Incremental Multi-graph Matching via Diversity and Randomness based Graph Clustering

Tianshu Yu[1], Junchi Yan[2], Wei Liu[3], Baoxin Li[1]

[1]Arizona State University, [2]Shanghai Jiao Tong University, [3]Tencent AI Lab
{tianshuy,baoxin.li}@asu.edu,yanjunchi@sjtu.edu.cn,wl2223@columbia.edu

Abstract. Multi-graph matching refers to finding correspondences across graphs, which are traditionally solved by matching all the graphs in a single batch. However in real-world applications, graphs are often collected incrementally, rather than once for all. In this paper, we present an incremental multi-graph matching approach, which deals with the arriving graph utilizing the previous matching results under the global consistency constraint. When a new graph arrives, rather than re-optimizing over all graphs, we propose to partition graphs into subsets with certain topological structure and conduct optimization within each subset. The partitioning procedure is guided by the diversity within partitions and randomness over iterations, and we present an interpretation showing why these two factors are essential. The final matching results are calculated over all subsets via an intersection graph. Extensive experimental results on synthetic and real image datasets show that our algorithm notably improves the efficiency without sacrificing the accuracy.

Keywords: Multi-graph matching; Incremental graph matching; Determinantal point process; Graph clustering

1 Introduction

Graph matching (GM), which refers to the problem of finding common vertex correspondences over a set of graphs by exploring both unary (vertex) and pairwise (edge) affinity, is a fundamental problem in computer vision and is known to be NP-hard [1]. Compared with vector data, expressive graph representation is often more welcomed when structural relation need to be considered. Due to its robustness against noise, GM has been adopted in various vision applications e.g. scene understanding [2], visual tracking [3], and object recognition [4], etc.

Though proven successful in exploiting structural data, two-graph matching still suffers from the inherent local ambiguity, which on one hand leads to non-smooth and difficult optimization problem. On the other hand, the affinity objective can be biased from ground truth correspondence. Though learning the affinity function [5] can mitigate such an issue, bringing more graphs for joint matching can be a more natural and effective way to dismiss local bias [6, 7].

However, rather than being obtained at one time, graphs are often collected over time in practice, e.g. photos taken by street-view vehicle, video from

surveillance camera, newly discovered protein v.s. existing protein. For this setting, the naive strategy by treating the old batch and new graphs as a new batch for matching from scratch is inefficient. Given previous matching results, the problem arises for how to utilize existing matchings to accelerate new matchings or even enhance the accuracy. Despite the practical importance, little effort has been made to address this online setting, which is the focus of this paper.

In contrast to the vast literature on offline multi-graph matching [7–14], the paper is tailored to the online setting with the following contributions:

- To our best knowledge, this is one of the first works for addressing the problem of incremental matching of multiple graphs.
- Compared with the offline baselines, our method can achieve or even improve the matching accuracy while notably decreases the computing cost.
- We present interpretation to the proposed structure and mechanism, which can be treated as a more general framework to [6] and [7].

2 Related works

Due to its importance and fundamentality, extensive work on graph matching have been performed and thoroughly reviewed in a recent survey [15]. We categorize the representative work by the following perspectives.

2.1 Affinity function model

Graph matching incorporates both unary *node-to-node*, and second-order, or even higher-order, *(hyper)edge-to-(hyper)edge* structural similarities. In its traditional setting, whereby no higher-order affinity is considered [16–20], the two graph matching problem can be formulated as a quadratic assignment problem (QAP) [1], being well-known NP-complete [21]. More recent work on hypergraph matching further explores the higher-order (and mostly third-order) in affinity function modeling [22–26] at the cost of increased time and space complexity, whereby tensor marginalization is often used.

In contrast to the above methods whereby the affinity function is predefined regardless their order or attribute layer, learning is adopted to adapt the affinity function to different settings [16, 19, 5]. Supervised [5] and unsupervised [16] learning paradigm have been explored to improve the matching accuracy. However, either learning based or learning-free affinity function modeling cannot completely address the inherent ambiguity and noise in two-graph matching.

2.2 Solvers and techniques

In the past decades there have emerged various GM solvers. A large body of literature are devoted to the study of different relaxation techniques, in order to mitigate the hard combinatorial problem in nature. Typical relaxations include i)

doubly-stochastic relaxation on the matching matrix [17, 27, 28]; ii) Semidefinite-programming (SDP) [29, 30]; iii) spectral relaxations [27, 31]. From the optimization perspective, different continuation methods [17, 32, 33] are widely adopted.

Different from these continuous relaxation methods which involves a post-processing step to binarize the final result, several methods tend to directly compute the solution in discrete assignment space. Sampling based methods [34, 35] directly generate discrete solutions via Monte Carlo Sampling. More recently, [36] devises a tailored Tabu search for graph matching.

Our approach involves iteratively solving pairwise matching. We depart from the above techniques, and follow a binarization-free composition based technique [7] to derive pairwise matchings which has been proved efficient and effective.

2.3 Multi-graph matching

Beyond two-graph matching, an emerging line of work tend to address the matching problem for multiple graphs jointly. These works [37–39, 9–11, 40, 13, 41] are motivated by the fact that: i) in real cases, it is more often that multiple graphs are available and needed for matching; ii) the availability to a batch of graphs provides global information to enable robust model against local noise.

Employing an independent two-graph matching method for each pair of graphs may result in the so-called cycle-inconsistency issue. Consider one toy example, for graph $\mathcal{G}_i$, $\mathcal{G}_j$, $\mathcal{G}_k$ of equal size with no outlier. The three naive matching solutions $\mathbf{X}_{ij}$, $\mathbf{X}_{ik}$, $\mathbf{X}_{kj}$ obtained independently from each pair of graphs, may lead to cycle-inconsistency: $\mathbf{X}_{ij} \neq \mathbf{X}_{ik}\mathbf{X}_{kj}$ as illustrated in [9].

To address this issue, various multi-graph matching models are proposed incorporating the so-called consistency either in an iterative or a one-shot manner. We follow the survey [15] to categorize these works in the following two folds.

