Layered Neighborhood Expansion for Incremental Multiple Graph Matching

Zixuan Chen[1,2], Zhihui Xie[1,2], Junchi Yan[1,2][0000−0001−9639−7679]⋆
Yinqiang Zheng[3], Xiaokang Yang[2]

[1] Department of Computer Science and Engineering, Shanghai Jiao Tong University
[2] MoE Key Lab of Artificial Intelligence, AI Institute, Shanghai Jiao Tong University
[3] National Institute of Informatics, Japan
{m13953842591,fffffarmer,yanjunchi,xkyang}@sjtu.edu.cn, yqzheng@nii.ac.jp

Abstract. Graph matching has been a fundamental problem in computer vision and pattern recognition, for its practical flexibility as well as NP hardness challenge. Though the matching between two graphs and among multiple graphs have been intensively studied in literature, the online setting for incremental matching of a stream of graphs has been rarely considered. In this paper, we treat the graphs as graphs on a super-graph, and propose a novel breadth first search based method for expanding the neighborhood on the super-graph for a new coming graph, such that the matching with the new graph can be efficiently performed within the constructed neighborhood. Then depth first search is performed to update the overall pairwise matchings. Moreover, we show our approach can also be readily used in the batch mode setting, by adaptively determining the order of coming graph batch for matching, still under the neighborhood expansion based incremental matching framework. Experiments on both online and offline matching of graph collections show our approach's state-of-the-art accuracy and efficiency.

Keywords: multi-graph matching, clustering, self-supervised learning

1 Introduction

Over the decades, to fuse the information among two or multiple graphs, the matching of graphs has attracted extensive attention in vision and learning communities. In general, graph matching (GM) aims to establish pairwise node correspondences over two or more graphs, whereby both cross-graph node-to-node and edge-to-edge affinity are considered. By incorporating geometrical edge information, the graph structure can be effectively explored resulting in better robustness against deformation and noise. This is in contrast to the node-wise information based matching models e.g. RANSAC [10] and Iterative Closet Point (ICP) [40]. However, it meanwhile lifts the node affinity based linear assignment

⋆ Corresponding author is Junchi Yan. Work was partly supported by National Key Research and Development Program of China 2018AAA0100704, NSFC (61972250, U19B2035), and SJTU Global Strategic Partnership Fund (2020 SJTU-CORNELL).

problem to the task of quadratic assignment programming (QAP) [22]. The global optimum for the former problem can be found in polynomial time by Hungarian method, while QAP is known NP-complete [12] in general. Hence, most existing graph matching methods often seek for approximate solutions.

Apart from its mathematical challenge, graph matching is also prone to suffer from the biased modeling of real-world data. In the presence of local noise over the two graphs for matching, the affinity model can be biased such that the mathematically global optimum may not correspond to the perfect matching for the data at hand. Considering the fact that in practice often a collection of graphs are available for matching, a natural idea is to perform joint matching of multiple graphs in one shot [31, 34]. The hope is that the local noise among individual graphs can be smoothed over. The so-called cycle-consistency enforcement has been a popular and useful technique which is based on the simple observation that the correct pairwise matching among three graphs form a closed loop.

We take one step further and consider the problem of incremental matching of graph stream over time. In online applications such as video analysis, and event log mining, graph data may be collected sequentially rather than in one shot. Therefore like many online variants of different algorithms, the community calls for effective and efficient online version of multiple graph matching. This setting is in fact rarely studied in literature until a recent work in [38].

This paper departures from the mainstream literature on offline multiple graph [25, 27, 31–33, 35]. Rather, we present a novel algorithm for incremental multiple graph matching (IMGM). The main contributions of the paper are:

i) We develop a novel and efficient IMGM solver based on maximum spanning tree on the super-graph (i.e. graph for matching is treated as a node) that gradually identifies and expands the neighborhood for the coming graph, by breadth first search to ensure the quality of the neighborhood, and then all the graphs are visited by depth first search for efficiency. In the procedure, the two-graph matchings are updated via matching composition which has been shown cost-effectiveness in previous works [31].

ii) By creating an adaptive order for sequential matching, we show that our method can be applied in the offline multiple graph matching setting effectively.

iii) Experiments on both synthetic data and real images clearly show the advantages of our method. In particular, it outperforms the state-of-the-art method [38] for incremental multiple graph matching remarkably, for both accuracy and efficiency. Our method also performs competitively in offline mode.

2 Related Works

There have emerged a series of surveys [1,7,11,29,36]. Our review divides existing works into matching of two-graph, multi-graphs and incremental matching.

Two-graph matching. Most works are devoted to the classic setting for two graph matching, which in its general form relates to QAP when up to second-order edge affinity is considered [2, 6, 9, 13, 20]. There are two ways of modeling the affinity matrix in QAP: i) learning-free models: using a fixed and parametric

form [17] e.g. Gaussian kernel to model the node-to-node, and edge-to-edge similarity. Due to the limited capacity of such simple affinity model, the complex landscape of the affinity information of real-world data may not be perfectly captured and as mentioned in the introduction part of this paper, the objective function for QAP can be biased which causes the difficulty for finding the correct matching in addition with the mathematical combinatorial problem itself. To improve the model capacity, one way is to lift the affinity matrix to higher-order (often third-order) tensor [4, 8, 23, 37, 39]. However the additional cost can be exponential in terms of the number of nodes in graph hurting their applicability.

Another seemingly promising direction is adopting learning to find appropriate parameters for affinity modeling rather simply determined by hand [16]. The key idea is to learn the affinity function either in supervised [2, 5, 20, 21] or unsupervised (or semi-supervised) setting [20, 21]. Deep neural networks are employed in recent works [30] to improve the model capacity, and also to enable end-to-end learning supervised by node correspondence information.

Offline and online multiple graph matching. Beyond two-graph, works [3, 15, 18, 24–27, 33, 34] tackle the problem of matching a collection of graphs.

For multiple graph matching, or more generally the correspondence problem from multiple views, the so-called cycle-consistency [33] has been a widely adopted concept and metric to evaluate the behavior of the matching solvers. Specifically, here we consider the three graphs G_i, G_j, G_k, and denote the two-graph matching as $\mathbf{X}_{ij}$, $\mathbf{X}_{jk}$, and $\mathbf{X}_{ik}$. Then for ground truth matching, the close loop will establish: $\mathbf{X}_{ik} = \mathbf{X}_{ij}\mathbf{X}_{jk}$, and the deviation signifies the inconsistency incurred by less accurate matchings. In another word, cycle-consistency is the necessary condition for perfect matching, and also an important indicator for the accuracy of matchings for multiple graphs.

