4) Clustering Accuracy (CA), which evaluates the relative clustering performance by calculating the intra-class and inter-class score of every pair $(\mathcal{G}_i, \mathcal{G}_j)$, as given by:

$$\text{CA} = 1 - \frac{1}{C}\left(\sum_{\mathcal{A}}\sum_{\mathcal{A}'\neq\mathcal{B}'}\frac{|\mathcal{A}'\cap\mathcal{A}||\mathcal{B}'\cap\mathcal{A}|}{|\mathcal{A}|\cdot|\mathcal{A}|} + \sum_{\mathcal{A}'}\sum_{\mathcal{A}\neq\mathcal{B}}\frac{|\mathcal{A}'\cap\mathcal{A}||\mathcal{A}'\cap\mathcal{B}|}{|\mathcal{A}|\cdot|\mathcal{B}|}\right) \tag{12}$$

Moreover, we adapt the accuracy for single cluster to multi-cluster by only considering the intra-cluster matchings, which we call **Matching Accuracy (MA)** as given by:

$$\text{Matching Accuracy} = \frac{1}{\sum_{\mathcal{A}'}|\mathcal{A}'|}\sum_{\mathcal{A}'}|\mathcal{A}'|\cdot Acc(\mathcal{A}') \tag{13}$$

where $Acc(\mathcal{A}')$ denotes the accuracy for single cluster $\mathcal{A}'$.

Note that all the experiments are performed on a laptop with 2.60GHZ 4-core CPU and 12G memory.

1) Synthetic dataset. We first randomly generate n_c 'reference' adjacency matrices $\mathbf{E}_i^r, i = 1, 2, .., n_c$ of a complete graph. For each 'reference' adjacency matrix, we add Gaussian perturbation to each 'reference' edge to generate n_g graphs belonging to this cluster. Specifically, the 'perturbed' edge weight q_{ij}^p is calculated by: $q_{ij}^p = q_{ij}^{r_k} + N(0, \upsilon), k = 1, 2, ..., n_c$, where υ is a hyperparameter of deformation. The edge affinity can be calculated by $\mathbf{K}_{ac;bd} = \exp(-\frac{(q_{ab}-q_{cd})^2}{\sigma^2})$, where σ^2 is the similarity sensitivity parameter. For outlier test, we follow the common inlier setting, whereby equal number of n_o outliers are added in each graph with the same number of inliers. As such, the matching matrix is always a permutation one to facilitate the derivation of our technique. This protocol has also been widely used in (Yan et al. 2016a; Yu et al. 2018). For different numbers of outliers, one can introduce dummy nodes to make equal size of graphs (Zhou and Torre 2012).

2) Real-world dataset. Willow-ObjectClass (Cho, Alahari, and Ponce 2013) contains images from Caltech-256 and PASCAL VOC2007, which are categorized into 5 classes: 109 Face, 66 Winebottle, 50 Duck, 40 Car and 40 Motorbike. We choose the three clusters except for 'Face' and 'Winebottle' as they are too easy for matching. With 10 points manually labeled for each image, we additionally randomly generate n_o outliers for outlier test. The adjacency matrix is constructed by sparse Delaunay triangulation. We set the affinity matrix re-weighted by $\beta \in [0, 1]$ for both length and angle affinity. We randomly choose n_g graphs for each cluster and mix them together as our real dataset.

Sparsification on affinity score matrix.

To make the input matrix more suitable for clustering, we adopt the standard processing (Ding 2004) to sparsify the matrix. For each pair of two graphs whose matching score corresponding to the element in $\mathbf{S}$, if either one is among the k-nearest neighbors of the other, or vice versa, the element is unchanged. Otherwise, it is zeroed. The parameter k is set to 10. There is another way of sparsifying via thresholding (Ding 2004), by which we find similar results are obtained hence we only report the results using k-NN preprocessing.

Table 1: Parameter setting details for experiments. Inlier #: n_i, outlier #: n_o, deform: υ, k-neighbor: k, sensitivity: σ^2, reweight: β, cluster #: n_c, cluster size: n_g.