Iterative methods [10, 9, 6, 11, 7] seek to find a tradeoff between affinity and consistency scores. In general, the authors write out the objective for multi-graph matching by adding up pairwise affinity terms $\{\mathrm{vec}(\mathbf{X})^\top \mathbf{K} \mathrm{vec}(\mathbf{X})\}_{i,j=1}^N$, and the pairwise matchings $\{\mathbf{X}\}_{i,j=1}^N$ are iteratively updated whereby in each iteration the cycle-consistency constraints are strictly, or gradually satisfied. Specifically, [9, 6] enforce the strict cycle-consistency constraint $\mathbf{X}_{ij} = \mathbf{X}_{ib}\mathbf{X}_{bj}$ over the whole iterative variable updating procedure. As a result, the matching accuracy may degenerate due to the inherent sensitivity to the starting point and updating rotating order. Such idea is also employed in [10] for extending two-graph matching method – Graduated Assignment [17] to multi-graph case. In contrast, a more flexible and robust mechanism is devised in [11, 7], whereby the consistency is gradually added over iterations.

One-shot methods [39, 40] try to enforce overall consistency as a post-step given the putative two-graph matchings. However, since the affinity information is totally discarded in the consistency enforcement procedure, these methods suffer from the sensitivity to the quality of putative matchings as initialization. In [42], the first-order affinity is utilized together with the consistency constraint, to improve robustness on real images. However, higher-order information is neglected.

Based on the above two lines of methods, the authors in [12] present a formulation where the affinity among graphs is encoded by a matrix stacked by the vectorized attributes of graphs, and the variables are reduced to a set of non-redundant bases inherent free from the consistency constraint. In [13], a tensor representation based approach for multi-graph matching is presented.

However, all the aforementioned work ignore an important setting, whereby the graphs arrive in a sequential way and online matching is needed. In fact the problem of online or incremental matching of graphs is nontrivial and existing methods may not be readily generalized to handle this new problem.

To our best knowledge, this is one of the first works for explicitly addressing the graph matching problem in an online fashion, whereby an incremental matching technique is presented. Experimental results show our solver can notably outperform the traditional multi-graph matching methods.

3 Preliminaries

We consider the case of one-to-one bijection matching. This setting has its technical consideration as cycle consistency constraint requires sizes of graphs to be identical. While most existing multi-graph matching methods employ such setting [25, 11, 39, 42, 14], unbalanced graph sizes can be handled by introducing dummy nodes or slack variables as in [18]. In case of bijection, a matching between graph $\mathcal{G}_i$ and $\mathcal{G}_j$ can be expressed via a permutation matrix $\mathbf{X}_{ij} \in \{0,1\}^{n\times n}$, where n is the number of vertices in each graph. Given the affinity matrix $\mathbf{K}_{ij} \in \mathcal{R}^{n^2\times n^2}$ encoding the vertex and edge similarities on diagonal and off diagonal respectively (see more detailed definition for $\mathbf{K}_{ij}$ in prior art [6]), the matching objective between graph $\mathcal{G}_i$ and $\mathcal{G}_j$ can be compactly defined as:

$$\begin{aligned} &\max_{\mathbf{x}} \mathcal{E}_{ij} = \mathbf{x}_{ij}^T \mathbf{K}_{ij} \mathbf{x}_{ij} \\ &\text{s.t.} \quad \mathbf{H}\mathbf{x}_{ij} = \mathbf{1}, \mathbf{x}_{ij} \in \{0,1\}^{n^2} \end{aligned} \tag{1}$$

where $\mathbf{x}_{ij} = \mathrm{vec}(\mathbf{X}_{ij})$ is the column-wise vectorized formation of $\mathbf{X}_{ij}$. $\mathbf{H}$ is a selection matrix enforcing the matching to be one-to-one. Due to the combinatorial nature, to obtain the optimal solution for $\mathcal{E}_{ij}$ is in general NP-hard, and it is often relaxed into continuous domain by replacing the constraint $\mathbf{x}_{ij} \in \{0,1\}^{n^2}$ with $\mathbf{x}_{ij} \in [0,1]^{n^2}$ for efficient and approximate optimization [27, 31, 18].

Beyond two-graph matching, multi-graph matching has recently received extensive attention, which is a process of simultaneously finding pairwise matchings between all given graphs to maximize the overall affinity score $\mathcal{E} = \sum_{i,j} \mathcal{E}_{ij}$. A naive approach is to solve each two-graph matching problem by maximizing each $\mathcal{E}_{ij}$ independently. However, this can lead to a fact that one matching solution $\mathbf{X}_{ij}$ does not agree with another solution derived by an alternative path: $\mathbf{X}_{ij} = \mathbf{X}_{ik}\mathbf{X}_{kj}$. To address this, cycle consistency [7] is proposed to constrain the matching results to be cycle consistent, which emphasizes any two matching paths between graph $\mathcal{G}_i$ and $\mathcal{G}_j$ should derive similar correspondence. For efficiency and simplicity, in this paper we follow the widely used first-order consistency

over graph $\mathcal{G}_k$ as in [6, 7], which is defined as:

$$\mathcal{C}_k = 1 - \frac{\sum_i \sum_{j>i} \|\mathbf{X}_{ij} - \mathbf{X}_{ik}\mathbf{X}_{kj}\|_F / 2}{nN(N-1)/2} \in (0, 1] \tag{2}$$

where N is the number of graphs, and $\|\cdot\|_F$ refers to the Frobenius norm. This equation measures how much a direct matching agrees to another matching via an intermediate graph. Thus the overall consistency over all graphs becomes:

$$\mathcal{C} = \frac{1}{N} \sum_k \mathcal{C}_k \tag{3}$$

By adding $\mathcal{C}$ as a regularizer to affinity score, multi-graph matching with consistency regularization yields to optimize:

$$\mathcal{E}^{\text{mgm}} = \lambda \mathcal{E} + (1 - \lambda)\mathcal{C} \tag{4}$$

where λ is a controlling parameter. As this objective is significantly different to those which are optimized using analytical or gradient-based methods, a heuristic multi-graph matching method over $\mathcal{E}^{\text{mgm}}$ is introduced in [7], in which the matching in previous iteration is replaced by the product of two permutation matrices on the path $\mathbf{X}'_{ij} = \mathbf{X}_{ik}\mathbf{X}_{kj}$ if $\mathcal{E}^{\text{mgm}}$ ascends. For its effectiveness and efficiency, we employ this strategy in part of our method.