We follow the protocol in [38] that categorizes existing multiple graph matching methods into two groups: i) two-stage methods that separate the local two-graph matching and post consistency based smoothing in two steps; ii) iterative methods that integrate the consistency and affinity optimization alternately.

In the two-step methods, first the pairwise matchings between two graphs or two node sets (the edge information may not be used) are first calculated by certain means, and then different smoothing techniques are devised to enforce cycle-consistency. Bijection and partial matching are respectively considered in the work [24] and [3]. One particular difficulty is that the number of cycles is exponential regarding with the number of graphs for matching. In a more recent study [14], the authors show that it is possible and theoretically guaranteed that only a subset of cycles need being sampled for joint optimization.

In contrast to the above post-smoothing methods, another line of study combine the optimization of both affinity and consistency in an iterative fashion. In general, it is difficult to incorporate the discrete consistency term for gradient based optimization. The authors in [31] devise a discrete composition based approach to generate new matching solution. The criterion refers to both affinity score and consistency improved by the new solution. In some other

works [27, 28, 33, 34], the consistency constraint is strictly obeyed which makes them less flexible, and more sensitive to error accumulation over iterations.

There are little study on the online setting whereby graphs arrive one by one. This setting is nontrivial and calls for efficient mechanism. We only identify one related work as directly termed by incremental multiple graph matching (IMGM) in [38]. The idea of [38] is to partition existing graphs into a few clusters, based on the criterion that each cluster shall maintain certain diversity as such the newly coming graph can be matched to only one cluster which is a good representative of the whole set. The diversity is fulfilled by the determinantal point process (DPP) [19] which sometimes can be more effective than random sampling.

Remarks. We argue that the success of the method in [38] is based on the assumption that a cluster can well cover the variation of the whole set of graphs, such that the new graph only need to match with the graphs in cluster. However it is difficult to ensure especially when the cluster size is small. In fact, there is a tradeoff between efficiency (small cluster) and effectiveness (representative cluster), and the clustering design in [38] essentially suffers such a dilemma and as shown in Fig. 3 (the bottom row) in [38]: its accuracy drops significantly by slightly increasing the cluster number k from 2 to 3 and 4.

We circumvent such a dilemma by avoiding explicit clustering of graphs which is time consuming and lacks flexibility to capture the complex variation of graph collection at hand. Instead, we propose to dynamically form a neighborhood based on breadth first search, allowing more easier and efficient matching for the new arrival graph. In other words, the involved graph subset for matching with the new graph is selected without any clustering step. Then we further use depth first search to cover all the nodes on the super-graph with compositional matching updating (akin to CAO-C [31]) to fulfill the whole matching process.

3 Preliminaries

Notations and definitions. In this paper, we use n for the number of nodes in each graph, N for the current number of graphs for processing, and M for the upper bound of the neighborhood size for a new graph, respectively. Furthermore, $L(N)$ denotes the number of two-graph matching performed when the N-th graph comes in a certain algorithm. This is useful for evaluating its complexity.

For bijection[4], the matching objective can be defined as [6]:

$$\begin{aligned} &\max_{\mathbf{x}} \mathcal{E}_{ij} = \mathrm{vec}(\mathbf{X}_{ij})^\top \mathbf{K}_{ij} \mathrm{vec}(\mathbf{X}_{ij}) \\ &\text{s.t.} \quad \mathbf{X}_{ij}\mathbf{1}_n = \mathbf{1}_n \quad \mathbf{1}_n^T \mathbf{X}_{ij} = \mathbf{1}_n^T \quad \mathbf{X}_{ij} \in \{0,1\}^{n\times n} \end{aligned} \tag{1}$$

where $\mathbf{X}_{ij}$ is the matching (permutation) matrix denoting node correspondences between graph G_i and G_j, and $\mathbf{K}_{ij}$ stands for the affinity matrix whose diagonal and offdiagonal elements encode the node-to-node and edge-to-edge affinity,

[4] We assume graphs are of equal size in this paper, which can be obtained by adding dummy nodes if needed as widely done in literature [6].

respectively [6,34]. To be self-contained, we also rewrite the definitions in terms of consistency metric as defined in [31,38] and other literature.

Definition 1. *Given N graphs $\mathbb{G} = \{G_k\}_{k=1}^{N}$ and the corresponding pairwise matchings $\mathbb{X} = \{\mathbf{X}_{ij}\}_{i=1,j=i+1}^{N-1,N}$, define the unary consistency of G_k as $C_u(k,\mathbb{X}) = 1 - \frac{\sum_{i=1}^{N-1}\sum_{j=i+1}^{N}\|\mathbf{X}_{ij}-\mathbf{X}_{ik}\mathbf{X}_{kj}\|_F/2}{nN(N-1)/2} \in (0,1]$.*

Definition 2. *Given $\mathbb{G}$ and their pairwise matchings $\mathbb{X}$, define the overall consistency as $C(\mathbb{X}) = \frac{\sum_{k=1}^{N} C_u(k,\mathbb{X})}{N} \in (0,1]$.*

Definition 3. *Given $\mathbb{G}$, pairwise matchings $\mathbb{X} = \{\mathbf{X}_{ij}\}_{i=1,j=i+1}^{N-1,N}$ and affinity matrices $\mathbb{K} = \{\mathbf{K}_{ij}\}_{i=1,j=i+1}^{N-1,N}$, the pairwise affinity score between G_i and G_j is defined as $S_{ij} = \mathrm{vec}(\mathbf{X}_{ij})^\top \mathbf{K}_{ij} \mathrm{vec}(\mathbf{X}_{ij})$.*

Definition 4. *A supergraph $\mathcal{H}$ is an undirected complete graph induced by graph set $\mathbb{G}$ and pairwise matchings $\mathbb{X}$. Its nodes, edges and edge weights denote graphs, pairwise matchings, and pairwise matching score, respectively:*

$$\mathcal{H} = \{V = \{G_k\}_{k=1}^{N}, E = \mathbb{X}, W = \{S_{ij}\}_{i,j=1}^{N,N}\}$$