Fig. 3: deform	$n_c = 1, n_g = 30, n_i = 10, n_o = 0, \upsilon = 0.15, \sigma^2 = .05$
Fig. 3: outlier	$n_c = 1, n_g = 30, n_i = 6, n_o = 4, \upsilon = 0.00, \sigma^2 = .05$
Fig. 3: real	$n_c = 1, n_g = 30, n_i = 10, n_o = 2, \beta = 0.9, \sigma^2 = .1$
Table 2: syn	$n_c = 5, n_g = 6, n_i = 10, n_o = 2, \upsilon = 0.1, \sigma^2 = .05, k = 10$
Table 2: real	$n_c = 3, n_g = 10, n_i = 10, n_o = 2, \beta = 0.9, \sigma^2 = .03, k = 10$
Fig 4: syn	$n_c = 5, n_g = 6, n_i = 10, n_o = 2, \upsilon = 0.1, \sigma^2 = .05, k = 10$
Fig 4: real	$n_c = 4, n_g = 10, n_i = 10, n_o = 1, \beta = 0.9, \sigma^2 = .03, k = 10$
Table 3: syn	$n_i = 10, n_o = 2, \upsilon = 0.1, \sigma^2 = .03, k = 10$
Table 3: real	$n_i = 10, n_o = 1, \beta = 0.9, \sigma^2 = .03, k = 10$

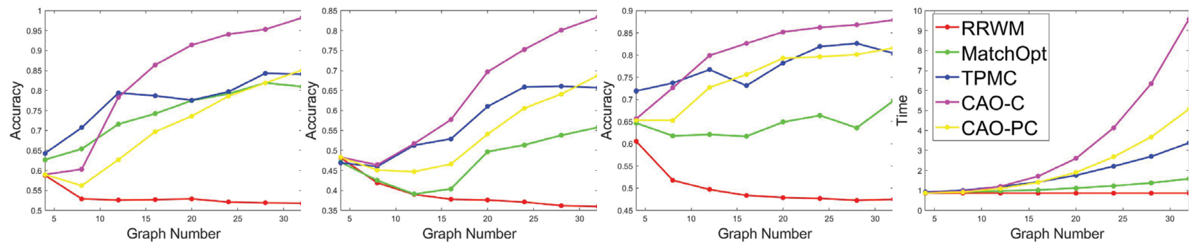
Evaluation Results

Results on single cluster multi-graph matching. Fig. 3 shows TPMC's performance as a traditional graph matching solver on synthetic and real-world datasets. We compare TPMC with MatchOpt and CAO-based methods in accuracy and time metrics. Note that the accuracy is identical to the accuracy metric for existing graph matching problem.

Specifically, TPMC has a significant improvement over MatchOpt on the accuracy, e.g., the accuracy increases about 11% to 19% on the car and deform dataset in Fig. 3(c) and Fig. 3(b). Compared with CAO-PC, it achieves nearly the same performance with a smaller time cost: TPMC has 1% to 2% accuracy loss than CAO-PC and nearly $2\times$ faster with regard to time cost across all dataset, as shown in Fig. 3(d). Moreover, significant speedup is spotted in Fig. 3(d) when comparing with CAO-C: it has nearly $3\times$ speedup with 6% accuracy cost on the real dataset. In general, TPMC gains better accuracy than MatchOpt and achieves competitive accuracy with CAO-based methods with a much small cost of time. Detailed settings can be found in Table 1.

Results on multi-cluster multi-graph matching setting. We further compare our decay algorithms with some state-of-the-arts, CAO-C, CAO-PC and RRWM for multi-clusters in graph matching. We use the sparsification technique as mentioned above to get the score matrix. We can see in Table 2 that both CAO and our DPMC-A outperform raw pairwise matching algorithm. Our algorithm, which has a smaller time cost than CAO, achieves competitive performance in all metrics. On the real dataset, one can see the significant accuracy and clustering metric improvement from the raw pairwise matching solver. It is also worth noting that our algorithms consistently surpass CAO-PC in every metric: the accuracy improves 3% to 6%, well as 20% speedup, as shown in Table 2. Compared with CAO-C, our methods take a small accuracy cost, i.e., 1% to achieve the competitive clustering result and nearly $3\times$ speedup: the clustering accuracy increases 4% to 6%, affinity contrast improves about 0.5, as well as consistent increment of CP and RI.

Results on varying the decay rate γ for DPMC-C. Fig. 4 shows the matching performance with various decay rates. As one can see, the overall performance remains stable when decay rate is in range [0.1, 0.7]. However, further increasing the decay rate will result in a significant performance drop: the clustering metric MA drops about 0.1 and 0.2 on the real and synthetic dataset, respectively. This means our DPMCF



(a) accuracy: outlier (b) accuracy: deform (c) accuracy: car (d) time: car

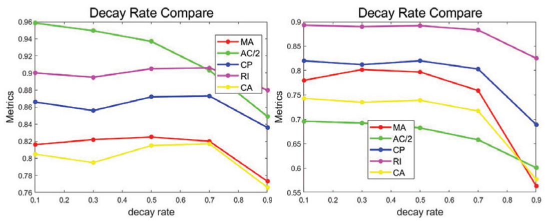
Figure 3: Evaluation on synthetic data (deform, outlier) and objects from Willow ObjectClass for single-cluster multi-graph matching. Our method TPMC performs more efficient and accurate than CAO-PC, while significantly faster than CAO-C.