As an extension of offline multi-graph matching, incremental multi-graph matching tries to match the $N + 1$th arriving graph when previous matchings of N graphs have been computed. It is desired that one can reuse the previous matching results when one or a few new graphs arrive. A naive way is to take the previous solution for N graphs as starting point for iterative updating with the $N + 1$ graph no matter what existing multi-graph matching method is used, under the expectation that the results obtained from N graphs can serve as better initialization than random guess and speedup the convergence. However, one still has to compute all $N + 1$ matchings $\mathbf{X}_{ij}$ each time a new graph arrives.

4 Proposed approach

4.1 Concepts for hypergraph topology

Before going into the details, it is worthwhile to mention some definitions here. A hypergraph $\mathfrak{G}$ consists of multiple graphs and the pairwise matchings between them. Mathematically, a hypergraph is defined as (also sketched in Figure 1):

$$\mathfrak{G} = \{\{\mathcal{G}_1, ..., \mathcal{G}_N\}, \{\mathcal{M}_{ij}\}\}, \quad i, j \in \{1, ..., N\} \quad i \neq j \tag{5}$$

where $\mathcal{M}_{ij}$ is the matching between $\mathcal{G}_i$ and $\mathcal{G}_j$. For hypergraph, there is a complex that covers the corresponding structure. Given an index subset $u \subseteq \{1, ..., N\}$, we re-number the index as $u = \{l_1, ..., l_k\}$. Then a sub-hypergraph $\mathfrak{G}_u$ contains a subset of k graphs and its matchings induced by the index set u:

$$\mathfrak{G}_u = \left\{\{\mathcal{G}_{l_1}, ..., \mathcal{G}_{l_k}\}, \{\mathcal{M}_{l_i l_j}\}\right\}, \quad l_i, l_j \in u, \quad l_i \neq l_j \tag{6}$$

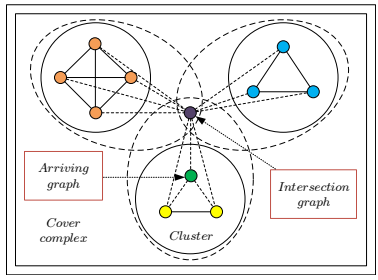


Fig. 1: The hypergraph with a petal-shape topology and its cover complex. There are 10 existing graphs and an arriving graph. Note that: i) graphs are re-clustered into k partitions overlapped by the intersection graph over iterations; ii) at each iteration, the intersection graph is re-selected from all graphs which is not necessarily the new graph; iii) the partitions are clustered by the criterion of randomness and diversity under a given cluster number (here $k = 3$).

According to the topology of hypergraph $\mathfrak{G}$ which is a fully connected structure, matchings are calculated between each pair of individual graphs. As the amount of such matchings is combinatorial to the number of graphs, it could be computationally intensive. However, if we can reduce the size of the hypergraph, the computational cost will be mitigated significantly. This fact motivates the idea to partition a hypergraph into several sub-hypergraphs, conduct optimization on each subset and merge all the matching results. Figure 1 demonstrates a general view of our method. We will detail the algorithm and give an intuitive interpretation in the following sections.

4.2 Algorithm details

As shown in Algorithm 1, a method called Incremental Multi-Graph Matching via randomness and diversity based graph clustering (abbr. **IMGM**) is proposed.

In general, the method consists of i) intersection graph selection and exclusion; ii) randomness and diversity based graph clustering without the intersection graph; iii) matching propagation along the intersection graph. These steps are performed iteratively and illustrated in Figure 1.

1) Selecting intersection graph shared by sub-hypergraphs. Applying multi-graph matching within each sub-hypergraph cannot guarantee the global cycle consistency, because there is no link between different sub-hypergraphs which also limit the effective use of global information across all the graphs. To establish the connection between sub-hypergraphs, we adopt the criterion in [6, 7] to select an intersection graph $\mathcal{G}_v$ (see illustration in Figure 1) by maximizing the consistency score according to Equation (2). Hence the reference graph becomes the intersection of all sub-hypergraphs by regarding it as belonging to each of the sub-hypergraphs. The resulting topology of the hypergraph is petal-shape, as shown in Figure 1 (note the resulting clusters are overlapped to each other

Algorithm 1 Incremental multi-graph matching via randomness and diversity based graph clustering (IMGM)

function IMGM($\mathbf{S}, \mathfrak{G}, \mathfrak{N}$) ▷ $\mathbf{S}$ - Similarity, $\mathfrak{G}$ - Hypergraph, $\mathcal{G}_N$ - New graph
  for each new graph $\mathcal{G}_N$: **do**
    $\mathfrak{G} = \mathfrak{G} \cup \{\mathcal{G}_N\}$
    Find intersection graph v by maximizing Equation (2) based on updated $\mathbb{X}$
    Update similarity $\mathbf{S}$ by Equation (1) based on updated $\mathbb{X}$
    Partition $\mathfrak{G} \setminus v$ into d clusters using k-DPP based on $\mathbf{S}$, or random partition
    for $u = 1, \ldots, d$ **do**
      Generate sub-hypergraph $\mathfrak{G}_u$ induced by u-th cluster
      $\mathfrak{G}_u = \mathfrak{G}_u \cup \{\mathcal{G}_v\}$
      Construct cover using $\mathfrak{G}_u, \quad u = 1, ..., U$
      Apply CAO$^{\mathrm{PC}}$ (or other multi-graph matching solvers) to obtain $\mathbb{X}_u$
    end for
    for $i \in \mathcal{I}_{u_1}, j \in \mathcal{I}_{u_2}, \quad i, j \neq v$ **do**
      $\mathbf{X}_{ij} = \mathbf{X}_{iv}\mathbf{X}_{vj}$ ▷ Construct length-two path through new graph
      $\mathbb{X} = \cup\mathbb{X}_u \cup \{\mathbf{X}_{ij}\}$
    end for
  end for
  return $\mathbb{X}$
end function