Definition 5. *Given a supergraph $\mathcal{H}$, a path from G_i to G_j is denoted as $\mathcal{P}_{ij}$, which is a set containing edges from G_i to G_j: $\mathcal{P}_{ij} = \{\mathbf{X}_{ik_1}, \mathbf{X}_{k_1k_2}, ..., \mathbf{X}_{k_sj}\}$. The consistent matching matrix along path $\mathcal{P}_{ij}$ is defined as:*

$$\overline{\mathbf{X}}_{ij} = \mathbf{X}_{ik_1}\mathbf{X}_{k_1k_2}...\mathbf{X}_{k_sj}$$

And the affinity score along path $\mathcal{P}_{ij}$ is defined as:

$$\overline{S}_{ij} = \mathrm{vec}(\overline{\mathbf{X}}_{ij})^\top \mathbf{K}_{ij} \mathrm{vec}(\overline{\mathbf{X}}_{ij})$$

Definition 6. *Given a supergraph $\mathcal{H}$, a spanning tree $\mathcal{T} = \{V = \{G_k\}_{k=1}^{N}, E = \mathbb{X}' \subsetneq \mathbb{X}\}$ is a sub-supergraph, with $N-1$ edges and satisfies that each two vertices has exactly one path in $\mathcal{T}$. The spanning tree with the highest accumulated pairwise affinity score over all its edges is called maximum spanning tree (MST).*

Problem formulation. The incremental multiple graph matching problem can be summarized as follows: given the $N-1$ graphs $\{G_k\}_{i=1}^{N-1}$ with their matchings $\mathbb{X}_{N-1}$, how to match the new graph G_N and update the existing matchings $\mathbb{X}_{N-1}$ (if necessary) and generate the new overall matching set $\mathbb{X}_N$. In contrast, traditional multiple graph matching is performed in a way that involves all the graphs at offline. Directly applying such joint matching each time a new graph comes is intimidating. Thus, new methods are in demand to address such a nontrivial problem. Our setting is the same with the recent work [38].

4 IMGM by Layered Neighbor Expansion

4.1 Problem Analysis and Approach Overview

To reduce the computing overhead, one natural idea appearing in [38] is to divide the whole graph set $\mathbb{G}$ into k clusters $\mathbb{C}_1, \mathbb{C}_2, \ldots, \mathbb{C}_k$. Let the new coming graph G_N match one of these clusters $\mathbb{C}_i$ according to some similarity metric between the cluster and G_N, and the matchings between $\mathbb{C}_i \cup \{G_N\}$ and the rest clusters $\mathbb{G} \backslash \mathbb{C}_i$ can be generated by certain means e.g. cycle consistency enforcement.

However, we argue such a clustering-oriented strategy may not inherently fit with the online graph matching problem. The reasons include: i) clustering is not optimized for the purpose of finding the most similar (or representative as in [38]) graphs for matching; ii) clustering results can be unstable especially for $K > 2$ clusters e.g. using k-means, which means it is difficult for the resulting cluster to effectively fit with the coming graph for matching; iii) clustering can be time consuming, especially for large-scale and complex techniques e.g. k-DPP (determinantal point process) [19] as used in [38] to ensure the efficacy.

In this paper, we hold a principle to find graphs similar to the new coming one, and the matchings can be conducted in a diffusion manner on the supergraph (see Definition 4), such that the new graph is regarded as a seed and the pairwise matchings are generated over the diffusion path. For efficiency, we first build a maximum spanning tree (MST) (see Definition 6) on the existing supergraph $\mathcal{H}_{N-1}$ as such the pairwise matching can be focused on the tree which also avoids matching inconsistency due to the existence of matching cycles. As shown in Fig. 1(c) based on the formed MST, we first find the most similar graph G_k to G_N in $\mathbb{G}_{N-1}$ and perform breadth first search (BFS) starting from G_k until at most M graphs are reached on $\mathcal{H}_{N-1}$. The hope is that the most similar graphs to G_N can be effectively found along $\mathcal{H}_{N-1}$ such that matchings among these graphs can be more reliable and accurate. Given these accurate early matchings, we then devise a greedier and more cost-effective means to perform depth first search (DFS) to match the rest of graphs on $\mathcal{H}_{N-1}$. Note that DFS can help the matching be efficiently and consistently conducted for each pair of graphs along the search path by reusing the pairwise matching, as shown in Fig. 1(d)(e): after computing matching $\mathbf{X}_{ik}$ and $\mathbf{X}_{kj}$, the skip matching between G_i and G_j can be readily computed by $\mathbf{X}_{ij} = \mathbf{X}_{ik}\mathbf{X}_{ik}$ without redundantly computing $\mathbf{X}_{ij}$ between G_i and G_j, which can further cause cycle inconsistency (see Definition 1). In this way, the local consistency along each DFS path is guaranteed. It is also worth noting that directly using DFS instead of BFS can be efficient but less effective because the error can be accumulated over the DFS path. In contrast, the BFS strategy ensures better matching quality in the beginning, which allows for the aggressive DFS stage later.

We shall emphasize that the above steps are integrated as a whole based on the MST structure. Hence we cannot replace either the BFS stage or DFS stage with other techniques e.g. k-nearest neighbor, clustering. One may also notice that our DFS mechanism ensures the local consistency along each DFS path though the overall consistency cannot be guaranteed by our approach. We call