Table 2: Evaluation of intra-cluster matching accuracy (MA) and clustering metric for the multi-cluster multi-graph matching.

metric	real-world dataset					synthetic dataset				
	RRWM	DPMC-C	DPMC-A	CAO-C	CAO-PC	RRWM	DPMC-C	DPMC-A	CAO-C	CAO-PC
MA	0.715	**0.887**	**0.887**	0.877	0.843	0.651	0.767	**0.796**	0.769	0.752
CP	0.907	**0.963**	**0.963**	0.937	0.937	0.653	0.820	**0.839**	0.767	0.771
RI	0.902	**0.956**	**0.956**	0.930	0.928	0.812	0.894	**0.903**	0.866	0.870
AC	1.511	1.874	**1.885**	1.373	1.351	1.081	1.322	**1.346**	1.091	1.091
CA	0.858	**0.936**	**0.936**	0.899	0.896	0.545	0.744	**0.765**	0.676	0.686
time	**0.804**	2.868	2.838	9.358	4.064	**1.657**	5.045	5.022	11.465	3.421

Table 3: Intra-cluster matching accuracy by varying cluster number and number of graphs in cluster or unbalancing cluster size.

	real-world dataset						synthetic dataset				
$n_c \times n_g$	RRWM	DPMC-C	DPMC-A	CAO-C	CAO-PC	$n_c \times n_g$	RRWM	DPMC-C	DPMC-A	CAO-C	CAO-PC
2×15	0.887	**0.940**	0.936	0.906	0.903	4×10	0.751	0.838	0.847	**0.874**	0.860
3×10	0.858	**0.936**	**0.936**	0.899	0.896	5×8	0.547	0.713	**0.717**	0.658	0.667
4×8	0.681	0.746	**0.779**	0.686	0.704	6×7	0.496	0.672	**0.675**	0.578	0.605
5×6	0.616	**0.701**	0.700	0.621	0.606	7×6	0.437	0.581	**0.609**	0.512	0.511
$1 \times [5, 10, 20]$	0.578	0.597	**0.601**	0.575	0.571	$4 \times 5, 1 \times 20$	0.440	0.506	**0.519**	0.490	0.503
$3 \times 5, 1 \times 15$	0.605	**0.680**	0.673	0.630	0.623	$3 \times 5, 1 \times [10, 15]$	0.542	0.613	**0.617**	0.576	0.613



(a) real data (b) synthetic data

Figure 4: Evaluation on synthetic data and objects from Willow ObjectClass for DPMC-C by varying the decay rate γ. When the decay rate increases more than 0.7, the performance of matching will be largely affected. Here AC/2 means the value of Affinity Contrast divided by 2.

is necessary for obtaining a good clustering result, and verifies the performance gain led by introducing a decay rate to punish the node in supergraph with lareger distance.

Results on varying cluster number and cluster size. Table 3 verifies the robustness of our methods with CAO-based methods on different cluster numbers and unbalanced cluster size. For cluster number test, our decay methods consistently outperforms CAO-C on clustering accuracy, while achieves a greater advantage over other methods as cluster number increases: the clustering accuracy gap with CAO-based methods goes from 3% to 8% when more clusters are split. In unbalanced cluster size test, nearly the same accuracy is achieved by both our two decay methods, which surpasses CAO-C by 3% to 5% on clustering accuracy.

Table 4: Clustering accuracy (CA) after DPMC by different methods for initialization. It is insensitive to initialization.

CA	RRWM	CAO-C	CAO-PC	MatchOpt
Init	0.541	0.689	0.658	0.351
DPMC	0.731	0.762	0.768	0.350

Results on initialization of DPMC. Table 4 verifies the robustness of our DPMC against different affinity matrices for MST construction. We obtain four different initializations via RRWM, CAO-C, CAO-PC, MatchOpt and evaluate their clustering performance (CA) by applying spectral clustering on each initial matrix. Then we apply DMPC to get final affinity matrices and evaluate them in the same way.

Note that when the initialization is reasonable (RRWM, CAO), DPMC can achieve good clustering performance given a decrease of 0.14 on initialization: CA only decreases 0.03 in RRWM initialization compared with CAO-PC initialization. However, if the initialization is bad (MatchOpt), DPMC still has difficulty in further improvement.