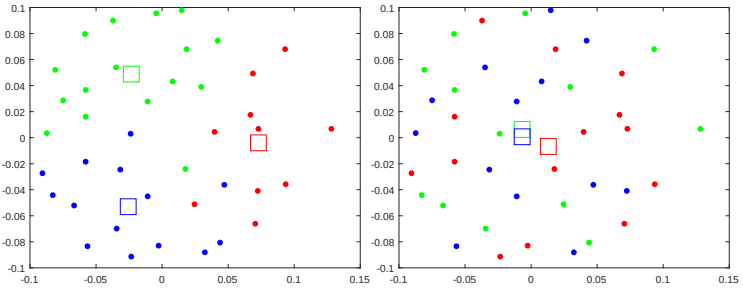
by the intersection graph). As a result, to obtain a cycle-consistent hypergraph $\mathfrak{G}$ for multi-graph matching, one only needs to consider the cycle consistency constraint within each sub-hypergraph.

2) Clustering sub-hypergraphs by randomness and diversity. We focus on two desirable properties for partitioning/clustering hypergraph in our setting: diversity (in iteration) and randomness (over iterations). We first describe the general idea as follows:

For the first property, traditional clustering methods partition data into several collections to aggregate *similar* objects and separate *dissimilar* ones. However, this strategy does not fit with our case, and the intuitive idea is that it is difficult to match two dissimilar graphs from two clusters via an intersection graph. Hence we adopt the policy that encourages diversity in each cluster, and the hope is that each cluster is representative for the whole hypergraph.

The second property, on the other hand, emphasizes the relative randomness of the partitions over iterations. If the partitions are fixed at each iteration, the optimization may fall into local optima and has less chance to escape. In this sense, the matching solution will converge in early iterations and not evolve along with the graph sequence. To introduce randomness, we expect that the partition can change to some extent from iteration to iteration, such that the heuristic procedure can explore more solution space.

Based on the above observations, we introduce two specific ways for partitioning at and over iterations: random sampling and determinantal point process (DPP) [43]. Since random sampling is trivial, we explain more about DPP partitioning in the following. DPP sampling relies on computation of similarity $\mathbf{S}_{ij}$ of



(a) Partition by spectral clustering (b) Partition by k-DPP

Fig. 2: Partitioning results using spectral clustering with *similarity* (left) and proposed DPP procedure with *diversity* (right) on 40 graphs. Graphs are converted to 2-D points by Multi-Dimensional Scaling [45] with pairwise affinity score. Square markers are the centroids of each cluster.

any pair of graphs i and j. To this end, we at first introduce the optimal energy $\mathbf{S}_{ij} = \mathcal{E}_{ij}$ as a measurement when $i \neq j$. For $i = j$, we let $\mathbf{S}_{ii} = 1.1 \times \max_{i,j,i\neq j} \mathcal{E}_{ij}$ without loss of generality. Then we let $\mathbf{S}_{ij} = \mathbf{S}_{ij} - \min_{i,j,i\neq j} \mathcal{E}_{ij}$. For N graphs with d partitions, we first compute the size of each partition by using N/d and rounding it properly. We then apply d times of k-DPP [44] to obtain the partitions. Readers are referred to [44] for more details about k-DPP. We visualize such partitioning strategies in Figure 2. This example contains 40 graphs and the 2-D points are obtained by Multi-Dimensional Scaling [45]. The square box corresponds to each cluster centroid. One can see the centroids in the left panel are more scattered than those in the right panel. The points within a cluster in the left panel are closer to each other, while the points within a partition in the right panel span the whole space as much as possible.

Remarks One may argue the adoption of dissimilarity based spectral clustering which can also generate scattered points in each cluster. However, this approach is deterministic causing the partition in each iteration frozen. Another alternative is using random sampling to form each cluster, while the drawback is the diversity in each cluster cannot be effectively ensured. In fact, DPP canensure the diversity in each cluster at each iteration as well as the randomness over iterations due to its inherent stochastic nature.

3) Propagating matching along the intersection. Specifically, for each cluster u with graph sample index set $\mathcal{I}_u$, by treating the reference graph as the intersection graph that forms the petal-shape topology we define a new objective for incremental matching:

$$\mathcal{E}^{\text{inc}} = \sum_{u=1}^{U} \left(\sum_{i,j \in \mathcal{I}_u} \lambda \mathcal{E}_{ij} + (1-\lambda)\mathcal{C}_u \right) \tag{7}$$

In fact, we can apply existing multi-graph matching algorithm e.g. the method called compositional affinity optimization with pairwise consistency (CAO^{PC}) in [7] over each sub-hypergraph independently to optimize this score function. Then we can obtain a fully-connected sub-hypergraph in the sense that each pair of graphs is matched in the sub-hypergraph. Concretely, for graph $\mathcal{G}_i$ and $\mathcal{G}_j$ from different sub-hypergraphs there is no direct link, we use the intersection graph $\mathcal{G}_k$ as intermediate node to generate a length-two matching $\mathbf{X}_{ij} = \mathbf{X}_{ik}\mathbf{X}_{kj}$ for $i \in \mathcal{I}_u, j \notin \mathcal{I}_u$. The optimal matchings for Equation (7), together with the generated matchings via the intersection graph, are denoted by $\mathbb{X} = \left\{\mathbf{X}_{ij}^*\right\}$ which can also be used as the starting point for matching next new graph.

Remarks Figure 1 demonstrates the topology generated from our algorithm. Each solid circle corresponds to a partition, while optimization is conducted within each dashed ellipse. In each dashed ellipse, we densely calculate matchings of each pair of graphs with consistency regularization.