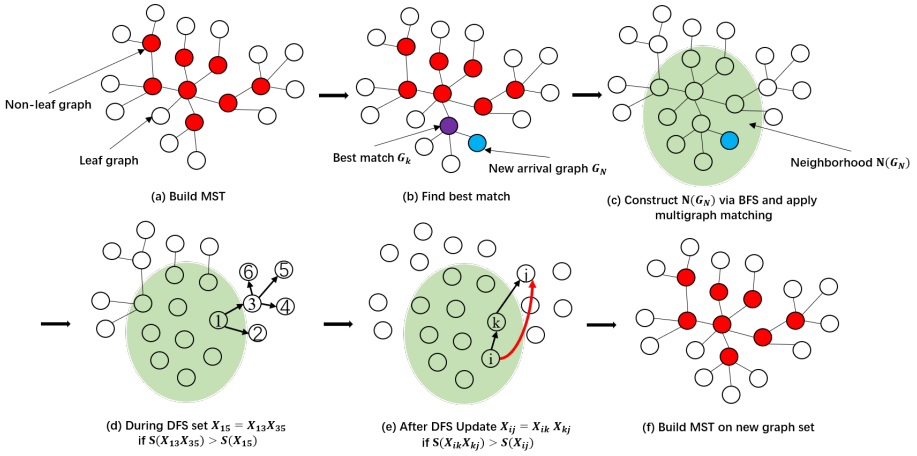


Fig. 1. Illustration of LNE-IMGM. (a) The red leaf nodes denote non-leaf graph set $\mathbb{C}$ and the white nodes denote leaf graph set $\mathbb{G}\backslash\mathbb{C}$. (b) The blue node denotes the new arrival graph G_N for incremental matching and the purple node denote the best match graph $G_k \in \mathbb{C}$. (c) The green ellipse denote the neighborhood $\mathcal{N}(G_N)$ with size $M = 10$. (d) We do DFS in numerical order and update the matchings during the search. (e) $G_i, G_k \in \mathcal{N}(G_N)$ and $G_j \in \mathbb{G}\backslash\mathcal{N}(G_N)$.

our approach **layered neighborhood expansion** since it involves the above starting graph finding, BFS and DFS to gradually expand the neighborhood.

An overview of our method is illustrated in Fig. 1. Compared with the traditional multi-graph algorithm with time complexity of $O(N^4)$, we reduce the complexity to $O(N^2)$ with two order of scalability improvement. We will cover each step of the algorithm in detail in the following sections.

4.2 Build MST and Find Best Match Graphing as Seed

We first build a Maximum Spanning Tree according to Definition 6. Denote D_i as the degree of node i on MST, and $E_{ij} = 1$ if node i and j are connected. It is clear that all non-leaf nodes have degree greater than 1, and all leaf nodes have only 1 degree. Suppose $\mathbb{C} = \{G_i, D_i > 1\}$ represents all non-leaf graph set. For each new graph G_N, it first matchings with every $G_i, i \in \mathbb{C}$. Let $k = \arg\max_{i\in\mathbb{C}} S_{iN}$ represent the best matched non-leaf graph. G_N is then linked to G_k on MST.

4.3 Neighbourhood Construction by BFS

On MST, all adjacent nodes are closely connected. To promote similarity, We construct neighbourhood $\mathcal{N}(G_N)$ by applying breadth first search (BFS) on MST rooted in G_N. Through BFS, we can not only search out the neighbourhood with graphs that are mostly similar to G_N, but also control the neighbourhood size. We denote $M = |\mathcal{N}(G_N)|$. After expansion, we perform multi-graph matching in $\mathcal{N}(G_N)$ to further optimize the matching.

Algorithm 1: Layered Neighborhood Expansion by BFS and DFS for Incremental MGM (LNE-IMGM)

Input: *//The only hyerparameter: size of the neighbourhood M*
1) Processed graphs $\mathbb{G}_{N-1} = \{G_1, ..., G_{N-1}\}$;
2) Super-graph $\mathcal{H}_{N-1}$ (Definition 4) by $N-1$ existing graphs $\mathbb{G}_{N-1}$, matchings $\mathbb{X}_{N-1}$ and affinity score $\mathbb{S}_{N-1} = \{S_{ij}\}(i, j = 1, ..., N-1)$;
3) New arrival graph G_N for matching;
if $N = 2$ **then**
  Perform pairwise matching and return $\mathbb{X}_N$ and $\mathbb{S}_N$;
Employ Prim algorithm to find Maximum Spanning Tree (Definition 6) over $\mathcal{H}_{N-1}$
// Build MST and Find Best Matching Graph:
Pairwise match G_N with each non-leaf graph $G_i \in \mathbb{C}$;
Find best match $G_k \in \mathbb{C}$ with maximum pairwise affinity score S_{Nk};
Connect G_N to the best matching graph G_k on MST;
// Neighbourhood Construction by BFS:
Apply breadth first search (BFS) on MST with root G_N, get neighbourhood $\mathcal{N}(G_N)$ with size at most M or early stop due to end of search;
Apply a multi-graph solver e.g. CAO-C [31] to obtain multi-graph matchings $\mathbb{X}_{bfs}$ on $\mathcal{N}(G_N)$;
Update $\mathbb{X}_N$ with $\mathbb{X}_{bfs}$;
// Local Optimization along each DFS path:
for *each* $G_i \in \mathcal{N}(G_N)$ **do**
  Set G_i as root node and apply depth first search within $\mathbb{G} \backslash \mathcal{N}(G_N)$;
  for *each* G_s *along DFS path* $\mathcal{P}_{is}$ **do**
    Update $\mathbf{X}_{is} = \overline{X}_{is}$, $S_{is} = \overline{S}_{is}$ if $\overline{S}_{is} > S_{is}$;
// Global Optimization along after DFS:
for *each* $G_i \in \mathcal{N}(G_N)$ **do**
  for *each* $G_j \in \mathbb{G} \backslash \mathcal{N}(G_N)$ **do**
    Find $G_k \in \mathcal{N}(G_N)$;
    Update $\mathbf{X}_{ij} = \mathbf{X}_{ik}\mathbf{X}_{kj}$, $S_{ij} = \overline{S}_{ij}$ if $\overline{S}_{ij} > S_{ij}$;
$\mathbb{G} = \mathbb{G} \cup \{G_N\}$;
Output: Multi-graph matchings $\mathbb{X}_N$ and $\mathbb{S}_N$

4.4 Whole Set Coverage and Matching by DFS

By expanding neighbourhood, we cover the whole graph set $\mathbb{G}$. For each graph in $\mathcal{N}(G_N)$, we perform depth first search(DFS) on MST, where it only search graphs in $\mathbb{G} \backslash \mathcal{N}(G_N)$. Assignment X_{rs} is updated to $\overline{\mathbf{X}}_{rs}$ along DFS path $\mathcal{P}_{rs}$ if $\overline{S}_{rs} > S_{rs}$ (see Definition 5).