Time complexity analysis. The speedup compared with CAO benefits from lower time complexity bound: our method, similar with MatchOpt, updates each edge along spanning tree $\mathcal{T}$, generally costs $\mathcal{O}(N^2 \log N + n^4 N^2)$ in total, where N and n denote the number of graphs and nodes, respectively. Specifically, it takes $\mathcal{O}(N^2 \log N)$ to construct the maximum spanning tree and $\mathcal{O}(n^3 N^2)$ to update edges via depth-first searching. In contrast, CAO based methods update every edge for iterations via composition paths, and suffer a lot of calculation time for pair consistency, i.e., it takes $\mathcal{O}(nN)$ time cost for each pair. Admittedly, due to the tree construction and depth-first searching, our method is relatively slower than MatchOpt, which only costs $\mathcal{O}(n^4 N^2 + n^3 N^3)$. We leave out the time cost of initial pairwise matching since it is used in all the methods.

Conclusion

This paper strikes an endeavor for robust matching of multiple graphs from a mixture of clusters (or clusters), which has not been addressed but important for practical applications. We have presented a tree based framework whereby compositional pairwise matching can be derived and the technique can be viewed as in the between of two strong baselines MatchOpt (Yan et al. 2015a) and CAO (Yan et al. 2016a) which achieves strong cost-effectiveness on traditional multi-graph matching problem. We then develop a decaying version that focuses more on within-cluster matching and show the first solver, to our best knowledge, for the multi-cluster multi-graph matching task. It outperforms by a significant margin against the existing multi-graph matching methods. Source code will be made publicly available.

References

Chen, Y.; Guibas, L.; and Huang, Q. 2014. Near-optimal joint object matching via convex relaxation. In *ICML*.

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

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

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

Ding, C. 2004. A tutorial on spectral clustering. In *Talk presented at ICML.(Slides available at http://crd. lbl. gov/ cding/Spectral/)*.

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

Garey, M. R., and Johnson, D. S. 1990. *Computers and Intractability; A Guide to the Theory of NP-Completeness*. W. H. Freeman & Co.

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

Hu, N.; Thibert, B.; and Guibas, L. 2018. Distributable consistent multi-graph matching. In *CVPR*.

Huang, Q., and Guibas, L. 2013. Consistent shape maps via semidefinite programming. In *Proc. Eurographics Symposium on Geometry Processing (SGP)*.

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

Kruskal, J. B. 1956. On the shortest spanning subtree of a graph and the traveling salesman problem. *Proceedings of the American Mathematical society* 7(1):48–50.

Lawler, E. L. 1963. The quadratic assignment problem. *Management Science*.

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

Leonardos, S.; Zhou, X.; and Daniilidis, K. 2017. Distributed consistent data association. *ICRA*.

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

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

Ngoc, Q.; Gautier, A.; and Hein, M. 2015. A flexible tensor block coordinate ascent scheme for hypergraph matching. In *CVPR*.

Pachauri, D.; Kondor, R.; and Vikas, S. 2013. Solving the multi-way matching problem by permutation synchronization. In *NIPS*.

Rand, W. M. 1971. Objective criteria for the evaluation of clustering methods. *Journal of the American Statistical association* 66(336):846–850.

Schütze, H.; Manning, C. D.; and Raghavan, P. 2008. *Introduction to information retrieval*, volume 39. Cambridge University Press.

Wang, Q.; Zhou, X.; and Daniilidis, K. 2018. Multi-image semantic matching by mining consistent features. In *CVPR*.

Yan, J.; Tian, Y.; Zha, H.; Yang, X.; Zhang, Y.; and Chu, S. 2013. Joint optimization for consistent multiple graph matching. In *ICCV*.

Yan, J.; Li, Y.; Liu, W.; Zha, H.; Yang, X.; and Chu, S. 2014. Graduated consistency-regularized optimization for multi-graph matching. In *ECCV*.

Yan, J.; Wang, J.; Zha, H.; Yang, X.; and Chu, S. 2015a. Consistency-driven alternating optimization for multigraph matching: A unified approach. *TIP*.

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

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

Yan, J.; Yin, X.; Lin, W.; Deng, C.; Zha, H.; and Yang, X. 2016b. A short survey of recent advances in graph matching. In *ICMR*.

Yu, T.; Yan, J.; Liu, W.; and Li, B. 2018. Incremental multi-graph matching via diversity and randomness based graph clustering. In *ECCV*.

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

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

Zhou, X.; Zhu, M.; and Daniilidis, K. 2015. Multi-image matching via fast alternating minimization. In *ICCV*.