4.3 Further discussion

Complexity The complexity of CAO^{PC} is $O(N^3n^3 + N^2\tau_{pair})$, where N and n are the numbers of the graphs and vertices, respectively. τ_{pair} refers to the complexity of the selected graph matching solver. In k-DPP sampling, an eigen-decomposition and a projection procedure are involved, with complexity $O(N^3)$ and $O(N^2)$, respectively. If the hypergraph is clustered into d partitions, and w.l.g. equal size for each partition, the complexity of CAO-PC step of IMGM becomes $O(N^3n^3/d^2 + N^2\tau_{pair})$. In this case, the complexity of k-DPP partitioning is $O(N^3/d^2)$. It is easily seen that the clustering and cover topology can reduce the complexity significantly. In general, the complexity of the proposed algorithm with k-DPP partitioning becomes $O(N^3n^3/d^2 + N^2\tau_{pair} + N^3/d^2)$.

Topological interpretation The complex concept for offline multi-graph matching has been discussed in [14], and the authors have proven that under the following conditions the hypergraph $\mathfrak{G}$ is cycle-consistent: 1) Each sub-hypergraph (not including $\mathfrak{G}$ itself) is cycle-consistent; 2) Each pair of sub-hypergraph is joint normal; 3) The cover complex of the hypergraph is topologically simply connected. Though the assumption in these statements is ideal, it still provides hints showing why our algorithm works. Since the proposed topological structure assures that, once cycle-consistency is reached out in each partition, the holistic cycle-consistency derived from intersection graph will be satisfied accordingly. Furthermore we provide an explicit strategy on how to construct such a topology which is not covered in [14].

Alternative topology First we observe that the topology of the proposed method can be viewed as a generalized versions of [6] and [7]. If the size of the partitions is the same as the number of graphs, our method becomes [6]. On the other hand, if there is only one partition, our method degenerates to [7]. In this sense, our topology leverages the previous two structures, thus can reach

out a good balance between accuracy and efficiency. Besides, other topologies sufficing the three conditions stated in previous paragraph can also be proposed, as long as the matching can be propagated through intersection graph (e.g., line or circle structure). We leave these variations to future work.

5 Experiments

Performance evaluation criteria We impose three popular measurements [6, 7] to evaluate the performance of algorithms: accuracy, affinity score and consistency (abbreviated as acc, scr and con, respectively). acc $= 1-\sum_{i,j}\|\mathbf{X}^*_{ij} - \mathbf{X}^{\mathrm{GT}}_{ij}\|_F^2/nN^2 \in [0,1]$ refers to the matching accuracy by comparing solution $\mathbf{X}^*_{ij}$ to ground-truth matching $\mathbf{X}^{\mathrm{GT}}_{ij}$. scr $= \frac{1}{N^2}\sum_{i,j}\frac{\mathrm{vec}(\mathbf{X}^*_{ij})^T\mathbf{K}_{ij}\mathrm{vec}(\mathbf{X}^*_{ij})}{\mathrm{vec}(\mathbf{X}^{\mathrm{GT}}_{ij})^T\mathbf{K}_{ij}\mathrm{vec}(\mathbf{X}^{\mathrm{GT}}_{ij})}$ calculates the overall affinity score. con $= \mathcal{C}$ referring to Equation 3. Note scr can be above 1 – recall the affinity function can be biased rendering an incorrect matching leads to a higher affinity score than the one by ground truth matching.

Comparing methods and settings As there is little existing multi-graph matching algorithm in an incremental fashion, we devise two baselines by extending the CAO$^{\mathrm{PC}}$ algorithm in [7] and Consistent Matching algorithm in [6]. To this end, we reuse the matching result $\mathbb{X}$ in the previous iteration as a starting point, and incorporate the arriving graph $\mathcal{G}_N$ in the current iteration for another round of batch based multi-graph matching. We term these two baselines **incremental CAO** and **incremental ConMatch**. We also employ **raw CAO**$^{\mathrm{PC}}$ which calculates matching from scratch without using the previous result. Permutation synchronization (**mSync**) [39], which processes the matchings to fulfill the cycle consistency, is also adopted for comparison. The proposed algorithms equipped with DPP and random sampling for graph clustering over iterations, are termed as **IMGM-D** and **IMGM-R**, respectively. All initial pairwise matchings are obtained using Reweighted Random Walk [18] which has been proved an effective and stable two-graph matching solver.

5.1 On synthetic random graph

We follow [7, 9] to implement tests on synthetic data, in which the pairwise affinity scores are randomly generated with Gaussion deformation perturbation. For each trial, a reference graph with node count $n_R = 10$ is created, by assigning a weight q^R_{ab} for edge (a, b) uniformly sampled from $[0, 1]$. Then a Gaussian noise $\mu \sim \mathcal{N}(0, \epsilon)$ is added as $q^D_{ab} = q^R_{ab} + \mu$ to generate a destination graph. The edge density of graphs is adjusted according to density parameter ρ. The edge affinity is thus calculated by $\mathbf{K}_{ac;bd} = \exp(-\frac{(q_{ab}-q_{cd})^2}{\sigma^2})$. We test different combinations of numbers of base graphs N_B and arriving graphs N_A. Three settings of such combinations is conducted as $(N_B, N_A) \in \{(25, 15), (30, 20), (35, 25)\}$. We keep to partition the hypergraph into 2 sub-hypergraphs in all tests (if not otherwise specified) and conduct 50 times for each test and calculate the average