After running DFS, some matchings between $\mathcal{N}(G_N)$ and $\mathbb{G} \backslash \mathcal{N}(G_N)$ are still not updated. To update those match between $G_i \in \mathbb{N}(G_N)$ and $G_j \in \mathbb{G} \backslash \mathcal{N}(G_N)$, suppose match between $G_k \in \mathcal{N}(G_N)$ and G_j has been updated during DFS, $\mathbf{X}_{ij}$ is updated to $\overline{\mathbf{X}}_{ij}$ along path $\mathcal{P}_{ij} = \{\mathbf{X}_{ik}, \mathbf{X}_{kj}\}$ if $\overline{S}_{ij} > S_{ij}$.

Algorithm 2: Layered Neighborhood Expansion for Multiple Graph Matching (LNE-MGM)

Input: 1) Graph streams $\mathbb{G}_N = \{G_1, G_2, ..., G_N\}$,
2) Pairwise affinity $\mathbf{K}_{ij}(i, j = 1, ..., N)$,
Do pairwise matching by RRWM to obtain initial $\mathbb{X}$;
Calculate affinity score for $G_i \in \mathbb{G}$;
Reorder graph streams $\mathbb{G}' = \{G'_1, G'_2, ..., G'_N\}$ by the graph-wise affinity score (descending order is suggested for better performance);
perform pairwise match between G'_1 and G'_2;
for $n = 3, 4, ..., N$ **do**
  Apply Alg. 1 to obtain updated $\mathbb{X}_n$and $\mathbb{S}_n$;
Output: Multi-graph matching $\mathbb{X}$

4.5 Adaptive Ordering for Batch of Graphs

LNE-IMGM can be applied offline by matching the graph batch one by one, i.e., in a pseudo online setting. Here we aim to improve this baseline strategy by reordering the graph sequence for matching. Specifically, the order is by the graph-wise affinity score over the graph set $\mathbb{G}$. Specifically, for graph G_i, its overall graph-wise affinity score is given by: $S_i = \sum_j S_{ij}$. For those with higher affinity score, we consider to match them first as we assume they are easier to match. Fig. 7(a),7(b) basically verifies our idea.

4.6 Time Complexity Analysis

The time complexity is compared in Table 1. We discuss each method in detail.

LNE-IMGM For each step, given $\mathbb{G}_{N-1} = \{G_1, G_2, ..., G_{N-1}\}$ as processed graphs with graph size n, and neighbourhood size M, we analyze the main overhead components as follows.

i) Two-graph Matching Solver. For pairwise matching process, we only match G_N with all non-leaf graphs $G_i \in \mathbb{C} \cup \mathcal{N}(G_N)$. Define the number of pairwise matching for G_N as $L(N)$, then $L(N) = |\mathbb{C} \cup \mathcal{N}(G_N)|$. Clearly $L(N) <= N$. The overall complexity for pairwise matching is $O(L(N)\tau_{pair})$, where τ_{pair} is the cost for calling a two-graph matching solver e.g. RRWM [6].

ii) Multi-graph Matching Solver. Our method can accept any multi-graph solver. In our experiment we use CAO-C which is the state-of-the-art MGM solver in [31] based on the compositional matching updating over pairwise matchings, whose time complexity is $O(M^4n + M^3n^3)$.

iii) Other Overhead. The time to compute MST is $O(N^2)$. The time for BST is $O(MN)$. The time for **L13-17** is $O((N - M)n^3)$. The complexity for **L18-21** in Alg. 1 is $O(M^2(N - M)n^3)$.

iv) Overall Complexity. The overall complexity can be reduced to $O(N^2 + M^2Nn^3 + M^4n + L(N)\tau_{pair})$. As M is constant for each step, the complexity thus can be roughly treated as $O(N^2 + Nn^3 + L(N)\tau_{pair})$.

Table 1. Time complexity for matching of IMGM per iteration. Note d is the number of clusters in [38]. It is often set to 2 to ensure accuracy, which limits [38]'s efficiency.

Method	Time complexity
CAO-C-INC [31]	$O(N^4n + N^3n^3 + N\tau_{pair})$
CAO-C-RAW [31]	$O(N^4n + N^3n^3 + N^2\tau_{pair})$
IMGM-D, IMGM-R [38]	$O(N^3n^3/d^2 + N^3/d^2 + N\tau_{pair})$
LNE-IMGM (ours)	$O(N^2 + M^2Nn^3 + M^4n + L(N)\tau_{pair})$
LNE-MGM (ours)	$O(N^3 + M^2N^2n^3 + NM^4n + N^2\tau_{pair})$

LNE-MGM The time to compute pairwise matching is $O(N^2\tau_{pair})$. The time to compute overall affinity score for each graph is $O(N^2n^3)$. The time to re-order graph stream is $O(N \log N)$. The overall complexity is $O(N^3 + M^2N^2n^3 + NM^4n + N^2\tau_{pair})$. As M is constant for each step, the complexity thus can be roughly treated as $O(N^3 + N^2n^3 + N^2\tau_{pair})$.

5 Experiments

Experiments run on a laptop with 3.2GHZ CPU and 16G memory. In synthetic data and real-world images, 50 trials and 20 trials are performed respectively, and the average are reported. For the plots, the settings are given in Table 2. The source code is public available at https://github.com/fffffarmer/LNE_IMGM.

5.1 Protocols and Compared Methods

Three popular evaluation metrics for multiple graph matching [31, 34] are used: accuracy (acc), affinity score (scr) and consistency (con, see Eq. 2). Accuracy is computed by comparing solution $\mathbf{X}^*_{ij}$ to ground-truth $\mathbf{X}^{\mathrm{GT}}_{ij}$:

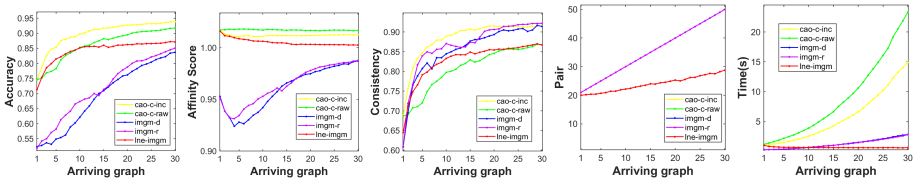
$$\mathrm{acc} = 1 - \sum_{i,j} \|\mathbf{X}^*_{ij} - \mathbf{X}^{\mathrm{GT}}_{ij}\|_F^2 / nN^2 \in [0, 1]$$

.

While the overall affinity score is calculated by:

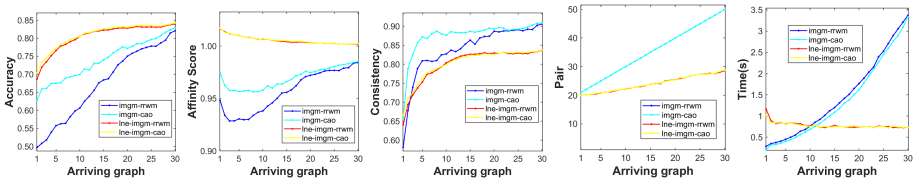
$$\mathrm{scr} = \frac{1}{N^2} \sum_{i,j} \frac{\mathrm{vec}(\mathbf{X}^*_{ij})^\top \mathbf{K}_{ij} \mathrm{vec}(\mathbf{X}^*_{ij})}{\mathrm{vec}(\mathbf{X}^{\mathrm{GT}}_{ij})^\top \mathbf{K}_{ij} \mathrm{vec}(\mathbf{X}^{\mathrm{GT}}_{ij})}$$

It is possible that scr > 1 as the affinity function may not perfectly reach the maximum at ground truth, especially in the presence of outliers. For comparison of time cost with previous works following [38], we leave the pairwise matching cost which is the multiplication of the number of runs $L(N)$ at step N and the unit time τ_{pair} for a calling of a certain two-graph matching solver e.g. RRWM. Then we define the time cost as (without two-graph



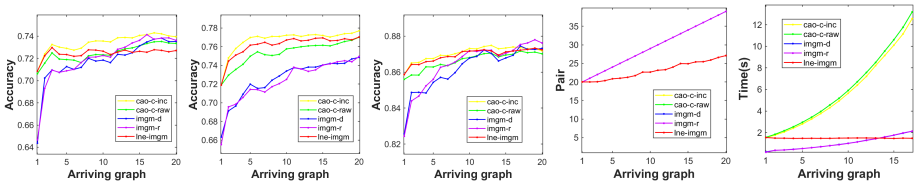
(a) Accuracy (b) Score (c) Consistency (d) Pair (e) Time

Fig. 2. Online evaluation on synthetic deformed graphs. LNE-IMGM is our method.



(a) Accuracy (b) Score (c) Consistency (d) Pair (e) Time

Fig. 3. Evaluation on using two-graph solver RRWM and multi-graph matching solver CAO-C for initialization of two IMGM methods.



(a) Duck (b) Car (c) Winebottle (d) Duck (e) Car

Fig. 4. Online evaluation on Willow-ObjectClass. CAO-C-INC means rerunning CAO-C initialized by the previous results by CAO-C. CAO-C-RAW denotes run from scratch.

matching part): $Time(N) = Total(N) - L(N)\tau_{pair}$. We separate $Total(N)$ into $Pair = L(N)\tau_{pair}$ and $Time(N)$ to have a more insightful study on overhead.

So far we have only identified one peer method [38] for multiple graph matching. This method has two variants termed as **IMGM-D** and **IMGM-R**, with DPP and random sampling for graph clustering over iterations, respectively. The number of clusters are set to 2 for these two methods in all tests (if not otherwise specified). In addition, we also follow the protocol in [38] to compare two additional baselines. One is the vanilla version of Composition Affinity Optimization with pairwise consistency method (CAO-C) [31], i.e., independently applying CAO-C every time a new graph comes with existing ones to form the new collection of graphs. The other is running CAO-C which is initialized by the matching results before the new graph's arrival. These two versions are termed as **CAO-C-RAW** and **CAO-C-INC**, respectively. In the experiments, all the initial two-graph matchings are computed using Reweighted Random Walk base Matching (RRWM) [6] as it has been proved effective and stable for two-graph

Table 2. Parameter details for online (top) and offline (bottom) tests. Note each plot in Fig. 5 contains synthetic and image data. RRWM is used as the two-graph matching solver. Affinity kernel $\sigma^2 = 0.05$, neighbor upper size $M = 20$, inlier # $n_i = 10$.

Results	Parameter settings
Fig. 2, 3, 5 (synthetic)	$n_o = 0$, $c = 1$, $\epsilon = 0.15$, $\rho = 1$, (N_A, N_B)=(20, 50)
Fig. 4, 5 (real)	$n_o = 4$, $\beta = 0.9$, (N_A, N_B)=(20, 40)
Fig. 7(a)	$n_o = 0$, $c = 1$, $\epsilon = 0.15$, $\rho = 1$, (N_A, N_B)=(20, 52)
Fig. 7(b)	$n_o = 4$, $\beta = 0.9$, (N_A, N_B)=(20, 52)
Fig. 7(c)	$n_o = 0$, $c = 1$, $\epsilon = 0.15$, $\rho = 1$, (N_A, N_B)=(30, 50)
Fig. 7(d)	$n_o = 4$, $\beta = 0.9$, (N_A, N_B)=(30, 50)
Fig. 6(d), 6(e)	$n_o = 0$, $c = 1$, $\epsilon = 0.15$, $\rho = 1$
Fig. 6(a), 6(b), 6(c)	$n_o = 4$, $\beta = 0.9$

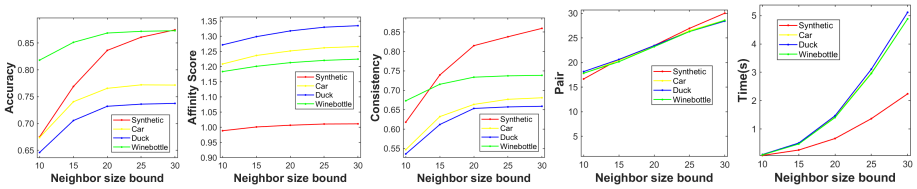
matching. To be fair, we set up CAO-C as the inner used multi-graph solver for all methods (**IMGM-D**, **IMGM-R**, **LNE-IMGM**, **LNE-MGM**).