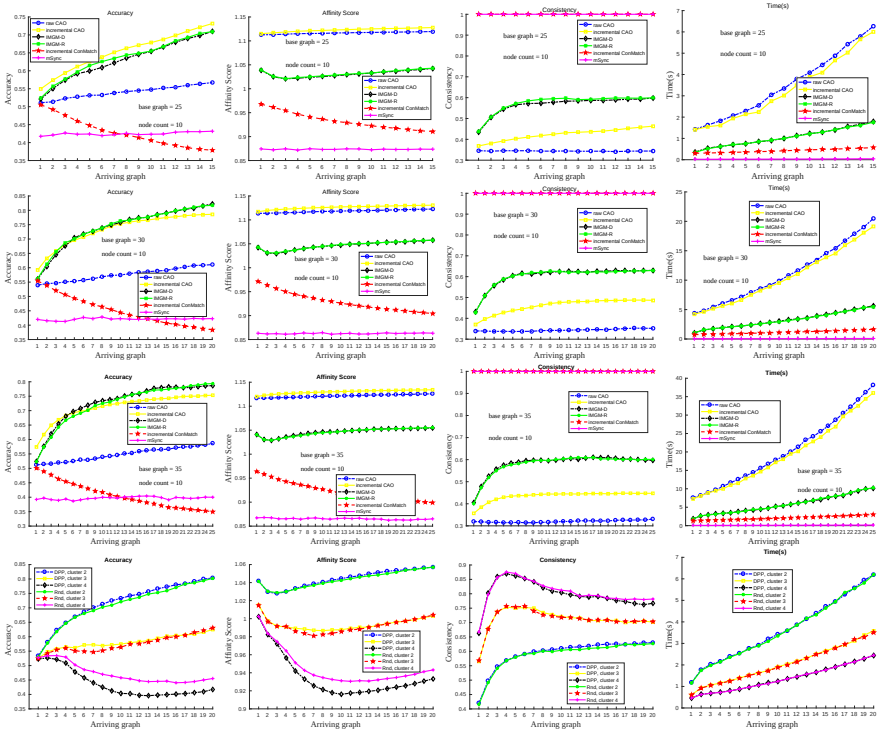
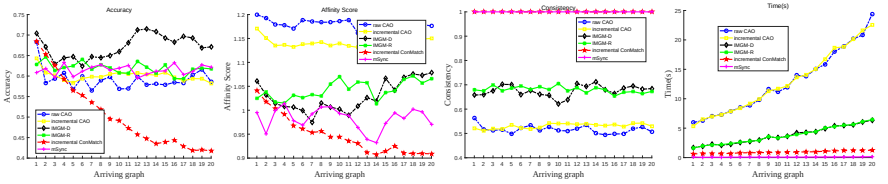


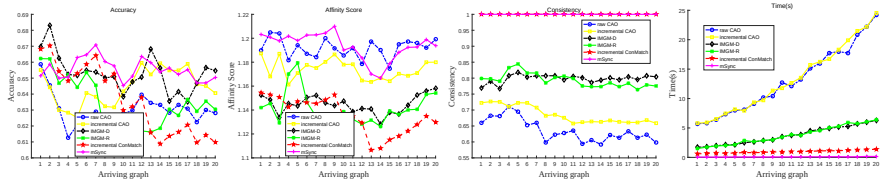
Fig. 3: Performance on synthetic data w.r.t. accumulated arriving graph count. Base graph number from top to bottom: 25, 30, 35, 30.

performance. We let $\rho = 0.9$, $\epsilon = 0.15$ and $\sigma^2 = 0.05$. The testing results are demonstrated in the first three rows of Figure 3. To evaluate of impact different partition size d, we conduct an additional test with $(N_B, N_A) = (30, 20)$ with $d \in \{2, 3, 4\}$. This test consists of 100 times of independent trials. The results for this additional test are shown in the last row of Figure 3, where DPP and Rnd correspond to IMGM-D and IMGM-R, respectively.

As one can see, when the size of base graphs is 25, our algorithm outperforms raw CAO in accuracy, and is very close to incremental CAO. When the size increases to 30, IMGM outperforms incremental CAO when more graphs arrive. In either case, raw CAO, incremental ConMatch and mSync have much lower accuracy. When there are 35 base graphs, both IMGM-D and IMGM-R outperform all other algorithms significantly, achieving state-of-the-art performance. Further, IMGM-D has a more stable accuracy growth along with arriving graphs. We can also observe that our algorithm reaches better global consistency than CAO-based methods. This is due to the fact that the matchings across sub-hypergraphs are generated via an intermediate graph, thus have higher consistency score. It should be noted that the matching accuracy of incremental



(a) Performance on CMU hotel



(b) Performance on CMU house

Fig. 4: Performance on CMU sequence w.r.t. accumulated arriving graph count.

ConMatch decreases along with graph sequence. This is because the topology of this algorithm does not evolve along with arriving graphs (there is always only one cluster), therefore randomness is missing from iteration to iteration. Notably, we also observe about 3 to 4 times of speedup compared to CAO algorithms. From the bottom row, one can observe that a larger cluster size d generally accelerates the computation and enhances the consistency. However, larger d results in accuracy drop – perhaps due to the cluster becomes too small to explore the joint matching effectively. Hence we need to find a trade-off between accuracy and efficiency controlled by d (see complexity analysis in Sec. 4.3). Last but not least, we also observe IMGM-D can often converge more quickly than IMGM-R, which compensates the additional overhead of performing DPP.

5.2 On real images: CMU sequence

The CMU pose dataset[1] consists of two image sequences. One is CMU house with 111 frames and 30 landmarks, and the other is CMU hotel with 101 frames and 30 landmarks. Each sequence contains an object gradually changing its poses. This dataset is widely tested in [18, 42, 7]. We use this dataset to evaluate the performance under partial similarity and outlier nodes. To this end, we follow the settings in [7] by selecting 10 inlier points in each frame, and randomly choose 4 points from the remaining landmarks in each frame rendering the matching more challenging. The graphs are constructing with sparse delaunay triangulation on both inliers and outliers following the method in [7]. The affinity is generated by $\mathbf{K}_{ac;bd} = \exp(-\frac{(q_{ab}-q_{cd})^2}{\sigma^2})$, where q_{ab} measures the Euclidean distance between point a and b normalized to $[0, 1]$ by dividing the largest edge length. The

[1] http://vasc.ri.cmu.edu/idb/html/motion/

diagonal elements of the affinity corresponding to node similarity are set to 0 as previous test. For each trail, we randomly sample $N_B = 30$ frames as base graphs, then randomly sample $N_A = 20$ frames from the remaining frames as arriving graphs. We conduct 20 times of single trial and calculate the average performance. Let $\sigma^2 = 0.05$. The performance curves are shown in Figure 4.