5.2 Experiments on Synthetic Dataset

In line with the compared works [31, 38], synthetic graphs are generated for each trial. The details are as follows. First, a reference graph of size $n_R = 10$ is randomly created, whose edge weight q^R_{ab} for edge (a, b) is sampled from the uniform distribution $[0, 1]$. Then a derived graph is generated by adding a Gaussian noise sampled from $\mu \sim \mathcal{N}(0, \epsilon)$ on the weight: $q^D_{ab} = q^R_{ab} + \mu$. Meanwhile, a density parameter ρ is used to control the edge density of graphs. For two-graph matching which is adopted as a building block in our method and the compared algorithms, the pairwise edge affinity is set as $\mathbf{K}_{ac;bd} = \exp(-\frac{(q_{ab}-q_{cd})^2}{\sigma^2})$. We denote (N_B, N_A) as number of base graphs and number of arriving graphs. For N_B based graphs, we make two different initializations for matching the base graphs: by RRWM pairwise matching or by CAO multi-graph matching.

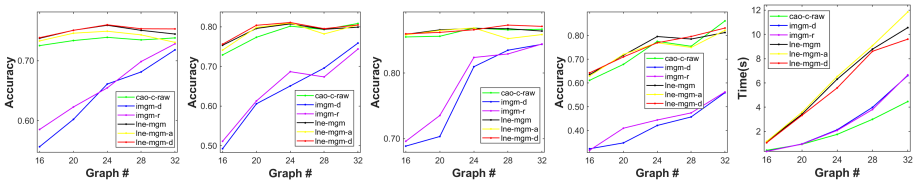
Online setting. We show the result of **RRWM–based** incremental matching result in Fig. 2. Our algorithm outperforms **IMGM-D** and **IMGM-R** in accuracy and affinity score, meanwhile is very close to **CAO-C-RAW** and its incremental updating version **CAO-C-INC**. Besides, our algorithm need much less times of pairwise matching in each step. In detail, the trend of *Pair* is a straight line $L(N) = N_B + kN_A$, where $k << 1$ in our algorithms. Most importantly, our method almost achieves constant in time, while the time taken by other algorithms increases significantly as more graphs arrive. Note that for each time a new graph comes, **CAO-C-RAW** has no online mechanism and we use it by repeatedly re-running it from scratch every time a new graph comes.

We also study the effects of two initialization strategies for **IMGM-D** and **LNE-IMGM**, i.e. adopting RRWM to match the base graphs and using a more advanced MGM solver CAO-C. In Fig. 3, our method is generally unaffected by initialization, while **IMGM-D** suffers changing from CAO-C to RRWM.



(a) Accuracy (b) Score (c) Consistency (d) Pair (e) Time

Fig. 5. Sensitivity test under **online IMGM** setting, for hyperparameter M i.e. the upper bound of expanded neighborhood size.



(a) Duck (b) Car (c) Winebottle (d) Synthetic (e) Synthetic

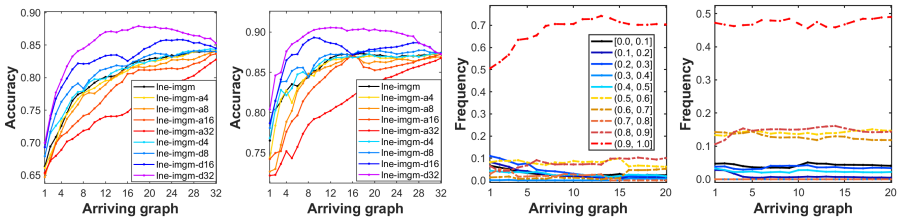
Fig. 6. Offline evaluation on synthetic graphs and Willow-ObjectClass. For our LNE-MGM, '-d' ('-a') denotes reordering is performed by the graph-wise affinity score in descending (ascending) order. Otherwise the matching order keeps by default.

We test the sensitivity on batch processing size and also with different ordering strategies. As shown in Fig. 7(a),7(b), the accuracy increases as we rank graph-wise affinity in descending order, which means gradually matching harder graphs can be a better strategy for overall matching accuracy. Meanwhile larger batch size can also lead to better accuracy which also well fits one's intuition.

Offline setting. Fig. 6 compares our offline versions (LNE-MGM-d and LNE-MGM-a) with CAO-based method CAO-C and IMGM_D and IMGM_R. The proposed method outperforms the baseline IMGM for synthetic random graphs, and performs competitively against CAO. IMGM-based algorithms are highly susceptible to the number of base graphs, resulting unsatisfactory performance in offline setting, while our approach performs more robustly, and even running from scratch without any base graph, it can still achieve strong result. Also, by using descending order, our algorithm gains further improvement.

5.3 Experiments on Real Image Dataset

We first test on the Willow-ObjectClass as provided and released by [5]. It consists of 5 classes of natural images collected from Caltech-256 and PASCAL07: 109 Face, 66 Winebottle, 50 Duck, 40 Car and 40 Motorbike images. There are 10 landmark points which are manually tagged on the object in each image. For object Duck and Car, we randomly permute all images. To verify the robustness of compared methods, four outliers are randomly chosen from the background. Delaunay triangulation is used to sparsify the graphs to construct the adjacency



(a) Synthetic (b) Winebottle (c) Synthetic (d) Car

Fig. 7. (a)(b): Sensitivity test for the batch processing size and ordering strategies on the synthetic deform graphs and Willow-ObjectClass Winebottle. The suffix '-a4'/'-d4' denotes the batch of 4 graphs are received in ascending/descending order by each graph's overall affinity score. Likewise for '-a8', '-d32' etc. The default is random ordering with batch size 1 which fits exactly the case for IMGM. (c)(d): Distribution of graph pair's accuracy over iterations by LEN-IMGM on synthetic graphs and Car. Different colors denote accuracy ranges [0,1] by step 0.1.

graph and the resulting affinity matrix, which is calculated as $\mathbf{K}_{ac;bd} = \beta\mathbf{K}^{\text{len}}_{ac;bd} + (1-\beta)\mathbf{K}^{\text{ang}}_{ac;bd}$. Here both edge length and angle similarity, where $\beta \in [0,1]$ is a controlling parameter regarding with the weight.

Online setting. When there are more arriving graphs in Winebottle and Duck tests, IMGM-D outperforms the counterparts on most arriving graphs as shown in Fig. 4. Meanwhile, in Car test with fewer arriving graphs, the proposed method gradually adapts the problem along with the new graphs, and achieves strong accuracy. Moreover we use Fig. 7(c),7(d) to show the pairwise matching accuracy distribution over iterations by LNE-IMGM for online MGM.

Offline setting. As shown in Fig. 6, on Willow-ObjectClass, our methods outperform or perform competitively against the offline solver CAO-C. Among the three variants of our method, the ascending reordering version performs generally best, suggesting the strategy for handling easier graphs first is useful.