On the hotel sequence, IMGM-D method achieves significant accuracy superiority against the other compared algorithms. In the house sequence test, IMGM-D performs competitively to state-of-the-art algorithms. Though the change of relative positions of landmarks is not severe across graphs, the outlier points hinder the matching algorithms as disturbing edges may appear along with outliers. The stable performance demonstrates the robustness of the proposed algorithm, especially IMGM-D, against outliers. The algorithms show similar behavior as in synthetic test on the metrics other than accuracy. Thus we only present accuracy in the next Willow-ObjectClass data test.

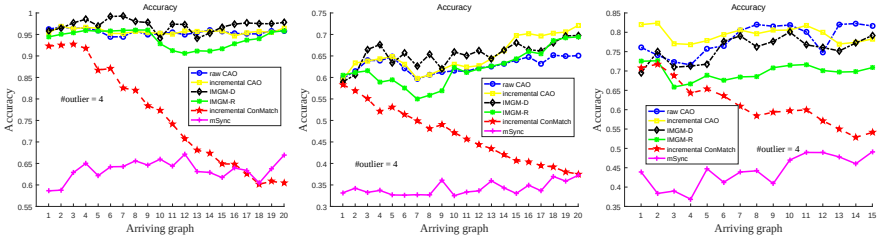
5.3 On real images: Willow-ObjectClass

The Willow-ObjectClass dataset is collected and released in [5], which consists of 5 classes of images collected from Caltech-256 and PASCAL07: 109 Face, 66 Winebottle, 50 Duck, 40 Car and 40 Motorbike images. All images are taken from natural scenes. For each image, 10 landmark points on the corresponding object are manually labelled. We select Winebottle, Duck and Car in our experiment with splits (N_B, N_A) of $(30, 20)$, $(30, 20)$ and $(25, 15)$, respectively. For Winebottle (or namely Bottle), as $N_B + N_A < 66$, we following the setting in previous test to randomly sample 50 images in each trial. For the resting two objects, we randomly permute all images. 4 SIFT feature points on the background are randomly selected as outliers. We still empoly sparse delaunay triangulation to construct the adjacency graph. Then the affinity is calculated as $\mathbf{K}_{ac;bd} = \beta\mathbf{K}^{\text{len}}_{ac;bd} + (1-\beta)\mathbf{K}^{\text{ang}}_{ac;bd}$ taking into account both edge length and angle similarity, where $\beta \in [0, 1]$ is a controlling parameter. While the definition of $\mathbf{K}^{\text{len}}_{ac;bd}$ is the same as used for the CMU sequence test, $\mathbf{K}^{\text{ang}}_{ac;bd}$ measures difference of the absolute angles between edge (a, b) and (c, d). The diagonal elements of affinity matrix are all set to 0 as before. $\beta = 0.9$ in this test. We conduct 20 times of independent trial and the mean accuracy is shown in Figure 5.

When there are more base graphs in Winebottle and Duck tests, IMGM-D outperforms the selected counterparts on most arriving graphs. On the other hand, in Car test with fewer base graphs, the proposed method gradually adapts the problem along with arriving graphs, and reaches competitive performance.

5.4 Discussion

We observe that the proposed method outperforms all peer algorithms when the size of graphs is sufficiently large. This may be due to the following two reasons. On one hand, the diversity in a sub-hypergraph must be sufficiently representative of the whole graph space, which is barely satisfied when the number of graphs is too small. On the other hand, the partitioning procedure is capable of



(a) Bottle (30 base graphs) (b) Duck (30 base graphs) (c) Car (25 base graphs)

Fig. 5: Accuracy on matching objects from Willow-ObjectClass dataset.

restricting the unreliability within a sub-hypergraph imposed by "outlier" graph, which may be propagated to all matching results if a whole batch optimization is employed. We also observe that, as the algorithm in [7] works on permutation matrices, if the partitions are stable along time, the matching solution after a period within each sub-hypergraph may fall into a permutation sub-group, and will not evolve any further. On the contrary, by imposing randomness in partitions, our algorithm can escape from poor local optima. Last but not least, we observe IMGM-D performs closely to IMGM-R on our synthetic test, and outperforms IMGM-R on all the real-world image tests. We conjecture this is due to the fact that for synthetic data, they are generated based on a shared base structure (with additional slight noise). However, the natural image data has a more complex distribution in their graph structure which can be better captured by diversity-based clustering.

6 Conclusion

In this paper, we present an incremental multi-graph matching approach called IMGM, which takes previous matchings and arriving graph as input, and performs optimization over a cycle-consistent topological structure. To the best of our knowledge, this is the first attempt to solve graph matching incrementally. We also analyze the functional topological structure of hypergraph and interpret the necessity of diversity and randomness in incremental settings. The proposed approach reduces the computational overhead, and improves the matching accuracy when there has accumulated enough graphs. Our paradigm is flexible to allow for the adoption other multi-graph matching methods as plugin.

Acknowledgement

This work was supported in part by a grant from ONR. Any opinions expressed in this material are those of the authors and do not necessarily reflect the views of ONR. Junchi Yan is supported by Tencent AI Lab Rhino-Bird Joint Research Program (No. JR201804) and NSFC 61602176.

References

1. Loiola, E.M., de Abreu, N.M., Boaventura-Netto, P.O., Hahn, P., Querido, T.: A survey for the quadratic assignment problem. EJOR (2007) 657–90
2. Wang, W., Lin, W., Chen, Y., Wu, J., Wang, J., Sheng, B.: Finding coherent motions and semantic regions in crowd scenes: a diffusion and clustering approach. In: ECCV. (2014)
3. Nie, W., Liu, A., Su, Y., Luan, H., Yang, Z., Cao, L., Ji, R.: Single/cross-camera multiple-person tracking by graph matching. Neurocomputing (2014)
4. Duchenne, O., Joulin, A., Ponce, J.: A graph-matching kernel for object categorization. In: ICCV. (2011)
5. Cho, M., Alahari, K., Ponce, J.: Learning graphs to match. In: ICCV. (2013)
6. Yan, J., Wang, J., Zha, H., Yang, X.: Consistency-driven alternating optimization for multigraph matching: A unified approach. IEEE Transactions on Image Processing (2015)
7. Yan, J., Cho, M., Zha, H., Yang, X., Chu, S.: Multi-graph matching via affinity optimization with graduated consistency regularization. TPAMI (2016)
8. Williams, M.L., Wilson, R.C., Hancock, E.: Multiple graph matching with bayesian inference. Pattern Recognition Letters (1997) 1275–1281
9. Yan, J., Tian, Y., Zha, H., Yang, X., Zhang, Y., Chu, S.: Joint optimization for consistent multiple graph matching. In: ICCV. (2013)
10. Sole-Ribalta, A., Serratosa, F.: Graduated assignment algorithm for multiple graph matching based on a common labeling. IJPRAI (2013)
11. Yan, J., Li, Y., Liu, W., Zha, H., Yang, X., Chu, S.: Graduated consistency-regularized optimization for multi-graph matching. In: ECCV. (2014)
12. Yan, J., Xu, H., Zha, H., Yang, X., Liu, H., Chu, S.: A matrix decomposition perspective to multiple graph matching. In: ICCV. (2015)
13. Shi, X., Ling, H., Hu, W., Xing, J., Zhang, Y.: Tensor power iteration for multi-graph matching. In: CVPR. (2016)
14. Hu, N., Thibert, B., Guibas, L.: Distributable consistent multi-graph matching. In: CVPR. (2018)
15. Yan, J., Yin, X., Lin, W., Deng, C., Zha, H., Yang, X.: A short survey of recent advances in graph matching. In: ICMR. (2016)
16. Leordeanu, M., Sukthankar, R., Hebert, M.: Unsupervised learning for graph matching. Int. J. Comput. Vis. (2012) 28–45
17. Gold, S., Rangarajan, A.: A graduated assignment algorithm for graph matching. TPAMI (1996)
18. Cho, M., Lee, J., Lee, K.M.: Reweighted random walks for graph matching. In: ECCV. (2010)
19. Caetano, T., McAuley, J., Cheng, L., Le, Q., Smola, A.J.: Learning graph matching. TPAMI **31**(6) (2009) 1048–1058
20. Egozi, A., Keller, Y., Guterman, H.: A probabilistic approach to spectral graph matching. TPAMI (2013)
21. Garey, M.R., Johnson, D.S.: Computers and Intractability; A Guide to the Theory of NP-Completeness. Freeman and Co., NY, USA (1990)
22. Zass, R., Shashua, A.: Probabilistic graph and hypergraph matching. In: CVPR. (2008)
23. Chertok, M., Keller, Y.: Efficient high order matching. TPAMI (2010)
24. Duchenne, O., Bach, F., Kweon, I., Ponce, J.: A tensor-based algorithm for high-order graph matching. TPAMI (2011)

25. Yan, J., Zhang, C., Zha, H., Liu, W., Yang, X., Chu, S.: Discrete hyper-graph matching. In: CVPR. (2015)
26. Ngoc, Q., Gautier, A., Hein, M.: A flexible tensor block coordinate ascent scheme for hypergraph matching. In: CVPR. (2015)
27. Leordeanu, M., Hebert, M.: A spectral technique for correspondence problems using pairwise constraints. In: ICCV. (2005)
28. Leordeanu, M., Hebert, M., Sukthankar, R.: An integer projected fixed point method for graph matching and map inference. In: NIPS. (2009)
29. Torr, P.H.S.: Solving markov random fields using semidefinite programmin. In: AISTATS. (2003)
30. Schellewald, C., Schnörr, C.: Probabilistic subgraph matching based on convex relaxation. In: EMMCVPR. (2005)
31. Cour, T., Srinivasan, P., Shi, J.: Balanced graph matching. In: NIPS. (2006)
32. Zaslavskiy, M., Bach, F.R., Vert, J.P.: A path following algorithm for the graph matching problem. TPAMI (2009)
33. Zhou, F., Torre, F.: Factorized graph matching. TPAMI (2016)
34. Lee, J., Cho, M., Lee, K.: A graph matching algorithm using data-driven markov chain monte carlo sampling. In: ICPR. (2010)
35. Suh, Y., Cho, M., Lee, K.M.: Graph matching via sequential monte carlo. In: ECCV. (2012)
36. Adamczewski, K., Suh, Y., Lee, K.: Discrete tabu search for graph matching. In: ICCV. (2015)
37. Solé-Ribalta, A., Serratosa, F.: Models and algorithms for computing the common labelling of a set of attributed graphs. CVIU (2011)
38. Huang, Q., Zhang, G., Gao, L., Hu, S., Butscher, A., Guibas, L.: An optimization approach for extracting and encoding consistent maps in a shape collection. ACM Transactions on Graphics (TOG) (2012)
39. Pachauri, D., Kondor, R., Vikas, S.: Solving the multi-way matching problem by permutation synchronization. In: NIPS. (2013)
40. Chen, Y., Leonidas, G., Huang, Q.: Matching partially similar objects via matrix completion. In: ICML. (2014)
41. Zhang, Q., Song, X., Shao, X., Zhao, H., Shibasaki, R.: Object discovery: Soft attributed graph mining. TPAMI **38**(3) (2016) 532–545
42. Zhou, X., Zhu, M., Daniilidis, K.: Multi-image matching via fast alternating minimization. In: ICCV. (2015)
43. Kulesza, A., Taskar, B., et al.: Determinantal point processes for machine learning. Foundations and Trends® in Machine Learning **5**(2–3) (2012) 123–286
44. Kulesza, A., Taskar, B.: k-dpps: Fixed-size determinantal point processes. In: ICML. (2011) 1193–1200
45. Mardia, K.V.: Some properties of clasical multi-dimesional scaling. Communications in Statistics-Theory and Methods **7**(13) (1978) 1233–1241