6 Conclusions

This paper has presented a novel method for incremental matching of sequentially arriving graphs. For each coming new graph, its neighborhood is expanded in two steps which allows for cost-effective matching updating. This is in contrast to the peer method [38] adopting clustering which is complicated and inefficient. Our method can also serve as an effective offline multi-graph matching solver by sequential matching, except a treatment on determining the processing order by graph-wise affinity score. Experiments show its state-of-the-art accuracy and efficiency on synthetic graphs and images. Future work will explore machine learning especially deep networks for incremental graph matching.

References

1. Bunke, H.: Graph matching: theoretical foundations, algorithms, and applications. In: Vision Interface (2000)
2. Caetano, T., McAuley, J., Cheng, L., Le, Q., Smola, A.J.: Learning graph matching. TPAMI **31**(6), 1048–1058 (2009)
3. Chen, Y., Guibas, L., Huang, Q.: Near-optimal joint object matching via convex relaxation. In: ICML (2014)
4. Chertok, M., Keller, Y.: Efficient high order matching. TPAMI (2010)
5. Cho, M., Alahari, K., Ponce, J.: Learning graphs to match. In: ICCV (2013)
6. Cho, M., Lee, J., Lee, K.M.: Reweighted random walks for graph matching. In: ECCV (2010)
7. Conte, D., Foggia, P., Sansone, C., Vento, M.: Thirty years of graph matching in pattern recognition. IJPRAI (2004)
8. Duchenne, O., Bach, F., Kweon, I., Ponce, J.: A tensor-based algor ithm for high-order graph matching. TPAMI (2011)
9. Egozi, A., Keller, Y., Guterman, H.: A probabilistic approach to spectral graph matching. TPAMI (2013)
10. Fischler, M.A., Bolles, R.C.: Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. Commun. ACM (1981)
11. Foggia, P., Percannella, G., Vento, M.: Graph matching and learning in pattern recognition in the last 10 years. IJPRAI **33**(1), 1450001 (2014)
12. Garey, M.R., Johnson, D.S.: Computers and Intractability; A Guide to the Theory of NP-Completeness (1990)
13. Gold, S., Rangarajan, A.: A graduated assignment algorithm for graph matching. TPAMI (1996)
14. Guibas, L.J., Huang, Q., Liang, Z.: A condition number for joint optimization of cycle-consistent networks. In: NeurIPS (2019)
15. Hu, N., Huang, Q., Thibert, B., Guibas, L.J.: Distributable consistent multi-object matching. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 2463–2471 (2018)
16. Hu, N., Rustamov, R.M., Guibas, L.: Graph matching with anchor nodes: A learning approach. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 2906–2913 (2013)
17. Hu, N., Rustamov, R.M., Guibas, L.: Stable and informative spectral signatures for graph matching. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 2305–2312 (2014)
18. 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)
19. Kulesza, A., Taskar, B., Liu, L.: Determinantal point processes for machine learning. Foundations and Trends R in Machine Learning (2012)
20. Leordeanu, M., Sukthankar, R., Hebert, M.: Unsupervised learning for graph matching. IJCV (2012)
21. Leordeanu, M., Zanfir, A., Sminchisescu, C.: Semi-supervised learning and optimization for hypergraph matching. In: ICCV (2011)
22. Loiola, E.M., de Abreu, N.M., Boaventura-Netto, P.O., Hahn, P., Querido, T.: A survey for the quadratic assignment problem. EJOR (2007)

23. Ngoc, Q., Gautier, A., Hein, M.: A flexible tensor block coordinate ascent scheme for hypergraph matching. In: CVPR (2015)
24. Pachauri, D., Kondor, R., Vikas, S.: Solving the multi-way matching problem by permutation synchronization. In: NIPS (2013)
25. Shi, X., Ling, H., Hu, W., Xing, J., Zhang, Y.: Tensor power iteration for multi-graph matching. In: CVPR (2016)
26. Solé-Ribalta, A., Serratosa, F.: Models and algorithms for computing the common labelling of a set of attributed graphs. CVIU (2011)
27. Solé-Ribalta, A., Serratosa, F.: Graduated assignment algorithm for multiple graph matching based on a common labeling. IJPRAI (2013)
28. Swoboda, P., Mokarian, A., Theobalt, C., Bernard, F., et al.: A convex relaxation for multi-graph matching. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 11156–11165 (2019)
29. Vento, M.: A long trip in the charming world of graphs for pattern recognition. Pattern Recognition **48**(2), 291–301 (2015)
30. Wang, R., Yan, J., Yang, X.: Learning combinatorial embedding networks for deep graph matching. In: ICCV (2019)
31. Yan, J., Cho, M., Zha, H., Yang, X., Chu, S.: Multi-graph matching via affinity optimization with graduated consistency regularization. TPAMI (2016)
32. Yan, J., Li, Y., Liu, W., Zha, H., Yang, X., Chu, S.: Graduated consistency-regularized optimization for multi-graph matching. In: ECCV (2014)
33. Yan, J., Tian, Y., Zha, H., Yang, X., Zhang, Y., Chu, S.: Joint optimization for consistent multiple graph matching. In: ICCV (2013)
34. Yan, J., Wang, J., Zha, H., Yang, X., Chu, S.: Consistency-driven alternating optimization for multigraph matching: A unified approach. IEEE Transactions on Image Processing **24**(3), 994–1009 (2015)
35. Yan, J., Xu, H., Zha, H., Yang, X., Liu, H., Chu, S.: A matrix decomposition perspective to multiple graph matching. In: ICCV (2015)
36. Yan, J., Yin, X., Lin, W., Deng, C., Zha, H., Yang, X.: A short survey of recent advances in graph matching. In: ICMR (2016)
37. Yan, J., Zhang, C., Zha, H., Liu, W., Yang, X., Chu, S.: Discrete hyper-graph matching. In: CVPR (2015)
38. Yu, T., Yan, J., Liu, W., Li, B.: Incremental multi-graph matching via diversity and randomness based graph clustering. In: ECCV (2018)
39. Zass, R., Shashua, A.: Probabilistic graph and hypergraph matching. In: CVPR (2008)
40. Zhang, Z.: Iterative point matching for registration of free-form curves and surfaces. IJCV (1994)